2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

Resource-Constrained Robotic Planning in the face of Mixed Uncertainty

Yihao Yin[1,2,3], Pian Yu[4], Andrea Turrini[2], Zhiming Chi[2,3], Yong Li[2], and Lijun Zhang[2,3]

***Abstract*— Robots operate under significant uncertainty, from quantifiable noise to unquantifiable unknowns, and must account for strict operational constraints, such as limited resources. In this paper, we consider the problem of synthesizing robust strategies to guide a robot's actions in fulfilling a given task, while ensuring the system never exhausts its resources. To solve this problem, we first model the robotic system as a Consumption Markov Decision Process with Set-valued Transitions (CMDPST), a unified framework modelling nondeterministic actions, quantifiable and unquantifiable uncertainty, and resource consumption. Then, we combine the CMDPST with the task specification, expressed as a Linear Temporal Logic over finite traces (LTL_f) formula. Lastly, we address the resource-constrained optimal robust strategy synthesis problem, which aims to synthesize a strategy that maximizes the probability of satisfying the LTL_f objective without resource exhaustion.**

Our solution involves two techniques: a direct unrolling-based method and a more efficient, optimized approach that leverages state-space pruning for better performance. Experiments on a warehouse transportation network show the effectiveness of the proposed solutions.

I. Introduction

Autonomous systems are increasingly adopted in safety-critical domains, from autonomous driving [1], [2] and planetary exploration [3], [4] to medical robotics [5]. A core challenge in these systems is decision-making under uncertainty while ensuring strict adherence to safety and operational constraints, such as limited energy, memory, or computational resources. These systems are often modeled as stochastic discrete-time dynamical systems tasked with achieving complex goals, and must guarantee not only the satisfaction of such goals but also the continuous maintenance of critical safety conditions during execution [6].

Markov Decision Processes (MDPs) have emerged as a standard formalism for modeling uncertain systems, thanks to their ability to capture probabilistic outcomes and sequential decision-making [7], [8]. However, traditional MDPs are deficient in directly encoding hard resource constraints or environmental uncertainties [9]–[11]. Recent extensions such as resource-constrained MDPs [12], [13] have sought to address these limitations by incorporating cost bounds or modeling adversarial uncertainties. Yet, most of these approaches focus on expected cost optimization and assume known probabilistic models, which limits their applicability in settings where the environment is uncertain or adversarial and safety guarantees are paramount.

*Corresponding author: pian.yu@ucl.ac.uk, liyong@ios.ac.cn
[1]Hangzhou Institute for Advanced Study (HIAS), UCAS, China.
[2]Key Laboratory of System Software (Chinese Academy of Sciences), Institute of Software Chinese Academy of Sciences, Beijing, China.
[3]University of Chinese Academy of Sciences, Beijing, China.
[4]University College London, London, UK.

Concurrently, formal languages like Linear Temporal Logic (LTL) [14] and its finite-trace variant LTL_f [15] have proven to be powerful tools for specifying high-level mission objectives in autonomous systems. These specifications allow expressing rich temporal requirements such as "eventually deliver the package to location B while avoiding obstacle zones" or "do not deplete fuel before reaching the charging station". By combining MDPs with LTL and LTL_f, recent work has enabled efficient task planning in stochastic systems with temporal objectives [16]–[20]. However, these methods often do not account for resource constraints. In addition, in many real-world robotic scenarios, uncertainty stems from both quantifiable sources, like sensor noise, and unquantifiable ones, such as adversarial environments or human error [21]. These aspects cannot be adequately captured by ordinary MDPs, which assume precisely specified probabilistic transitions.

To address these challenges, we propose in this work *Consumption Markov Decision Processes with Set-valued Transitions* (CMDPSTs), a unified framework modelling nondeterministic actions, quantifiable and unquantifiable uncertainty, and resource consumption. This model is built upon *Markov Decision Processes with Set-valued Transitions* (MDPSTs) [22]–[25]. MDPSTs generalize MDPs by allowing transitions to be defined as sets of possible successor distributions, thereby capturing both quantifiable (e.g., stochastic) and unquantifiable (e.g., adversarial or nondeterministic) uncertainties. In particular, probabilistic weights in MDPST capture quantifiable uncertainty, while set-valued transitions model unquantifiable uncertainty without assuming precise probability distributions. Recent work has demonstrated the effectiveness of MDPSTs in robust planning with temporal-extended objectives [25]. To further capture the notion of limited and renewable resources, we extend the MDPST formalism by introducing "resource levels", "reload states", and "capacity bounds", allowing one to model systems that must periodically replenish critical resources to safely complete their tasks.

In this work, we propose to formulate the resource-constrained robust planning problem for robotic systems in the face of mixed (quantifiable and unquantifiable) uncertainty as the robust strategy synthesis problem for CMDPSTs. In addition, LTL_f is employed as a concise and compact specification language.

The main contributions of this paper are summarized as follows. i) We introduce CMDPSTs (a unified modelling framework) and formulate a novel *resource constrained optimal robust strategy synthesis* problem, which aims to compute a strategy that optimizes LTL_f satisfaction while

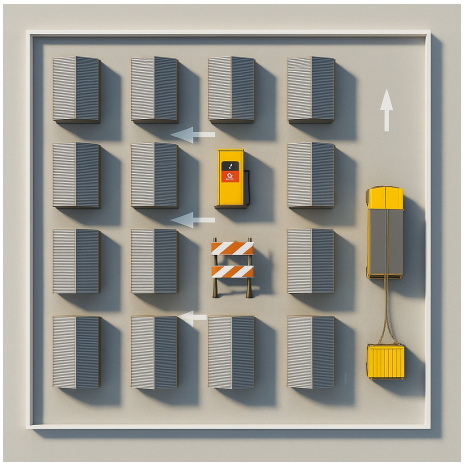

Fig. 1. Warehouse network

preventing resource exhaustion. ii) A naïve approach is first proposed to solve the problem by converting the CMDPST into an unrolled MDPST and applying an existing algorithm. To improve efficiency, we then introduce an optimization that prunes the CMDPST prior to unrolling, significantly reducing the state space required for synthesis. Experimental results show that our pruning method consistently achieves notable runtime improvements over the naïve approach, all while preserving the correctness of the synthesized strategy.

The remainder of the paper is organized as follows. Section II presents a motivating example illustrating the challenges of resource-constrained planning under mixed uncertainty. Section III reviews preliminaries on LTL$_f$ and MDPSTs. Section IV formalizes Consumption MDPSTs. Section V describes the naïve unrolling-based synthesis pipeline. Section VI develops the optimized approach. Section VII reports experimental evaluations on the warehouse benchmarks. Section VIII concludes the paper.

II. Motivating Example

As an example of a Consumption MDPST, we consider a warehouse transportation scenario where an automated guided vehicle (AGV) is required to transport goods across multiple warehouses; see Fig. 1 for an example. Within this warehouse network, we consider three typical situations: 1) some warehouses are temporarily inaccessible due to construction or maintenance (obstacle states); 2) some warehouses are equipped with charging stations (reload states), where the AGV can replenish its battery; and 3) certain warehouses are designated as task-related locations, such as pickup, transfer, and delivery points. For simplicity, we assume that each warehouse is of exactly one of these types.

The AGV is required to accomplish a transportation task with temporal constraints such as reach the warehouse W_p to pick up goods, move to warehouse W_t for transfer, or deliver the goods to the target warehouse W_d, while always avoiding obstacle warehouses W_o.

Each movement of the AGV (forward, backward, left, or right) consumes a certain amount of energy. Since the distance between warehouses may be long, the AGV must carefully plan its route to avoid energy depletion, which may require visiting reload warehouses (charging stations). Meanwhile, external disturbances may occur: for instance, a human operator may intervene and manually redirect the AGV, causing it to deviate from the planned route. Thus, the AGV's motion is subject to both stochastic uncertainty (e.g., execution noise) and nondeterministic disturbances (e.g., human intervention).

Unlike simple shortest-path planning, this scenario requires the AGV to satisfy a *temporal-extended task*, which can be naturally formalized by formal languages like LTL$_f$. The strategy controlling the AGV should not merely identify a shortest route to the final destination, but it must also respect resource constraints. Our goal is to compute robust strategies that guarantee the AGV never runs out of energy, while maximizing the probability of successfully completing the temporal task despite environmental or human-induced disturbances.

III. Preliminaries

In this section we first briefly recall Linear Temporal Logic on Finite Traces (LTL$_f$) [15], [26] and its equivalent deterministic finite automata representation, and then the Markov Decision Processes with Set-valued Transitions (MDPSTs) modelling framework [22], [23].

A. LTL$_f$

LTL$_f$ [15] has the same syntax as LTL [14], but is interpreted on *finite* yet unbounded horizons. The syntax of an LTL$_f$ formula over a finite set of atomic propositions AP is defined as

$$\varphi ::= p \in \mathsf{AP} \mid \neg\varphi \mid \varphi \wedge \varphi \mid \bigcirc\varphi \mid \varphi \,\mathcal{U}\, \varphi.$$

Here $\bigcirc$ (strong Next) and $\mathcal{U}$ (Until) are temporal operators. We also use common operators, such as $\Diamond$ (eventually) and $\Box$ (always), where $\Diamond\varphi \equiv \text{true}\,\mathcal{U}\,\varphi$ and $\Box\varphi \equiv \neg\Diamond\neg\varphi$.

Let $\pi = \pi_0\pi_1\pi_2\ldots\pi_N \in (2^{\mathsf{AP}})^*$ be a finite trace and φ be an LTL$_f$ formula. We denote by $\pi[i\ldots] = \pi_i\ldots\pi_N$ the fragment that starts at position i. The semantics of φ is defined inductively as follows:

$$\begin{aligned}
\pi \models p &\iff p \in \pi_0,\\
\pi \models \neg\varphi &\iff \pi \not\models \varphi,\\
\pi \models \varphi_1 \wedge \varphi_2 &\iff \pi \models \varphi_1 \text{ and } \pi \models \varphi_2,\\
\pi \models \bigcirc\varphi &\iff N > 0 \text{ and } \pi[1\ldots] \models \varphi,\\
\pi \models \varphi_1 \,\mathcal{U}\, \varphi_2 &\iff \exists 0 \le k \le N.\pi[k\ldots] \models \varphi_2 \text{ and},\\
&\qquad\qquad \forall 0 \le j < k.\pi[j\ldots] \models \varphi_1.
\end{aligned}$$

The language of φ, denoted $[\varphi]$, is the set of all finite traces over 2^{AP} that satisfy φ, i.e., $[\varphi] = \{\,\pi \in (2^{\mathsf{AP}})^* \mid \pi \models \varphi\,\}$.

For an LTL$_f$ formula φ over AP, one can construct a deterministic finite automaton (DFA) $\mathcal{A}_\varphi = (\Sigma, Q, \bar{q}, \delta, F)$ recognizing $[\varphi]$, where $\Sigma = 2^{\mathsf{AP}}$ is the alphabet, Q is a finite set of states, $\bar{q} \in Q$ is the initial state, $\delta\colon Q \times \Sigma \to Q$ is the transition function, and $F \subseteq Q$ is the set of accepting states. A run $\rho = q_0q_1\ldots q_{N+1} \in Q^+$ over a word $w =$

$w_0 w_1 \dots w_N \in \Sigma^*$ is a sequence of states such that $q_0 = \bar{q}$ and $q_{i+1} = \delta(q_i, w_i)$ for each $0 \leq i \leq N$. The word $w = w_0 w_1 \dots w_N \in \Sigma^*$ is accepted by $\mathcal{A}_\varphi$ if $q_{N+1} \in F$.

B. MDPSTs

We now introduce MDPSTs, a framework capable of modeling both quantifiable and unquantifiable uncertainties in robotics [24], [25]. An MDPST is a tuple $\mathcal{M} = (S, \bar{s}, A, \mathcal{F}, \mathcal{T}, L)$ where S is a finite set of states, $\bar{s} \in S$ is the designated initial state, A is a finite set of actions ($A(s) \subseteq A$ denotes applicable actions at s), $\mathcal{F}\colon S \times A \to 2^{2^S}$ is the set-valued transition function, $\mathcal{T}\colon S \times A \times 2^S \to (0, 1]$ is the transition probability function with $\sum_{\Theta \in \mathcal{F}(s,a)} \mathcal{T}(s, a, \Theta) = 1$ for all $s \in S$ and $a \in A(s)$, and $L\colon S \to 2^{\mathsf{AP}}$ is the labeling function. The main difference between MDPSTs and traditional MDPs is its representation of *transition uncertainty*. MDPs map state-action pairs to a probability distribution over all states S, while MDPSTs map a state-action pair to a probability distribution over the power set of S. The adversarial environment decides what successor in a subset will be selected and thus forms a *feasible distribution* described below, where $\mathrm{Distr}(X)$ denotes the set of probability distributions over the set X.

Definition 1 (Feasible distribution): Given an MDPST $\mathcal{M} = (S, \bar{s}, A, \mathcal{F}, \mathcal{T}, L)$, $s \in S$, and $a \in A(s)$, $\mathfrak{h}^a_s \in \mathrm{Distr}(S)$ is a *feasible distribution* of (s, a), denoted by $s \xrightarrow{a} \mathfrak{h}^a_s$, if there exist $\alpha_\Theta \in \mathrm{Distr}(\Theta)$ for each $\Theta \in \mathcal{F}(s, a)$ such that $\mathfrak{h}^a_s(s') = \sum_{\Theta \in \mathcal{F}(s,a)} \mathcal{T}(s, a, \Theta) \cdot \alpha_\Theta(s')$ for each $s' \in S$. We denote by $\mathfrak{H}^a_s$ the set of all feasible distributions $\mathfrak{h}^a_s$ of (s, a).

This definition captures all valid ways the system can evolve under uncertain transition sets. Feasible distributions bridge the gap between set-valued transitions and executable policies. Each $\alpha_\Theta(s')$ represents the (unknown) likelihood of transitioning to s' given Θ was selected.

Conceptually, $\mathfrak{H}^a_s$ defines all possible transition distributions consistent with the MDPST's uncertainty model.

A (finite) *path* $\xi \in (S \times A)^* \times S$ of an MDPST $\mathcal{M}$ is an alternating sequence of states and actions ending with a state, $\xi = s_0 a_0 s_1 \dots s_N$, such that for each $0 \leq i < N$, there is $\Theta_i \in \mathcal{F}(s_i, a_i)$ such that $\mathcal{T}(s_i, a_i, \Theta_i) > 0$ and $s_{i+1} \in \Theta_i$; we let $lst(\xi) = s_N$ denote its last state. Infinite paths are defined similarly. We let FPaths (Paths, resp.) denote the sets of all finite (infinite, resp.) paths of $\mathcal{M}$.

There are two sources of nondeterminism in MDPSTs: the choice of the next action in the current state and the choice of the corresponding feasible distribution. The former is resolved by a *strategy*, while the latter by a *nature*.

Let $\perp \notin A$ be a fresh symbol we use to denote termination. We now define a strategy that we look for in a planning problem.

Definition 2 (Strategy): A function $\sigma\colon S \to \mathrm{Distr}(A \cup \{\perp\})$ is a *strategy* if $\sigma(s) \in \mathrm{Distr}(A(s) \cup \{\perp\})$ for each $s \in S$.

In an MDPST, a nature is an adversarial player that tries to prevent the agent from fulfilling its tasks.

Definition 3 (Nature): A function $\gamma\colon S \times A \to \mathrm{Distr}(S)$ is a *nature* if $\gamma(s, a) \in \mathfrak{H}^a_s$ for each $s \in S$ and $a \in A(s)$.

We denote by $\Sigma_\mathcal{M}$ and $\Gamma_\mathcal{M}$ the set of all strategies and natures for $\mathcal{M}$, respectively.

Given an MDPST $\mathcal{M}$, a strategy $\sigma \in \Sigma_\mathcal{M}$, a nature $\gamma \in \Gamma_\mathcal{M}$, and a state s, σ and γ induce a probability measure over the finite paths of $\mathcal{M}$ from the state s as follows: the basic measurable events are the cylinder sets of finite paths, where the *cylinder set* of a finite path ξ is the set $Cyl(\xi) = \{\, \xi' \in \text{Paths} \mid \xi \text{ is a prefix of } \xi' \,\}$. The probability $\Pr^{\sigma,\gamma}_s$ of the cylinder set $Cyl(\xi)$ is defined inductively as follows: $\Pr^{\sigma,\gamma}_s(Cyl(s)) = 1$; $\Pr^{\sigma,\gamma}_s(Cyl(t)) = 0$ for $t \neq s$; and $\Pr^{\sigma,\gamma}_s(Cyl(\xi)) = \Pr^{\sigma,\gamma}_s(Cyl(\xi')) \cdot \sigma(lst(\xi'))(a) \cdot \gamma(lst(\xi'), a)(t)$ for $\xi = \xi' a t$. The probability of ξ is defined as $\Pr^{\sigma,\gamma}_s(\xi) = \Pr^{\sigma,\gamma}_s(Cyl(\xi)) \cdot \sigma(lst(\xi))(\perp)$. Standard measure-theoretical arguments ensure that $\Pr^{\sigma,\gamma}_s$ extends uniquely to the σ-field generated by cylinder sets (see, e.g., [27]). We write $\Pr^{\sigma,\gamma}$ for $\Pr^{\sigma,\gamma}_{\bar{s}}$ and define $\Pr^\sigma$ as $\Pr^\sigma = \min_{\gamma \in \Gamma_\mathcal{M}} \Pr^{\sigma,\gamma}$.

We say that a strategy σ is *terminating* if $\Pr^\sigma(\text{FPaths}) = 1$.

IV. Consumption MDPSTs

In addition to quantifiable and unquantifiable uncertainties, we now introduce our model called *Consumption MDPSTs* (CMDPSTs) which take into account resource constraints. This model in fact generalizes consumption MDPs [12], since MDPSTs are more general than MDPs.

Definition 4: A CMDPST is a tuple $\mathcal{M} = (S, \bar{s}, A, \mathcal{F}, \mathcal{T}, L, C, R, \mathsf{cap})$ where

- $(S, \bar{s}, A, \mathcal{F}, \mathcal{T}, L)$ is an MDPST,
- $C\colon S \times A \to \mathbb{R}_{\geq 0}$ is a resource consumption function,
- $R \subseteq S$ is the set of reload states where the resource level can be reset to full capacity, and
- $\mathsf{cap} \in \mathbb{R}_{\geq 0}$ is the maximum resource capacity.

The CMDPST framework introduces resource-aware path analysis, where for any finite path $\xi = s_0 a_0 s_1 \dots s_{N+1}$, its cumulative resource consumption is computed as $\sum_{i=0}^{N} C(s_i, a_i)$. To support persistent operation in long-horizon tasks, specific states $R \subseteq S$ act as *reload states* where the accumulated cost resets to zero.

A strategy is considered *feasible* if, under all adversarial choices by nature, the cost of every finite path occurring with non-zero probability never exceeds the capacity cap. This ensures that the system remains operational under worst-case uncertainty without violating its resource constraints. Formally, given a finite path $\xi = s_0 a_0 s_1 \dots s_{N+1}$, let $Z = \{0, N\} \cup \{\, 0 \leq i \leq N \mid s_i \in R \,\}$ be the positions where we reset the cost to zero. The maximum resource cost of ξ is

$$\mathsf{Cost}(\xi) = \max_{z \in Z} \sum_{i=z}^{\min Z \cap \mathbb{N}_{>z}} C(s_i, a_i)$$

and we call ξ *feasible* if $\mathsf{Cost}(\xi) \leq \mathsf{cap}$; we call a strategy σ *feasible* if $\Pr^\sigma(\{\, \xi \in \text{FPaths} \mid \xi \text{ is feasible} \,\}) = 1$. The intuition is that, we can divide the feasible path ξ into several fragments whose ends are reload/reset states in which the total cost of each fragment must not exceed cap.

Let us now motivate CMDPSTs for robot planning under uncertainties and resource constraints. Consider again the

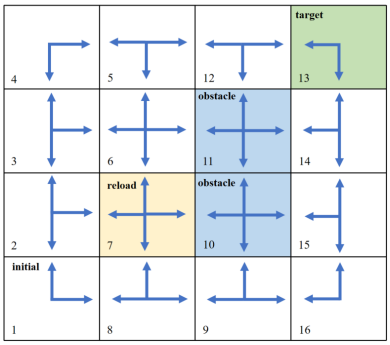

Fig. 2. Simplified Warehouse Planning

warehouse scenario introduced in Section II with 16 distinct warehouses arranged in a 4×4 grid; each warehouse corresponds to a state $s \in \mathcal{S} = \{1, 2, \ldots, 16\}$, shown in Fig. 2. Warehouse 7 is designated as a *reload state*, where the AGV may recharge its battery. Warehouses 10 and 11 are *obstacle states*, which are inaccessible due to construction. The AGV starts at warehouse 1 and must accomplish a delivery task with temporal requirements: it must eventually reach warehouse 13, the designated *delivery target* while always *avoiding* warehouses 10 and 11, which are closed.

At each state, the AGV can choose to move according to one of the directions $A = \{\texttt{UP}, \texttt{DOWN}, \texttt{RIGHT}, \texttt{LEFT}\}$, if possible. At warehouse $s \in S$, each action $a \in A(s)$ incurs a cost $C(s, a) \in \mathbb{R}_{\geq 0}$ representing the energy consumed. The AGV is equipped with a battery of limited capacity cap, and its energy decreases according to $C(s, a)$. When the AGV reaches a reload warehouse, its battery is recharged to cap.

The AGV's motion is modeled as a CMDPST, incorporating both stochastic uncertainty and adversarial nondeterminism (e.g., human operator intervention). The stochastic transition function $\mathcal{T}$ is defined as follows. When the AGV attempts to move to a desired warehouse i under an action:

- with probability 0.8, the action succeeds, and the AGV transitions to warehouse i;
- with probability 0.2, the action underperforms due to actuation noise, and the AGV transitions to warehouse $i-1$, if $i > 1$.

To model environmental (e.g., human intervention) uncertainty, the set-valued transition function $\mathcal{F}$ includes an additional nondeterministic option. For instance, the environment may force the AGV into warehouse $i-2$, regardless of the intended action, provided $i > 2$. As a concrete example, let the AGV be at warehouse 2 with 5 units of remaining energy and consider the $\texttt{RIGHT}$ action having $C(2, \texttt{RIGHT}) = 2$. The intended target is warehouse 7 and the possible successor sets and their probabilities are $\mathcal{F}(2, \texttt{RIGHT}) = \{\{7, 5\}, \{6, 5\}\}$ and $\mathcal{T}(2, \texttt{RIGHT}, \{7, 5\}) = 0.8$ and $\mathcal{T}(2, \texttt{RIGHT}, \{6, 5\}) = 0.2$. If the AGV moves to warehouse 7, the reload rule applies and its energy level is reset to cap; otherwise, the remaining energy decreases to 3.

Our goal is to synthesize a feasible strategy (i.e., the AGV never runs out of energy) and optimally robust (i.e., it maximizes the probability of delivering the goods independently on how humans interfere with the AGV).

Definition 5 (β-robust strategy): Given a CMDPST $\mathcal{M}$, an LTL$_f$ formula φ, and a threshold β, we say that a strategy $\sigma \in \Sigma_{\mathcal{M}}$ is *β-robust for* φ if it is feasible and $\Pr^{\sigma}([\varphi]) \geq \beta$.

A β-robust strategy σ is *optimal* for φ if $\Pr^{\sigma}([\varphi]) \geq \Pr^{\sigma'}([\varphi])$ for all β-robust strategies σ' for φ.

Problem formulation: To synthesize an optimal robust strategy that ensures the satisfaction of a high-level task specified by an LTL$_f$ formula φ while guaranteeing that the resource is never exhausted under adversarial environment behaviors, we formulate the *Resource Constrained Optimal Robust Strategy Synthesis* problem: given a CMDPST model $\mathcal{M}$ and an LTL$_f$ specification φ, the objective is to compute a *feasible* strategy (i.e., the resource never runs out under all adversarial choices by nature) such that, the probability of satisfying the LTL$_f$ specification φ is maximized.

V. Naïvely Constrained Optimal Robust Strategy Synthesis

We first introduce a naïve method for solving the resource constrained optimal robust strategy synthesis problem given a CMDPST $\mathcal{M}$ and an LTL$_f$ formula φ over AP. The basic idea is to convert the CMDPST to an MDPST so we can use a well-developed approach to synthesize an optimal robust strategy from an MDPST. The algorithm is given as follows:

1) construct a DFA $\mathcal{A}_\varphi$ from φ such that $\mathcal{L}(\mathcal{A}_\varphi) = [\varphi]$,
2) build the product CMDPST $\mathcal{M}^\times$ of $\mathcal{M}$ and $\mathcal{A}_\varphi$,
3) unroll $\mathcal{M}^\times$ into a MDPST $\mathcal{M}^\otimes$, and
4) compute the optimal robust strategy on $\mathcal{M}^\otimes$.

Note that it is possible to combine steps 2) and 3) together without constructing the intermediate product CMDPST $\mathcal{M}^\times$ explicitly. However, in order to make our presentation simpler and to further include optimizations, we will introduce the construction step by step.

DFA construction: The first LTL$_f$-to-DFA conversion was proposed in [15] and the current state of the art conversions normally employ a compositional methodology [28]–[30]. In our method, we use the off-the-shelf tool from [29]. Assume that the constructed DFA is $\mathcal{A}_\varphi = (\Sigma = 2^{\mathsf{AP}}, Q, \bar{q}, \delta, F)$ with $\mathcal{L}(\mathcal{A}_\varphi) = [\varphi]$.

Product $\mathcal{M}^\times$ construction: Given a CMDPST $\mathcal{M} = (S, \bar{s}, A, \mathcal{F}, \mathcal{T}, L, C, R, \mathsf{cap})$ and the DFA $\mathcal{A}_\varphi$, the *product CMDPST* $\mathcal{M}^\times$ of $\mathcal{M}$ and $\mathcal{A}_\varphi$ is defined as the CMDPST $\mathcal{M}^\times = (S^\times, \bar{s}^\times, A^\times, \mathcal{F}^\times, \mathcal{T}^\times, L^\times, C^\times, R^\times, \mathsf{cap}^\times, F^\times)$ where

- $S^\times = S \times Q$ and $\bar{s}^\times = (\bar{s}, \bar{q})$;
- $A^\times = A$;
- $\mathcal{F}^\times$ and $\mathcal{T}^\times$ are defined as follows: for each $(s, q) \in S^\times$ and $a \in A^\times$, let $q' = \delta(q, L(s))$. Then $\mathcal{F}^\times((s, q), a) = \{\, \Theta \times \{q'\} \mid \Theta \in \mathcal{F}(s, a) \,\}$ and, for each $\Theta \in \mathcal{F}(s, a)$, $\mathcal{T}^\times((s, q), a, \Theta \times \{q'\}) = \mathcal{T}(s, a, \Theta)$;
- $L^\times((s, q)) = L(s)$ for each $(s, q) \in S^\times$;
- $C^\times((s, q), a) = C(s, a)$;
- $R^\times = \{\, (s, q) \in S^\times \mid s \in R \,\}$;

- $\mathsf{cap}^\times = \mathsf{cap}$; and
- $F^\times = \{\, (s,q) \in S^\times \mid q \in F \,\}$.

By building the product $\mathcal{M}^\times$, it is easy to identify which paths in $\mathcal{M}$ generate traces in $[\varphi]$, since every path in $\mathcal{M}^\times$ that can reach a final state in $F^\times$ is one of such paths. This is, however, not enough to derive a feasible strategy as we still need to consider the resource constraints in the $\mathcal{M}^\times$. To this end, we can convert the CMDPST $\mathcal{M}^\times$ to a MDPST $\mathcal{M}^\otimes$, where a well-developed optimal robust strategy approach (e.g., [24]) can be exploited. The conversion approach is quite simple yet effective: we record in each state also its current resource level. We call this conversion approach the *unroll* approach.

Unrolled $\mathcal{M}^\otimes$ construction: For the input CMDPST $\mathcal{M}^\times = (S^\times, \bar{s}^\times, A^\times, \mathcal{F}^\times, \mathcal{T}^\times, L^\times, C^\times, R^\times, \mathsf{cap}^\times, F^\times)$, we now define the *unrolled product MDPST* $\mathcal{M}^\otimes$ as the MDPST $\mathcal{M}^\otimes = (S^\otimes, \bar{s}^\otimes, A^\otimes, \mathcal{F}^\otimes, \mathcal{T}^\otimes, L^\otimes, F^\otimes)$ where

- $S^\otimes \subseteq S^\times \times [0, \mathsf{cap}]$ is the smallest set such that $\bar{s}^\otimes = (\bar{s}^\times, \mathsf{cap}) \in S^\otimes$ and that is closed under the functions $\mathcal{F}^\otimes$ and $\mathcal{T}^\otimes$ defined below,
- $A^\otimes((s,c)) = A^\times(s)$ for all $(s,c) \in S^\otimes$,
- $\mathcal{F}^\otimes$ and $\mathcal{T}^\otimes$ are constructed as follows: for all $(s,c) \in S^\otimes$ and $a \in A^\times(s)$ such that $C^\times(s,a) \le c$ and for each $\Theta \in \mathcal{F}^\times(s,a)$, let $\Theta^\otimes = \{\, (s', \mathsf{cap}) \mid s' \in \Theta \cap R^\times \,\} \cup \{\, (s', c - C^\times(s,a)) \mid s' \in \Theta \setminus R^\times \,\}$; then $\Theta^\otimes \in \mathcal{F}^\otimes((s,c),a)$ and $\mathcal{T}^\otimes((s,c),a,\Theta^\otimes) = \mathcal{T}^\times(s,a,\Theta)$,
- $L^\otimes((s,c)) = L^\times(s)$ for all $(s,c) \in S^\otimes$, and
- $F^\otimes = \{\, (s,c) \in S^\otimes \mid s \in F^\times \,\}$.

This construction preserves the original transition probabilities while explicitly encoding current resource level into the state space. The resource level c in each state (s,c) of $S^\otimes$ is positive, and this will guarantee that every path in $\mathcal{M}^\otimes$ is feasible.

Proposition 1: Given a product CMDPST $\mathcal{M}^\times$, let $\mathcal{M}^\otimes$ be its unrolled MDPST. Then, the following properties hold:

1) for every feasible strategy $\sigma^\times \in \Sigma_{\mathcal{M}^\times}$ there is a strategy $\sigma^\otimes \in \Sigma_{\mathcal{M}^\otimes}$ such that for every feasible path $\xi^\times \in \mathrm{FPaths}_{\mathcal{M}^\times}$ there is a path $\xi^\otimes \in \mathrm{FPaths}_{\mathcal{M}^\otimes}$ such that $\mathrm{proj}(\xi^\otimes) = \xi^\times$ and $\Pr^{\sigma^\otimes}_{\mathcal{M}^\otimes}(\xi^\otimes) = \Pr^{\sigma^\times}_{\mathcal{M}^\times}(\xi^\times)$;
2) for every strategy $\sigma^\otimes \in \Sigma_{\mathcal{M}^\otimes}$, there is a feasible strategy $\sigma^\times \in \Sigma_{\mathcal{M}^\times}$ such that for every path $\xi^\otimes \in \mathrm{FPaths}_{\mathcal{M}^\otimes}$ we have that $\Pr^{\sigma^\times}_{\mathcal{M}^\times}(\mathrm{proj}(\xi^\otimes)) = \Pr^{\sigma^\otimes}_{\mathcal{M}^\otimes}(\xi^\otimes)$,

where $\mathrm{proj}((s_0,c_0)a_0 \dots (s_n,c_n)) = s_0 a_0 \dots s_n$.

Proof: Both properties are proved by equating $\sigma^\times(s^\times)$ and $\sigma^\otimes((s^\times,c))$; for the first property, the path $\xi^\otimes$ is obtained from $\xi^\times = s_0^\times a_0 \dots s_n^\times$ as $\xi^\otimes = (s_0^\times, c_0)a_0 \dots (s_n^\times, c_n)$ where $c_0 = \mathsf{cap}$ and for each $0 \le i < n$, $c_{i+1} = \mathsf{cap}$ if $s_{i+1}^\times \in R^\times$ and $c_{i+1} = c_i - C(s_i,a_i)$ otherwise. A direct application of the definitions yields the stated properties. ∎

We now extend the warehouse transportation example from Fig. 1 to incorporate *resource consumption* and the *unrolling construction*. Each warehouse $s \in \mathcal{S} = \{1, \dots, 16\}$ is now augmented with a resource level $c \in [0, \mathsf{cap}]$, resulting in unrolled states of the form (s,c). The AGV starts at $(1, \mathsf{cap})$, i.e., at warehouse 1 with the battery fully charged.

At each step, the AGV chooses action a among the actions $A = \{\mathtt{UP}, \mathtt{DOWN}, \mathtt{LEFT}, \mathtt{RIGHT}\}$. Executing a at state (s,c) consumes $C(s,a)$ units of resource, leading to transitions $(s,c) \xrightarrow{a} (s', c - C(s,a))$ for each $s' \in \Theta \in \mathcal{F}(s,a)$ when $C(s,a) \le c$; when $C(s,a) > c$, action a is disabled.

Reload warehouses allow resource replenishment: since warehouse 7 is a *reload state*, any state $(7,c)$ reached is reset to $(7, \mathsf{cap})$. Obstacle warehouses (i.e., warehouses 10 and 11) remain prohibited in the unrolled model, and the labeling function $L^\otimes$ is inherited from the base CMDPST.

As a concrete example, suppose the AGV is at unrolled state $(2,5)$ and executes $\mathtt{RIGHT}$ with $C(2, \mathtt{RIGHT}) = 2$. In the original model, the intended target is warehouse 7. In the unrolled model, the possible successor sets are $\{(7, \mathsf{cap}), (5,3)\}$ with probability 0.8 and $\{(6,3), (5,3)\}$ with probability 0.2. Here, the remaining resource decreases from 5 to 3 by the cost $C(2, \mathtt{RIGHT}) = 2$, except when reaching warehouse 7, where it is reset to $(7, \mathsf{cap})$ due to the reload mechanism.

Optimal robust strategy extraction: Now, we will introduce the standard approach for MDPSTs to extract an optimal robust strategy. Let $\mathcal{M}^\otimes = (S^\otimes, \bar{s}^\otimes, A^\otimes, \mathcal{F}^\otimes, \mathcal{T}^\otimes, L^\otimes, F^\otimes)$ be an unrolled MDPST. To make our presentation more general, we also assume that we are given a winning region $F^\otimes \subseteq W \subseteq S^\otimes$ in which the agent has a strategy to reach almost surely target states $F^\otimes$ against any nature. Of course, we can always let $W = F^\otimes$ and the following procedure will still be correct.

To obtain the optimal robust strategy, we first define a *robust value function* $V \colon S^\otimes \to [0,1]$ defined as:

$$V(s) = \max_\sigma \min_\gamma \Pr_s^{\sigma,\gamma}(\Diamond W)$$

where σ is the agent's strategy and γ is the nature's strategy (resolving uncertainty).

In order to compute the robust value function, we resort to solving a *robust Bellman equation system*. More precisely, for all $s \in W$, we set $V(s) = 1$. For $s \in S^\otimes \setminus W$, $V(s)$ needs to satisfy the following constraint:

$$V(s) = \max_{a \in A^\otimes(s)} \sum_{\Theta \in \mathcal{F}^\otimes(s,a)} \mathcal{T}^\otimes(s,a,\Theta) \cdot \min_{s' \in \Theta} V(s')$$

This definition captures the worst-case reachability probability under both adversarial nondeterminism and probabilistic uncertainty.

After computing the robust value function V, we are in fact able to record the action that leads to the maximal value in V. Indeed, it is easy to obtain an optimal robust strategy $\sigma^\otimes \colon S^\otimes \to A^\otimes \cup \{\bot\}$ on $\mathcal{M}^\otimes$ where $\sigma^\otimes(s)$ is the action in $A^\otimes$ that leads to the maximal robust value $V(s)$ if $V(s) > 0$, or just $\sigma^\otimes(s) = \bot$ when $V(s) = 0$. Note that, for reachability goals, there are history-independent optimal strategies on MDPSTs [24].

In order to obtain an optimal robust strategy on the original CMDPST $\mathcal{M}$, we will use the resource level and the DFA

state encoded in a state $s \in S^{\otimes}$ as part of the memory. Thus, we will obtain a history-dependent optimal robust strategy σ on $\mathcal{M}$. The strategy $\sigma\colon S \times M \to A$ where $M = Q \times [0, \mathsf{cap}]$ and the execution of σ is defined as follows:

- The initial memory state of the strategy is $\bar{m} = \langle \bar{q}, \mathsf{cap} \rangle$.
- For a state $s \in S$ and a current memory state $m = \langle q, c \rangle$, we have $\sigma(s, m) = \sigma^{\otimes}(s, q, c)$ and then the memory state m' is updated to $\langle \delta(q, L(s)), c - C(s, a) \rangle$.
- When $m = \langle q, c \rangle$ involves a final state $q \in F$ in the DFA $\mathcal{A}_{\varphi}$, we let $\sigma(s, m) = \bot$. This means that the agent has fulfilled the task and decides to terminate.

Theorem 1: Our synthesized strategy σ is an optimal robust strategy for $\mathcal{M}$ to fulfill the LTL$_f$ specification φ.

Proof: The proof is based on a direct application of the definitions and Proposition 1 to equate the probability values between feasible paths of $\mathcal{M}^{\times}$ and paths of $\mathcal{M}^{\otimes}$. ■

VI. Optimized Robust Strategy Synthesis

In the previous section, we introduced the unrolled approach to convert a CMDPST $\mathcal{M}^{\times}$ to an MDPST $\mathcal{M}^{\otimes}$ which records the current resource level in each state. This naïve approach builds an enormous state space for the MDPST and thus gives an optimal robust strategy of large size. To this end, we propose to first prune the CMDPST $\mathcal{M}^{\times}$ before converting to the MDPST. Recall that the size of our synthesized strategy is proportional to that of the MDPST. Therefore, our optimization is to remove all states from which the agent does not have a strategy maintaining feasibility.

We observe that the approach we present in this section can also be used in the design phase of the system to check whether there exists a feasible strategy given a CMDPST $\mathcal{M}$ and a set T of target states. That is, when deploying a system to complete a task, it is often necessary to check first whether the current system has a feasible strategy to possibly reach the target states instead of computing an optimal robust strategy directly. Any deployment error found in this phase can be fixed immediately. For instance, once we find out that there are no feasible strategies, we can look for the root causes, such as that there are not enough charging stations or their spatial distribution is not reasonable, so we can add more or adjust the locations of the charging stations in order to make it possible to fulfill the given (set of) task(s).

Another advantage of our proposed optimization is that it only involves simple graph exploration and value accounting. Thus, it is easy to implement. The central idea in our optimization is to compute the feasible region where the agent has a strategy that keeps it moving against any nature. Then, we can just check whether some target state is in the feasible region. For pruning, we only need to remove all states and transitions that are outside the feasible region.

1) Feasible region computation: The FeasibleRegion algorithm is shown as Algorithm 1 and, on input a CMDPST $\mathcal{M}$, it returns the set of states S_f that are reachable from $\bar{s}$ by a feasible path, as well as the map $\vec{a}$ that assigns to each state s the set of actions that preserve feasibility. To compute S_f and $\vec{a}$, FeasibleRegion makes use of two variables s and c, keeping the state currently under evaluation and its cost, and two additional data structures: a set E to remember the reached states that need to be evaluated and a vector $\vec{c}$ that stores the minimum cost to reach each state from either the initial state or a reload state already in S_f or E. The computation proceeds as follows: starting from the initial state $s = \bar{s}$ and cost $c = 0$ (Line 4), we add it to S_f (Line 6) and consider each action $a \in A(s)$ that preserves feasibility, i.e., $c + C(s, a) \leq \mathsf{cap}$ (Line 7). Since feasibility is preserved, we add it to $\vec{a}(s)$ (Line 8) and for each possible a-successor $s' \in \cup_{\Theta \in \mathcal{F}(s,a)} \Theta$ (Line 9), we check whether we have a lower cost to reach it (Line 10); if this is the case, then we update its cost (Line 11) to $\vec{c}(s') = c + C(s, a)$ or to 0 if it is a reload state (Lines 12) and add it to E for further analysis (Line 13), as some action in $A(s')$ might have now become feasible. Once we have dealt with all actions and successor states, we take the next state to be evaluated from E (Line 14) by calling PickOne(E). This auxiliary procedure removes one arbitrary element from E and returns it together with the updated E; if no such element exists, then $(\emptyset, \lrcorner)$ is returned, where $\lrcorner \notin S$ is a fresh symbol. We then update the current cost c of s (Line 15) and repeat these steps until we have no other state to evaluate (Line 16) so we can return S_f and $\vec{a}$ (Line 17).

Algorithm 1 FeasibleRegion

Require: CMDPST $\mathcal{M} = (S, \bar{s}, A, \mathcal{F}, \mathcal{T}, L, C, R, \mathsf{cap})$
1: initialize $\vec{a} \in A^S$ to be $\emptyset$ for each state s
2: initialize $\vec{c} \in \mathbb{R}_{\geq 0}^{S \cup \{\lrcorner\}}$ to be ∞ for each state s
3: **if** $\bar{s} \in R$ **then** $\vec{c}(\bar{s}) \leftarrow 0$
4: $S_f \leftarrow \emptyset$; $E \leftarrow \emptyset$; $s \leftarrow \bar{s}$; $c \leftarrow 0$
5: **repeat**
6: $\quad S_f \leftarrow S_f \cup \{s\}$
7: $\quad$ **for all** $a \in A(s)$ such that $c + C(s, a) \leq \mathsf{cap}$ **do**
8: $\quad\quad \vec{a}(s) \leftarrow \vec{a}(s) \cup \{a\}$
9: $\quad\quad$ **for all** $s' \in \cup_{\Theta \in \mathcal{F}(s,a)} \Theta$ **do**
10: $\quad\quad\quad$ **if** $\vec{c}(s') > c + C(s, a)$ **then**
11: $\quad\quad\quad\quad \vec{c}(s') = c + C(s, a)$
12: $\quad\quad\quad\quad$ **if** $s' \in R$ **then** $\vec{c}(s') \leftarrow 0$
13: $\quad\quad\quad\quad E \leftarrow E \cup \{s'\}$
14: $\quad E, s \leftarrow$ PickOne(E)
15: $\quad c \leftarrow \vec{c}(s)$
16: **until** $s = \lrcorner$
17: **return** $S_f, \vec{a}$

The fact that FeasibleRegion terminates follows from the fact that S is finite and that every time a state s is added to E (Line 13) its cost is strictly decreased (cf. Lines 10-12) and the cost cannot be negative (cf. Definition 4), so s can be added to E only a finite number of times.

It is easy to observe the following properties about S_f and $\vec{a}$ returned by FeasibleRegion:

Lemma 1: Given a CMDPST $\mathcal{M}$, let $(S_f, \vec{a}) =$ FeasibleRegion$(\mathcal{M})$. Then the following properties hold:

1) $S_f \subseteq S$ and for each $s \in S_f$, $\vec{a}(s) \subseteq A(s)$.
2) For each $s \in S_f$, $a \in \vec{a}(s)$, and $\Theta \in \mathcal{F}(s, a)$, $\Theta \subseteq S_f$.

3) For each feasible path $\xi \in \text{FPaths}$, $lst(\xi) \in S_f$.
4) For each $s \in S_f$, there exists a feasible path $\xi \in \text{FPaths}$ such that $lst(\xi) = s$.
5) For each $s \in S \setminus S_f$, there is no path $\xi \in \text{FPaths}$ such that $lst(\xi) = s$ that is feasible.
6) For each $s \in S_f$ and each feasible path $\xi \in \text{FPaths}$ such that $lst(\xi) = s$, for each $a \in \vec{a}(s)$ and each $s' \in \cup_{\Theta \in \mathcal{F}(s,a)} \Theta$, the path $\xi a s'$ is feasible.
7) For each $s \in S_f$ and each feasible path $\xi \in \text{FPaths}$ such that $lst(\xi) = s$, for each $a \in A(s) \setminus \vec{a}(s)$ and each $s' \in \cup_{\Theta \in \mathcal{F}(s,a)} \Theta$, the path $\xi a s'$ is not feasible.

Proof: The proof is based on a simple inspection of FEASIBLEREGION's code and on specific invariants for the outer loop (Lines 5-16) like "for all $s \in S_f \cup E$ there is a feasible path $\xi \in \text{FPaths}$ such that $lst(\xi) = s$" or "for all $s \in S_f \cup E$, $\vec{c}(s)$ equals the cost of a feasible path ending in s that either starts in $\bar{s}$ or in a reload state in S_f and no reload state occurs in its interior sub-path". ■

2) Target feasibility checking: Given a CMDPST $\mathcal{M}$ and a set of target states T, we can easily verify whether T can be reached by some feasible path by first computing $(S_f, \vec{a}) = \text{FEASIBLEREGION}(\mathcal{M})$ and then checking whether $T \cap S_f \neq \emptyset$. Similarly, we can verify whether all states in T can be reached by some feasible path by checking $T \subseteq S_f$. Note however that even if $T \subseteq S_f$ holds, there is no guarantee that $\Pr^{\sigma}(\Diamond T) = 1$ for some feasible strategy σ: there can still be some nature preventing T to be reached with probability 1.

3) Feasibility pruning: Pruning a CMDPST to its feasible region is rather easy: it is enough to remove all states not in S_f and all transitions from the states s in S_f whose action is not in $\vec{a}(s)$; by Lemma 1, all paths in the pruned CMDPST are feasible. Formally, given a CMDPST $\mathcal{M} = (S, \bar{s}, A, \mathcal{F}, \mathcal{T}, L, C, R, \mathsf{cap})$ and $(S_f, \vec{a}) = \text{FEASIBLEREGION}(\mathcal{M})$, the pruned CMDPST $\mathcal{M}^{\triangleright} = (S^{\triangleright}, \bar{s}^{\triangleright}, A^{\triangleright}, \mathcal{F}^{\triangleright}, \mathcal{T}^{\triangleright}, L^{\triangleright}, C^{\triangleright}, R^{\triangleright}, \mathsf{cap}^{\triangleright})$ has

- $S^{\triangleright} = S_f$ and $\bar{s}^{\triangleright} = \bar{s}$;
- $A^{\triangleright} = \cup_{s \in S_f} \vec{a}(s)$;
- $\mathcal{F}^{\triangleright}$ is defined as $\mathcal{F}^{\triangleright}(s,a) = \mathcal{F}(s,a)$ only for each $s \in S^{\triangleright}$ and $a \in \vec{a}(s)$;
- $\mathcal{T}^{\triangleright}$ is defined as $\mathcal{T}^{\triangleright}(s,a,\Theta) = \mathcal{T}(s,a,\Theta)$ only for each $s \in S^{\triangleright}$, $a \in \vec{a}(s)$, and $\Theta \in \mathcal{F}^{\triangleright}(s,a)$;
- $L^{\triangleright}$ is defined as $L^{\triangleright}(s) = L(s)$ only for each $s \in S^{\triangleright}$;
- $C^{\triangleright}$ is defined as $C^{\triangleright}(s,a) = C(s,a)$ only for each $s \in S^{\triangleright}$ and $a \in A^{\triangleright}(s)$;
- $R^{\triangleright} = R \cap S_f$; and
- $\mathsf{cap}^{\triangleright} = \mathsf{cap}$.

Similarly to CMDPST unrolling, we can prove the following correctness result about feasibility pruning:

Proposition 2: Given a CMDPST $\mathcal{M}$, let $\mathcal{M}^{\triangleright}$ be its feasibility pruned CMDPST. Then, we have that

1) for every feasible strategy $\sigma \in \Sigma_{\mathcal{M}}$ there exists a strategy $\sigma^{\triangleright} \in \Sigma_{\mathcal{M}^{\triangleright}}$ such that for every feasible path $\xi \in \text{FPaths}_{\mathcal{M}}$ we have $\Pr^{\sigma^{\triangleright}}_{\mathcal{M}^{\triangleright}}(\xi) = \Pr^{\sigma}_{\mathcal{M}}(\xi)$;
2) for every strategy $\sigma^{\triangleright} \in \Sigma_{\mathcal{M}^{\triangleright}}$, there exists a feasible strategy $\sigma \in \Sigma_{\mathcal{M}}$ such that for every path $\xi^{\triangleright} \in \text{FPaths}_{\mathcal{M}^{\triangleright}}$ we have $\Pr^{\sigma}_{\mathcal{M}}(\xi^{\triangleright}) = \Pr^{\sigma^{\triangleright}}_{\mathcal{M}^{\triangleright}}(\xi^{\triangleright})$.

Proof: Both properties can be proved by using the same strategy on both CMDPSTs and then by a direct application of the definitions and Lemma 1. ■

Proposition 2 ensures that pruning does not alter the maximum value we can obtain in solving the optimal robust strategy synthesis problem, similarly to Proposition 1 and its use in the proof of Theorem 1.

VII. EXPERIMENTS

In this section, we evaluate our approach on an extended AGV transportation scenario. Compared with the motivating example in Section II, the experimental setting is enriched in two aspects. First, instead of a single AGV, we consider multiple AGVs navigating the warehouse network simultaneously, which increases the interaction between agents and the complexity of the strategy space. AGVs operate in the same environment and interactions are handled by disabling actions that would lead to collisions, i.e., multiple agents cannot occupy the same warehouse simultaneously. Second, the action model is augmented: in addition to the movements considered before, AGVs are now allowed to traverse alternative connections between warehouses (e.g., shortcuts or cross-aisle passages), which leads to a larger set of possible transitions. These extensions make the experiments more representative of realistic logistics scenarios and provide a more demanding benchmark for testing both the performance and accuracy of the proposed methods. To systematically assess performance, we vary the warehouse network size from 4 to 16 and report the resulting state space, runtime, and synthesis outcomes.

All simulations are carried out on a laptop with an Intel Core i5-13500HX fourteen-core processor, 16GB DDR5 RAM, and the implementation code can be found at: https://github.com/yihaoyin/CMDPST.git .

In the baseline 4-warehouse scenario, the original MDPST consists of 26 states and 110 transitions. After applying the unrolling construction given in Section V, the model expands to 286 states and 1846 transitions. When further combined with the DFA encoding the LTL$_f$ formula $\neg W_o \mathcal{U} W_d$, which requires to eventually reach the target delivery warehouse W_d while avoiding any obstacle warehouse W_o, the resulting product model contains 858 states and 4014 transitions. On this product, the optimal robust strategy extraction yields a reachability probability of 0.883 with a computation time of 0.52 seconds. We then apply the state-space pruning technique proposed in Section VI, which eliminates states and transitions that are not feasible. This reduction shrinks the product to 275 states and 1743 transitions, leading to a much shorter time of 0.19 seconds while obtaining the same reachability probability value (0.883) as expected.

To test the scalability of our optimal robust strategy synthesis approach, we evaluate networks having from 4 to 16 warehouses, with the results reported in Table I. The different columns report the running time in seconds of the synthesis pipelines, the number of states and transitions in the unrolled MDPST, and the computed probability value. Moreover, "naïve" refers to the baseline pipeline consisting of unrolling

TABLE I
CMDPST OPTIMAL STRATEGY SYNTHESIS

Ware-houses	Time (s) naïve	pruned	# States naïve	pruned	# Transitions naïve	pruned	Prob-ability
4	0.52	0.22	858	275	4014	1743	0.883
5	1.17	0.66	1386	489	8472	3982	0.804
6	2.42	1.61	2046	738	15093	7166	0.833
7	5.89	3.52	2838	1105	24651	12898	0.842
8	9.76	6.94	3762	1494	37530	20111	0.882
9	17.41	10.22	4818	1935	55488	29756	0.875
10	28.59	17.39	6006	2436	76911	41707	0.811
11	44.78	25.33	7326	2993	103512	56782	0.875
12	63.37	41.93	8778	3606	135201	74657	0.811
13	92.55	63.59	10362	4275	173256	96440	0.875
14	139.76	89.31	12078	5000	217203	121415	0.811
15	183.25	127.57	13926	5781	268752	151130	0.875
16	246.73	193.29	15906	6618	326949	184381	0.811

and Bellman updates (Section V), while "pruned" corresponds to the pruning-based pipeline composed by pruning (Section VI), unrolling, and Bellman updates.

The results in Table I clearly demonstrate that the pruning-based method presented in Section VI consistently reduces the state space and achieves notable runtime improvements over the naïve approach, without compromising the correctness of the synthesized strategy.

VIII. CONCLUSION

In this paper, we considered the robust planning for robotic systems with mixed uncertainty and constrained resources. To this end, we formulated a novel resource-constrained optimal robust strategy synthesis problem over CMDPSTs and LTL_f specifications. CMDPSTs support modelling adversarial transition uncertainty, state-dependent resource consumption, reload states, and bounded capacity, enabling a formal treatment of the considered systems in a unified framework. Then we proposed a synthesis framework based on an unrolled product construction between the CMDPST and a DFA encoding the LTL_f formula. To address the challenge of adversarial transitions and unknown successor sets in MDPSTs, we further developed a feasible region-based pruning to reduce the size of the analyzed system while preserving its feasibility properties. Experiments confirmed the practicality of our approach.

Aknowledgement: This work was supported in part by the CAS Project for Young Scientists in Basic Research (Grant No. YSBR-040), and the National Key R&D Program of China (Grant No. 2025YFE0220300).

REFERENCES

[1] J. Levinson, J. Askeland, J. Becker, J. Dolson, D. Held, S. Kammel, J. Z. Kolter, D. Langer, O. Pink *et al.*, "Towards fully autonomous driving: Systems and algorithms," in *IV*, 2011, pp. 163–168.

[2] W. Xu, J. Pan, J. Wei, and J. M. Dolan, "Motion planning under uncertainty for on-road autonomous driving," in *ICRA*, 2014, pp. 2507–2512.

[3] P. Arm, G. Waibel, J. Preisig, T. Tuna, R. Zhou, V. Bickel, G. Ligeza, T. Miki, F. Kehl, H. Kolvenbach *et al.*, "Scientific exploration of challenging planetary analog environments with a team of legged robots," *Sci. Robotics*, vol. 8, no. 80, 2023.

[4] M. S. Bahraini, A. Zenati, and N. Aouf, "Autonomous cooperative visual navigation for planetary exploration robots," in *ICRA*, 2021, pp. 9653–9658.

[5] P. E. Dupont, B. J. Nelson, M. Goldfarb, B. Hannaford, A. Menciassi, M. K. O'Malley, N. Simaan, P. Valdastri, and G.-Z. Yang, "A decade retrospective of medical robotics research from 2010 to 2020," *Sci. Robotics*, vol. 6, no. 60, 2021.

[6] A. Abate, M. Prandini, J. Lygeros, and S. Sastry, "Probabilistic reachability and safety for controlled discrete time stochastic hybrid systems," *Automatica*, vol. 44, no. 11, pp. 2724–2734, 2008.

[7] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 2014.

[8] Mausam and A. Kolobov, *Planning with Markov Decision Processes: An AI Perspective*, ser. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2012.

[9] M. Cognetti, P. Salaris, and P. R. Giordano, "Optimal active sensing with process and measurement noise," in *ICRA*, 2018, pp. 2118–2125.

[10] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. MIT Press, 2005.

[11] N. E. Du Toit and J. W. Burdick, "Robot motion planning in dynamic, uncertain environments," *IEEE Trans. Robotics*, vol. 28, no. 1, pp. 101–115, 2011.

[12] F. Blahoudek, T. Brazdil, P. Novotny, M. Ornik, P. Thangeda, and U. Topcu, "Qualitative controller synthesis for consumption Markov decision processes," in *CAV*, 2020, pp. 421–447.

[13] F. Blahoudek, P. Novotnỳ, M. Ornik, P. Thangeda, and U. Topcu, "Efficient strategy synthesis for mdps with resource constraints," *IEEE Trans. Autom. Contr.*, vol. 68, no. 8, pp. 4586–4601, 2022.

[14] A. Pnueli, "The temporal logic of programs," in *FOCS*, 1977, pp. 46–57.

[15] G. De Giacomo and M. Y. Vardi, "Linear temporal logic and linear dynamic logic on finite traces," in *IJCAI*, 2013, pp. 854–860.

[16] X. C. Ding, M. Lazar, and C. Belta, "LTL receding horizon control for finite deterministic systems," *Automatica*, vol. 50, no. 2, pp. 399–408, 2014.

[17] P. Schillinger, M. Bürger, and D. V. Dimarogonas, "Hierarchical LTL-task MDPs for multi-agent coordination through auctioning and learning," *IJRR*, vol. 38, no. 2–3, pp. 213–242, 2019.

[18] M. Guo and M. M. Zavlanos, "Probabilistic motion planning under temporal tasks and soft constraints," *IEEE Trans. Autom. Control.*, vol. 63, no. 12, pp. 4051–4066, 2018.

[19] M. Cai, S. Xiao, B. Li, Z. Li, and Z. Kan, "Reinforcement learning based temporal logic control with maximum probabilistic satisfaction," in *ICRA*, 2021, pp. 806–812.

[20] T. Wongpiromsarn, M. Ghasemi, M. Cubuktepe, G. Bakirtzis, S. Carr, M. O. Karabag, C. Neary, P. Gohari, and U. Topcu, "Formal methods for autonomous systems," *arXiv preprint arXiv:2311.01258*, 2023.

[21] K. Muvvala, A. M. Wells, M. Lahijanian, L. E. Kavraki, and M. Y. Vardi, "Stochastic games for interactive manipulation domains," in *ICRA*, 2024, pp. 2513–2519.

[22] F. W. Trevizan, F. G. Cozman, and L. N. de Barros, "Planning under risk and knightian uncertainty," in *IJCAI*, 2007, pp. 2023–2028.

[23] ——, "Mixed probabilistic and nondeterministic factored planning through Markov decision processes with set-valued transitions," in *Workshop on A Reality Check for Planning and Scheduling Under Uncertainty at ICAPS*, 2008, p. 62.

[24] P. Yu, S. Zhu, G. De Giacomo, M. Kwiatkowska, and M. Y. Vardi, "The trembling-hand problem for LTLf planning," in *IJCAI*, 2024, pp. 3631–3641.

[25] P. Yu, Y. Li, D. Parker, and M. Kwiatkowska, "Planning with linear temporal logic specifications: Handling quantifiable and unquantifiable uncertainty," *arXiv preprint arXiv:2502.19603*, 2025.

[26] J. A. Baier and S. McIlraith, "Planning with temporally extended goals using heuristic search," in *ICAPS*, 2006, pp. 342–345.

[27] P. Billingsley, *Probability and Measure*. Wiley, 1995.

[28] S. Bansal, Y. Li, L. M. Tabajara, and M. Y. Vardi, "Hybrid compositional reasoning for reactive synthesis from finite-horizon specifications," in *AAAI*, 2020, pp. 9766–9774.

[29] G. De Giacomo and M. Favorito, "Compositional approach to translate LTLf/LDLf into deterministic finite automata," in *ICAPS*, 2021, pp. 122–130.

[30] S. Bansal, Y. Kankariya, and Y. Li, "DAG-based compositional approaches for LTLf to DFA conversions," in *FMCAD*, 2024, pp. 227–235.