Hierarchical Motion Planning under Probabilistic Temporal Tasks and Safe-Return Constraints

Meng Guo[1], Tianjun Liao[2], Junjie Wang[1] and Zhongkui Li[1]

***Abstract*—Safety is crucial for robotic missions within an uncertain environment. Common safety requirements such as collision avoidance are only state-dependent, which can be restrictive for complex missions. In this work, we address a more general formulation as safe-return constraints, which require the existence of a return-policy to drive the system back to a set of safe states with high probability. The robot motion is modeled as a Markov Decision Process (MDP) with probabilistic labels, which can be highly non-ergodic. The robotic task is specified as Linear Temporal Logic (LTL) formulas over these labels, such as surveillance and transportation. We first provide theoretical guarantees on the re-formulation of such safe-return constraints, and a baseline solution based on computing two complete product automata. Furthermore, to tackle the computational complexity, we propose a hierarchical planning algorithm that combines the feature-based symbolic and temporal abstraction with constrained optimization. It synthesizes simultaneously two dependent motion policies: the outbound policy minimizes the overall cost of satisfying the task with a high probability, while the return policy ensures the safe-return constraints. The problem formulation is versatile regarding the robot model, task specifications and safety constraints. The proposed hierarchical algorithm is more efficient and can solve much larger problems than the baseline solution, with only a slight loss of optimality. Numerical validations include simulations and hardware experiments of a search-and-rescue mission and a planetary exploration mission over various system sizes.**

***Index Terms*—Task and Motion Planning, Linear Temporal Logic, Formal Methods, Safety, MDPs**

I. Introduction

Autonomous mobile robots are often deployed in uncertain and unsafe environments that are otherwise risky for humans. For instance, an autonomous ground vehicle (AGV) is deployed in an office for a search-and-rescue mission after disaster: to search for injured victims and bring them to medical stations, to shut down certain machines in different rooms, and to report fire hazards or gas leakage; also an autonomous rover is deployed in a rough terrain for a planetary exploration mission: to gather specimens from areas of interest, to assemble them into containers and store in the storage area; and to charge often. Different from simple navigation tasks, such tasks require not only planning on the task-level regarding which areas to visit and actions to perform there, but also planning on the motion-level regarding how to reach a desired area. In some cases, these tasks are long-term meaning that they should be repeated infinitely often. Furthermore, even though the approximate structure of the environment is known, its exact features within the structure can only be estimated. This has two direct consequences: first, the robot movement within the environment become uncertain, e.g., drifting due to different terrains, and blockage due to debris; second, the properties relevant to the task become uncertain, e.g., which areas contains human victims, and which charging station is functional. In other words, uncertainty in the environment model can cause non-determinism both in the task planning and motion execution. A formal consideration of this uncertainty when planning for general complex tasks remains challenging. Some recent work addresses this issue with model-based optimization [1], [2], [3] or data-driven online learning [4], [5]. However, the computational complexity remains to be the bottleneck and most validations are performed on systems with small number of states.

On the other hand, safety is an indispensable aspect to consider when deploying robotic systems, especially so in an uncertain and potentially unsafe environment. Depending on the application, robotic safety can be defined in various ways. Safety in motion planning [6] is commonly defined as the avoidance of a subset of the state space, namely the *unsafe* regions. This has been made more general with temporal task specifications, e.g., always stop in front of human, infinitely often charge itself, and always travel at the right lane, see [1], [4], [7], [8], [9]. These safety rules are useful and often directly imposed as an additional requirement of the overall task. However, there are certain types of safety requirements that can *not* be expressed in this way. Safe-return constraints require that the robot should be able to return to a set of safety states whenever requested during the mission. Such constraints are particularly important for non-ergodic systems where not all states are reachable for any other states. Considering the same example of search-and-rescue missions, the robot should avoid one-way doors, stairs and debris where it may get trapped in some rooms; during the planetary exploration mission, the robot should avoid steep cliffs and valleys where it can easily descend but hard to ascend back. In other words, such safe-return constraints are less state-dependent but more *policy-dependent*, thus more general in terms of expressiveness and practical relevance. More importantly, these constraints are often conflicting with the long-term task goals, e.g., the task specifies to search and rescue victims at all time, while the safe-return constraints require to return and stay at the base. Consequently, they can not be added directly in the task specification, instead a separate return policy should be

The authors are with [1]the State Key Laboratory for Turbulence and Complex Systems, Department of Mechanics and Engineering Science, College of Engineering, Peking University, Beijing, China; [2]the Academy of Military Sciences, Beijing, China. This work was supported by the National Natural Science Foundation of China under grants 62203017, T2121002, U2241214; by the Ministry of Education under grant 2021ZYA05004; and by Beijing Natural Science Foundation under grant JQ20025. Corresponding author: zhongkli@pku.edu.cn.

synthesized to fulfill the constraints and further restrict the task execution. To the best of our knowledge, such safe-return constraints have been mainly addressed in the reinforcement learning community [10], [11] to ensure safety during the learning process. They have not been studied along with the complex temporal tasks as in this work.

In this work, we address the motion planning problem over MDPs with probabilistic labels, where the tasks are given as Linear Temporal Logic (LTL) formulas. Furthermore, we take into account a general safety requirement as safe-return constraints, which demands the existence of a return-policy to drive the system back to home states with a high probability. It is first shown that the safe-return constraints can be re-formulated as accumulated rewards given the value function in the associated safety automaton. Then, the baseline solution is proposed by solving two coupled linear programs in the respective product automaton. However, this solution quickly becomes intractable when the system size increases. To overcome this bottleneck, a hierarchical and approximate planning algorithm based on the combination of symbolic and temporal abstractions with constrained optimization is proposed. Two abstracted semi-MDPs and the associated motion policies as *temporal options* are constructed simultaneously for the regions whose features are relevant to the task and safety specifications. Afterwards, two high-level task policies are synthesized in sequence within the respective product automata between the semi-MDPs and task automata. The return policy ensures that the safe-return constraints are fulfilled, and the outbound policy minimizes the overall cost to satisfy the task. It is shown that this hierarchical solution is more efficient and can solve systems of much larger sizes, compared with the alternative and baseline solutions, with only a slight reduction of optimality. The results are validated rigorously via simulations and hardware experiments of various robotic missions with different system sizes.

The main contribution lies in three aspects: (i) the formulation of a new planning problem for MDPs with labelling features, where both the tasks and safe-return constraints are given as complex LTL formulas; (ii) the theoretical analyses that ensure the correctness of the problem re-formulation; (iii) the hierarchical planning algorithm that synthesizes simultaneously and efficiently the outbound policy for task satisfaction and the return policy for safety constraints.

The rest of the paper is organized as follows: Sec. III introduces some preliminaries of LTL. The problem formulation is given in Sec. IV. Sec. V provides the theoretical analyses to re-formulate the problem and therefore the baseline solution. The hierarchical planning framework is presented in Sec. VI. Simulation and experiment results are shown in Sec. VII. Finally, Sec. VIII concludes with future work.

II. Related Work

A. MDPs with Temporal Tasks

MDPs provide a powerful model for robots acting in an uncertain environment. Uncertainty arises in various aspects of the system such as the properties of the workspace and the outcome of an action [12]. A common objective is to reach a set of terminal states while maximizing expected cumulative reward. When the underlying MDP is fully-known, a variety of algorithms can be applied to find the optimal acting policy for the robot, such as dynamic programming [13], [14], value iteration and policy iteration [12]. The resulting policy maps the current state to the set of optimal actions, deterministically or stochastically. Otherwise, if the underlying MDP is only partially-known, online learning algorithms [15] can be used. Furthermore, instead of the simple reachability task, there have been many efforts to address the same problem, rather to satisfy high-level temporal tasks specified in various formal languages, such as Probabilistic Computation Tree Logic (PCTL) in [16] and Linear Temporal Logics (LTL) in [1], [2], [17], [18]. Such languages are intuitive and expressive when specifying complex control tasks, such as surveillance, transportation and emergency response, see [8], [19], [20], [9]. Different cost optimizations are considered along with the task satisfiability, such as maximum reachability for constrained MDPs in [17], [18], the minimal bottleneck cost in [21], and pareto curves under multiple temporal objectives in [22]. Verification toolboxes are provided in [23], [24] for certain task formats. Moreover, robust control policies for a temporal task are studied when the underlying MDP is uncertain, for instance, [3] maximize the accumulated time-varying rewards, [25] maximizes the satisfiability under uncertain transition measures, and our previous work [1] allows for infeasible tasks to be only partially satisfied by the MDP. Lastly, some recent work builds upon methods from reinforcement learning to combine data-driven approach with the aforementioned model-based planning methods. For example, the work [4] constructs the product automaton on-the-fly while interacting with the environment. The work [5] instead proposes a reward shaping method to maximize the task satisfiability without directly learning the underlying transition model, while [26] relies on the actor-critic methods to approximate over large set of state-action pairs. Due to the exponential complexity and data inefficiency, most of the work above is validated over case studies of limited sizes. In this work, we assume the model as a known MDP with labelling features, to focus instead on the consideration of safe-return constraints and more importantly, feature-based symbolic and temporal abstraction methods to reduce the computational complexity.

B. Safety in Planning

Safety in motion planning [6] is commonly defined as the avoidance of a subset of the state space, namely the unsafe regions. In the literature for planning over temporal tasks, it can be more general and specified directly as an additional requirement in the task specification, e.g., “always avoid obstacles”, “infinitely often visit charging station” and “always stop in front of human”, see [1], [4], [7], [8], [9]. Consequently, they are treated similarly as other performance-related tasks. The work in [27], [28] specifies a variety of safety rules as sub-task formulas and synthesizes a minimum-violating policy for these rules. Our earlier work [8] proposes to separate the performance and safety requirements as soft and hard specification, respectively. The hard specifications have to be satisfied at all time and soft specifications can be improved gradually during exploration. In contrast, the safety constraints considered in this work as *safe-return* constraints, which require that the robot should be able to return to a set of safety states anytime when requested during the mission. They are particularly important for non-ergodic systems, and significantly different from the aforementioned definitions: (i) whether a region is unsafe depends on the existence of a return policy, rather on the region itself; (ii) two dependent polices, outbound policy and return policy, should be constructed, whereas traditionally only one policy is needed for both the task and safety specifications [1], [7], [8], [28]; (iii) the safe-return constraints are often conflicting with the task goal, thus can not be added directly to the task specification; (iv) traditional safety definitions mentioned above should be included in task specification, rather than the safe-return constraints. Such safe-return constraints have been shown to be useful during reinforcement learning, e.g., [10], [11]. However, they have not been studied in the context of complex temporal tasks.

C. Hierarchical Temporal and Symbolic Abstraction

Uniform discretization of a continuous high-dimensional system often leads to complexity explosion, thus is only applicable to toy cases. Temporal and symbolic abstraction techniques are proposed to address this complexity issue. The notion of *options* as closed-loop policies for taking actions over a period of time is pioneered in [29], which allows for high-level planning over the semi-MDPs for extended planning horizons. Feature-based or symbolic abstraction [30] is another effective way to tackle the planning problem sequentially at different levels of granularity. For instance, it is a common practice to plan first over regions of interest in the workspace given the temporal task, see [8], [17], [19], [20], which is then executed by the low-level feedback motion controller to navigate among these regions. Moreover, [31], [32] proposes an automated abstraction algorithm for a general nonlinear system to satisfy a LTL formula, [7] constructs a finite abstraction model as uncertain MDPs for the class of switched diffusion systems. Bi-simulation or approximation relations [33] are widely used to represent the relation between the original system and the abstraction. In this work, the feature-based symbolic and temporal abstractions are combined and applied to both the safe-return constraints and the task specifications, which are dependent and thus constructed simultaneously.

III. Preliminaries

A. Linear Temporal Logic (LTL)

The ingredients of a Linear Temporal Logic (LTL) formula are a set of atomic propositions AP and several Boolean and temporal operators. Atomic propositions are Boolean variables that can be either true or false. A LTL formula is specified according to the syntax [34]: $\varphi \triangleq \top \mid p \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid \bigcirc \varphi \mid \varphi_1 \,\mathsf{U}\, \varphi_2$, where $\top \triangleq \texttt{True}$, $p \in AP$, $\bigcirc$ (*next*), U (*until*) and $\bot \triangleq \neg\top$. For brevity, we omit the derivations of other operators like $\square$ (*always*), $\Diamond$ (*eventually*), $\Rightarrow$ (*implication*). The semantics of LTL is defined over the set of infinite words over 2^{AP}. Intuitively, $p \in AP$ is satisfied on a word $w = w(1)w(2)...$ if it holds at $w(1)$, i.e., if $p \in w(1)$. Formula $\bigcirc\varphi$ holds true if φ is satisfied on the word suffix that begins in the next position $w(2)$, whereas $\varphi_1 \mathsf{U} \varphi_2$ states that φ_1 has to be true until φ_2 becomes true. Finally, $\Diamond\varphi$ and $\square\varphi$ are true if φ holds on w eventually and always, respectively. Thus, given any word over AP, it can be verified whether w satisfies the formula, denoted by $w \models \varphi$. The full semantics and syntax of LTL are omitted here due to limited space, see e.g., [34].

B. Deterministic Rabin Automaton (DRA)

The set of words that satisfy a LTL formula φ over AP can be captured through a Deterministic Rabin Automaton (DRA) $\mathcal{A}_\varphi$ [34], defined as $\mathcal{A}_\varphi \triangleq (Q,\, 2^{AP},\, \delta,\, q_0,\, \text{Acc}_\mathcal{A})$, where Q is a set of states; 2^{AP} is the alphabet; $\delta \subseteq Q \times 2^{AP} \times Q$ is a transition relation; $q_0 \in Q$ is the initial state; and $\text{Acc}_\mathcal{A} \subseteq 2^Q \times 2^Q$ is a set of accepting pairs, i.e., $\text{Acc}_\mathcal{A} = \{(H^1_\mathcal{A}, I^1_\mathcal{A}), (H^2_\mathcal{A}, I^2_\mathcal{A}), \cdots, (H^N_\mathcal{A}, I^N_\mathcal{A})\}$ where $H^i_\mathcal{A},\, I^i_\mathcal{A} \subseteq Q$, $\forall i = 1, 2, \cdots, N$. An infinite run $q_0 q_1 q_2 \cdots$ of $\mathcal{A}$ is *accepting* if there exists *at least one* pair $(H^i_\mathcal{A},\, I^i_\mathcal{A}) \in \text{Acc}_\mathcal{A}$ such that $\exists n \geqslant 0$, it holds $\forall m \geqslant n,\, q_m \notin H^i_\mathcal{A}$ and $\overset{\infty}{\exists} k \geqslant 0,\, q_k \in I^i_\mathcal{A}$, where $\overset{\infty}{\exists}$ stands for "existing infinitely many". Namely, this run should intersect with $H^i_\mathcal{A}$ *finitely* many times while with $I^i_\mathcal{A}$ *infinitely* many times. There are translation tools [35] to obtain $\mathcal{A}_\varphi$ given φ with complexity $2^{2^{\mathcal{O}(|\varphi| \log |\varphi|)}}$.

IV. Problem Formulation

In this section, we formally define the considered system model, the task specification, the safe-return constraints, and the complete problem formulation.

A. Labeled MDP

In order to model different workspace properties, the definition of a standard MDP [12] is extended with a labeling function over the states, namely:

$$\mathcal{M} \triangleq \big(X,\, U,\, D,\, p_D,\, AP,\, L,\, c_D,\, x_0\big), \tag{1}$$

where X is the finite state space; U is the finite control action space (with a slight abuse of notation, $U(x)$ also denotes the set of control actions *allowed* at state $x \in X$); $D \triangleq \{(x,\, u) \mid x \in X,\, u \in U(x)\}$ is the set of allowed transitions as the possible state-action pairs; $p_D \colon X \times U \times X \to [0, 1]$ is the transition probability function for each transition in D, such that $\sum_{\check{x} \in X} p_D(x, u, \check{x}) = 1,\ \forall (x,\, u) \in D$; $c_D \colon D \to \mathbb{R}^{>0}$ is the cost function associated with an action; AP is a set of atomic propositions as the properties of interest; $L \colon X \to 2^{AP}$ returns the properties held at each state (or simply labels); and lastly $x_0 \in X,\ l_0 \in L(x_0)$ are the initial states and labels. Such models can be obtained by combining the robot motion controller with the environment model.

Remark 1. The above model can be extended to probabilistically-labeled MDPs, as proposed in our previous work [1]. It can incorporate probabilistic labels at each state, which are useful for modeling uncertain environments. Moreover, for large-scale systems, such model $\mathcal{M}$ can often be constructed algorithmically from data without much manual inputs, by combining the workspace data and the robot motion model, e.g., the office blueprint including walls and doors, and the satellite depth image of mountains and valleys. More details can be found in the simulation of Sec. VII. ■

Example 1. The robot motion for the search-and-rescue mission is modeled as follows: X is a set of regions with the desired granularity [6]; U maps to the navigation function among these regions; p_D is computed based on the feasibility of such navigation; L is the features such as different rooms and whether there are victims. ■

B. Task Specification

Different from the simple navigation task, we take into account a more complex task specification as LTL formulas φ over the same atomic propositions AP from (1) above. They can be used to specify both temporal and spatial requirements over the system, of which the exact syntax and properties are given in Sec. III-A. They are general enough to specify most high-level tasks such as surveillance, safety, service and response. Many useful templates can be found in related work [4], [7], [8], [28], [36], [37], [38].

Example 2. The task to surveil regions r_1, r_2, r_3 infinitely often can be specified as $\varphi = (\Box\Diamond r_1) \wedge (\Box\Diamond r_2) \wedge (\Box\Diamond r_3)$; The task to pick an object from region r_1 and drop it at r_2 is given by $\varphi = \Diamond((\texttt{pick} \wedge r_1) \wedge \Diamond(\texttt{drop} \wedge r_2))$; The task to provide supply at r_1 once a certain material is detected low is given by $\varphi = \Box(\texttt{low} \rightarrow \Diamond(\texttt{supply} \wedge r_1))$. ■

At stage $T \geqslant 0$, the robot's past path is given by $X_T = x_0 x_1 \cdots x_T \in X^{(T+1)}$, the past sequence of observed labels is given by $L_T = l_0 l_1 \cdots l_T \in (2^{AP})^{(T+1)}$ and the past sequence of control actions is $U_T = u_0 u_1 \cdots u_T \in U^{(T+1)}$. It should hold that $p_D(x_t, u_t, x_{t+1}) > 0$ and $l_t = L(x_t)$, $\forall t \geqslant 0$. The complete past is then given by $R_T = x_0 l_0 u_0 \cdots x_T l_T u_T$. Denote by $\boldsymbol{X}_T$, $\boldsymbol{L}_T$ and $\mathbf{R}_T$ the set of all possible past sequences of states, labels, and runs up to stage T. For brevity, $X_T \triangleq R_T|_X$ and $L_T \triangleq R_T|_L$ denote the sequence of states and labels associated with the run R_T. For infinite sequences, $T = \infty$ and $\boldsymbol{X}_\infty$, $\boldsymbol{L}_\infty$, $\mathbf{R}_\infty$ denote the set of all infinite sequences of states, labels, and runs, respectively. A *finite-memory* policy is defined as $\boldsymbol{\mu} = \mu_0 \mu_1 \cdots \mu_T$. The control policy at stage $t \geqslant 0$ is given by $\mu_t : \mathbf{R}_t \times U \rightarrow [0, 1]$, $\forall t \geqslant 0$. Denote by $\overline{\boldsymbol{\mu}}$ the set of all such finite-memory policies. Given a control policy $\boldsymbol{\mu} \in \overline{\boldsymbol{\mu}}$, the underlying system $\mathcal{M}$ evolves as a Markov Chain (MC), denoted by $\mathcal{M}|\boldsymbol{\mu}$. The probability measure on the smallest σ-algebra, over all possible infinite sequences within $\mathcal{M}|\boldsymbol{\mu}$ that contain R_T, is the unique measure [34]: $Pr^{\boldsymbol{\mu}}_{\mathcal{M}}(R_T) = \prod_{t=0}^{T} p_D(x_t,\, u_t,\, x_{t+1}) \cdot \mu_t(R_t, u_t)$, where $\mu_t(R_t, u_t)$ is defined as the probability of choosing action u_t given the past run $\mathbf{R}_t$. Then, the probability of $\mathcal{M}$ satisfying φ under a finite-memory policy $\boldsymbol{\mu}$ is defined by:

$$\mathbf{Sat}^{\boldsymbol{\mu}}_{\mathcal{M}} \triangleq Pr^{\boldsymbol{\mu}}_{\mathcal{M}}(\varphi) \triangleq Pr^{\boldsymbol{\mu}}_{\mathcal{M}}\big(R_\infty \in \mathbf{R}^{\boldsymbol{\mu}}_{\mathcal{M}} \,|\, R_\infty|_L \models \varphi\big), \quad (2)$$

where $\mathbf{R}^{\boldsymbol{\mu}}_{\mathcal{M}} \subset \mathbf{R}_\infty$ is the set of all infinite runs of system $\mathcal{M}$ under policy $\boldsymbol{\mu}$; $R_\infty|_L$ is the infinite sequence of labels associated with a run R_∞ in $\mathbf{R}^{\boldsymbol{\mu}}_{\mathcal{M}}$; and the satisfaction relation "$\models$" is introduced in Sec. III-A. Namely, the *satisfiability* equals to the probability of all infinite runs whose associated labels satisfy the task. More details on the probability measure can be found in [34]. Moreover, the *cost* of policy $\boldsymbol{\mu}$ over $\mathcal{M}$ is given by the expected mean cost of these infinite runs, namely:

$$\begin{aligned}\mathbf{Cost}^{\boldsymbol{\mu}}_{\mathcal{M}} &\triangleq \mathbb{E}_{R_\infty \in \mathbf{R}^{\boldsymbol{\mu}}_{\mathcal{M}}}\{\mathbf{Cost}(R_\infty)\} \\ &\triangleq \mathbb{E}_{R_\infty \in \mathbf{R}^{\boldsymbol{\mu}}_{\mathcal{M}}}\{\lim_{t\rightarrow\infty} \sum_{t=0}^{\infty} \frac{1}{t}\, c_D(x_t, u_t)\},\end{aligned} \quad (3)$$

where $\mathbf{Cost}(R_\infty)$ is the *mean* total cost [12] of an infinite run R_∞; and $c_D(\cdot)$ is the cost of applying u_t and x_t from (1). Given only the task, our previous work in [1] proposes an approach to optimize the above cost by formulating two dependent Linear Programs for the plan prefix and suffix.

Remark 2. Un-discounted cost summation is often used for the stochastic shortest path problems [14], [39], which are not suitable here as many general LTL tasks require liveness property over *infinite* runs. ■

C. Safe-Return Constraints

As discussed in Sec. I and II, traditionally safety is defined as the avoidance of unsafe states, see [6], which are *state-dependent* and mostly pre-defined offline. Despite its intuitiveness, it has serious drawbacks in scenarios where safety depends on whether the system can follow a certain strategy to return to the safe states, or where unsafe states can only be determined online during execution. Such safety measure is now *policy-dependent* and covers the traditional notion.

Formally, we consider the safe-return requirements φ_{r} specified as LTL formulas with the following format:

$$\varphi_{\mathrm{r}} = \varphi_{\mathrm{r}}^1 \wedge \Diamond\Box\varphi_{\mathrm{r}}^2, \quad (4)$$

where φ_{r}^1 are syntactically co-safe LTL (sc-LTL) formulas over the same propositions [34], [40]; φ_{r}^1 denotes the transient requirement while $\varphi_{\mathrm{r}}^2 = \bigvee_{l \in L_s} l$, which contains the set of labels $L_s \subset AP$ associated with a set of safe states that the system should *stay* in the end. Note that sc-LTL formulas can be satisfied by a good prefix of finite length [40]. Similar to (2), the probability of $\mathcal{M}$ satisfying φ_{r} under a finite-memory policy $\boldsymbol{\mu}_{\mathrm{r}} \in \overline{\boldsymbol{\mu}}$ is given by:

$$\begin{aligned}\mathbf{Sat}^{\boldsymbol{\mu}_{\mathrm{r}}}_{\mathcal{M}} \triangleq Pr^{\boldsymbol{\mu}_{\mathrm{r}}}_{\mathcal{M}}(\varphi_{\mathrm{r}}) \triangleq Pr^{\boldsymbol{\mu}_{\mathrm{r}}}_{\mathcal{M}}\big(R_\infty \in \mathbf{R}^{\boldsymbol{\mu}_{\mathrm{r}}}_{\mathcal{M}} \,|\, \\ L_1 L_2^\omega \models \varphi_{\mathrm{r}},\, R_\infty|_L = L_1 L_2^\omega\big),\end{aligned} \quad (5)$$

where $\mathbf{R}^{\boldsymbol{\mu}_{\mathrm{r}}}_{\mathcal{M}}$ is the set of infinite runs of system $\mathcal{M}$ under the policy $\boldsymbol{\mu}_{\mathrm{r}}$; one such run is denoted by R_∞ and its projection $R_\infty|_L$ has the prefix-suffix format of $L_1 L_2^\omega$, where L_1 is a finite prefix that satisfies φ_{r}^1 and L_2^ω is a cyclic suffix that satisfies φ_{r}^2 as defined in (4).

Example 3. For the search-and-rescue mission, it crucial that the robot can return to and stay at its base station via the designated exit, e.g., $\varphi_{\mathrm{r}} = \Diamond\texttt{ex} \wedge \Diamond\Box\texttt{bs}$. ■

Note that other safety constraints, such as collision avoidance, charging often, and emergency response, should be included in the task φ rather than safe-return constraints φ_{r}, as in [4], [7], [8], [9], [27], [28]. More importantly, there are now *two* finite-memory policies: $\boldsymbol{\mu}_\circ$ called the *outbound* policy that drives the robot to satisfy task φ; and $\boldsymbol{\mu}_{\mathtt{r}}$ called the *return* policy that ensures the safety constraints $\varphi_{\mathtt{r}}$. Due to the existence of the outbound policy, the satisfiability of system $\mathcal{M}$ w.r.t. the safe-return requirements $\varphi_{\mathtt{r}}$ in (5) is modified as follows:

$$\begin{aligned}\mathbf{Safe}_{\mathcal{M}}^{\boldsymbol{\mu}_\circ,\boldsymbol{\mu}_{\mathtt{r}}} &\triangleq \mathbf{Sat}_{\mathcal{M}'_\circ}^{\boldsymbol{\mu}_{\mathtt{r}}} = Pr_{\mathcal{M}'_\circ}^{\boldsymbol{\mu}_{\mathtt{r}}}(\varphi_{\mathtt{r}})\\ &= Pr_{\mathcal{M}'_\circ}^{\boldsymbol{\mu}_{\mathtt{r}}}\big(R_\infty \in \mathbf{R}_{\mathcal{M}'_\circ}^{\boldsymbol{\mu}_{\mathtt{r}}}\,|\,R_\infty|_L \models \varphi_{\mathtt{r}}\big),\end{aligned} \tag{6}$$

where $\mathcal{M}'_\circ$ is the modified model with an initial state distribution generated by system $\mathcal{M}$ under the outbound policy $\boldsymbol{\mu}_\circ$; the word $R_\infty|_L$ follows the prefix-suffix format as defined in (5). Thus, the safe-return constraints are defined as follows.

Definition 1. Given system $\mathcal{M}$, an outbound policy $\boldsymbol{\mu}_\circ$ is called $\chi_{\mathtt{r}}$*-safe* if there exists a *return* policy $\boldsymbol{\mu}_{\mathtt{r}}$ such that the probability of $\mathcal{M}$ satisfying $\varphi_{\mathtt{r}}$ is lower-bounded:

$$\mathbf{Safe}_{\mathcal{M}}^{\boldsymbol{\mu}_\circ,\boldsymbol{\mu}_{\mathtt{r}}} \geqslant \chi_{\mathtt{r}}, \tag{7}$$

or *equivalently*: given $\boldsymbol{\mu}_\circ$ and $\mathcal{M}$, $\exists \boldsymbol{\mu}_{\mathtt{r}}$, s.t. $\mathbf{Sat}_{\mathcal{M}|\boldsymbol{\mu}_\circ}^{\boldsymbol{\mu}_{\mathtt{r}}} \geqslant \chi_{\mathtt{r}}$, where $\chi_{\mathtt{r}} \in [0,\,1]$ is the given safety bound. ■

It is worth clarifying that the task requirement φ and safe-return constraints $\varphi_{\mathtt{r}}$ are *fundamentally* different from the multi-objective tasks considered in [18], [23]. This is due to the fact that in most cases there does not exist *any* policy $\boldsymbol{\mu}$ that can satisfy φ and $\varphi_{\mathtt{r}}$ simultaneously, no matter how their relative priorities are set. For instances, consider the surveillance task φ in Example 2 and the safety constraint $\varphi_{\mathtt{r}}$ in Example 3. They are mutually exclusive as φ requires it to surveil several regions infinitely often, while $\varphi_{\mathtt{r}}$ requires the system to return and stay at the base.

D. Problem Statement

Problem 1. Given the labeled MDP $\mathcal{M}$ from (1), the task φ and the safe-return requirement $\varphi_{\mathtt{r}}$, our goal is to synthesize the outbound policy $\boldsymbol{\mu}_\circ$ *and* the return policy $\boldsymbol{\mu}_{\mathtt{r}}$ that solve the constrained optimization below:

$$\begin{aligned}&\min_{\boldsymbol{\mu}_\circ,\boldsymbol{\mu}_{\mathtt{r}}\in\overline{\boldsymbol{\mu}}} \quad \mathbf{Cost}_{\mathcal{M}}^{\boldsymbol{\mu}_\circ}\\ &\text{s.t.} \quad \mathbf{Sat}_{\mathcal{M}}^{\boldsymbol{\mu}_\circ} \geqslant \chi_\circ \;\text{ and }\; \mathbf{Safe}_{\mathcal{M}}^{\boldsymbol{\mu}_\circ,\boldsymbol{\mu}_{\mathtt{r}}} \geqslant \chi_{\mathtt{r}},\end{aligned} \tag{8}$$

where $\chi_\circ, \chi_{\mathtt{r}} \in [0,\,1]$ are given lower bounds for satisfiability and safety in (2) and (7), respectively; the overall cost $\mathbf{Cost}_{\mathcal{M}}^{\boldsymbol{\mu}_\circ}$ for the task is defined in (3). ■

The above formulation has two implications: first, the synthesis of the outbound and inbound policies are coupled. The overall cost can not be optimized without ensuring *both* the task satisfiability and the safe-return constraint; second, the system can evolve either under the outbound policy $\boldsymbol{\mu}_\circ$ or the return policy $\boldsymbol{\mu}_{\mathtt{r}}$, of which the switch is triggered by an *external* "return" request. Furthermore, the lower bounds $\chi_\circ$, $\chi_{\mathtt{r}}$ can affect the obtained policy greatly. For a highly-uncertain workspace, $\chi_\circ$ should be set low while $\chi_{\mathtt{r}}$ should be high, as no valid outbound policies exist if $\chi_\circ$ is set too high. On the other hand, for fairly certain models, both $\chi_\circ, \chi_{\mathtt{r}}$ can be set high. More detailed discussions can be found in the numerical studies later in Sec. VII.

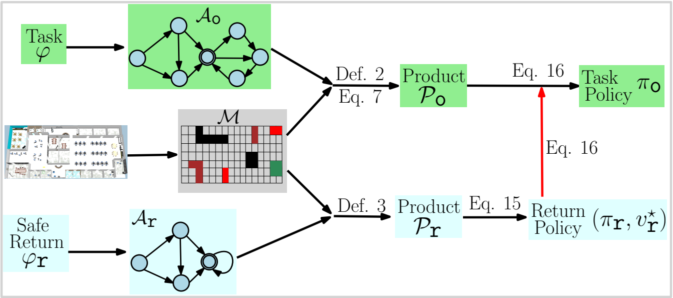


Fig. 1. Framework of the proposed baseline solution, which includes constructing the complete product automata $\mathcal{P}_{\mathtt{r}}$ and $\mathcal{P}_\circ$, and then solving two sequential optimizations.

Remark 3. Most related work synthesizes only *one* policy to satisfy φ and $\varphi_{\mathtt{r}}$ simultaneously, see [1], [4], [7], [8], [36]. This is however not directly possible as discussed after Def. 1, since the safe-constraints $\varphi_{\mathtt{r}}$ are conflicting with φ in most cases; Second, the outgoing policy $\boldsymbol{\mu}_\circ$ depends on the property of the return policy $\boldsymbol{\mu}_{\mathtt{r}}$ as in (7), thus can not be synthesized independently. Lastly, as elaborated earlier, only $\boldsymbol{\mu}_\circ$ is activated during execution, while $\boldsymbol{\mu}_{\mathtt{r}}$ may never be activated if not requested. In other words, $\varphi_{\mathtt{r}}$ serves more as a constraint, instead of an actual task. ■

V. Theoretical Analyses and Baseline Solutions

In this section, we first provide the theoretical analyses on how the task and safe-return constraints in (8) can be re-formulated as constrained reachability problems in the respective product automata. Based on these results, we propose the baseline solution that solves two sequential optimizations using linear programming (LP) within these product automata, as summarized in Fig. 1, Lastly, we show that this solution quickly becomes intractable as the system size grows.

A. Product Automaton and AMECs

As introduced in Sec. III-B, we can construct the DRA $\mathcal{A}_\circ$ associated with the task formula φ via the translation tools [35]. Denote it by $\mathcal{A}_\circ = (Q,\, 2^{AP},\, \delta,\, q_0,\, \text{Acc}_{\mathcal{A}})$, where the detailed notations are omitted here. Then we can construct a product automaton [34] between $\mathcal{M}$ and $\mathcal{A}_\circ$.

Definition 2. The product $\mathcal{P}_\circ \triangleq \mathcal{M} \times \mathcal{A}_\circ$ is a 7-tuple:

$$\mathcal{P}_\circ = (S,\, U,\, E,\, p_E,\, c_E,\, s_0,\, \text{Acc}_{\mathcal{P}}), \tag{9}$$

where the state $S \subseteq X \times Q$ satisfies $\langle x,\, q\rangle \in S$, $\forall x \in X$ and $\forall q \in Q$; the action set U is the same as in (1) and $U(s) = U(x)$, $\forall s = \langle x, q\rangle \in S$; $E = \{(s,u)\,|\,s \in S,\, u \in U(s)\}$; the transition probability $p_E\colon S \times U \times S \to [0,\,1]$ is defined by

$$p_E\big(\langle x, q\rangle,\, u,\, \langle \check{x}, \check{q}\rangle\big) = p_D(x,\, u,\, \check{x}), \tag{10}$$

where (i) $\langle x, q\rangle, \langle \check{x}, \check{q}\rangle \in S$; (ii) $(x, u) \in D$; and (iii) $\check{q} = \delta(q,\, L(x))$. The label l fulfills the condition from q to $\check{q}$ in $\mathcal{A}_\circ$; the cost function $c_E\colon E \to \mathbb{R}^{>0}$ is given by $c_E\big(\langle x, q\rangle,\, u\big) = c_D(x, u)$, $\forall(\langle x, q\rangle,\, u) \in E$; the initial state is $s_0 = \langle x_0, q_0\rangle \in S$; the accepting pairs are defined as $\text{Acc}_{\mathcal{P}} = \{(H^i_{\mathcal{P}},\, I^i_{\mathcal{P}}), i =$

$1, \cdots, N\}$, where $H^i_{\mathcal{P}} = \{\langle x, q\rangle \in S \,|\, q \in H^i_{\mathcal{A}}\}$ and $I^i_{\mathcal{P}} = \{\langle x, q\rangle \in S \,|\, q \in I^i_{\mathcal{A}}\}$, $\forall i = 1, \cdots, N$. ■

The product $\mathcal{P}_\circ$ computes the intersection between the traces of $\mathcal{M}$ and the words of $\mathcal{A}_\circ$, to find the admissible robot behaviors that satisfy the task φ. Furthermore, the Rabin accepting condition of $\mathcal{P}_\circ$ is the same as in Sec. III-B. To transform this condition into equivalent graph properties, we first compute the accepting maximum end components (AMECs) of $\mathcal{P}_\circ$ associated with its accepting pairs $\text{Acc}_{\mathcal{P}}$. Denote by $\Xi_{acc} = \{S'_1, U'_1), \cdots (S'_C, U'_C)\}$ the set of AMECs associated with $\text{Acc}_{\mathcal{P}}$, where $S'_c \subset S$ and $U'_c : S'_c \to 2^U$, $\forall c = 1, \cdots, C$. Simply speaking, S'_c is the set of states the robot should converge to, while U'_c specifies the allowed actions at each state $s \in S'_c$ to remain inside S'_c. Note that $S'_{c_1} \cap S'_{c_2} = \varnothing$, $\forall c_1, c_2 = 1, \cdots, C$. Denote by

$$S^{\circ}_{\Xi} \triangleq \bigcup_{c=1}^{C} S'_c, \tag{11}$$

where $(S'_c, U'_c) \in \Xi_{acc}$ the union of all AMECs associated with $\mathcal{P}_\circ$. We omit the derivation of Ξ_{acc} here and refer the readers to Definitions 10.116, 10.117 and 10.124 of [34] for theoretical details and [41] for software implementation.

Definition 3. Let $\mathcal{A}_\mathrm{r}$ be the DRA associated with the safe-return constraints φ_r. Then the product $\mathcal{P}_\mathrm{r} \triangleq \mathcal{M} \times \mathcal{A}_\mathrm{r}$ can be constructed analogously as $\mathcal{P}_\circ$ above, of which the details are omitted here. ■

Note that we omit the subscripts for all elements in $\mathcal{P}_\circ$ and $\mathcal{P}_\mathrm{r}$ above for clarity. Subscripts will be added whenever necessary to distinguish the same elements in different product automata. Furthermore, the control policies for the product automata $\mathcal{P}_\circ$ and $\mathcal{P}_\mathrm{r}$ are denoted by $\boldsymbol{\pi}_\circ$ and $\boldsymbol{\pi}_\mathrm{r}$, respectively. Note that due to the deterministic nature of DRA, a policy in the product automaton, e.g., $\boldsymbol{\pi}_\circ$ and $\boldsymbol{\pi}_\mathrm{r}$, can be *uniquely* mapped to a policy, e.g., $\boldsymbol{\mu}_\circ$ and $\boldsymbol{\mu}_\mathrm{r}$ in $\mathcal{M}$, and vice versa.

B. Task Satisfiability Reformulation

Given the product $\mathcal{P}_\circ$, the following theorem is commonly used in related work [1], [2], [17], [42], [18] to convert the task satisfiability to the reachability in $\mathcal{P}_\circ$.

Theorem 1. *The probability of φ being satisfied by $\mathcal{M}$ under policy $\boldsymbol{\mu}_\circ$ at stage 0 can be re-formulated as:*

$$\textbf{Sat}^{\boldsymbol{\mu}_\circ}_{\mathcal{M}} = \mathbb{E}^{\boldsymbol{\pi}_\circ}_{\widetilde{\mathcal{P}}_\circ} \left\{ \sum_{t=0}^{\infty} b_{s_t, S^{\circ}_{\Xi}} \right\}, \tag{12}$$

where $\widetilde{\mathcal{P}}_\circ \triangleq (1 - b_{s,S^{\circ}_{\Xi}}) \cdot \mathcal{P}_\circ$, $b_{s,S^{\circ}_{\Xi}} \triangleq \mathbb{1}_{\{s \in S^{\circ}_{\Xi}\}}$ is an indicator function; S°_{Ξ} is defined in (11)*; and $\boldsymbol{\pi}_\circ$ is the policy for $\widetilde{\mathcal{P}}_\circ$ corresponding to $\boldsymbol{\mu}_\circ$.*

Proof. It has been proven in [1], [18], [34] that once the system $\mathcal{M}$ enters the union set of AMECs S°_{Ξ} in $\mathcal{P}_\circ$, it can remain inside and satisfy the accepting condition of $\mathcal{P}_\circ$ by following the transition conditions given by U'_c, e.g., the Round-robin policy [34] or the balanced policy [1]. In other words, the probability of satisfying φ equals to the probability of entering S°_{Ξ} *at least once* in $\mathcal{P}_\circ$. Note that $\widetilde{\mathcal{P}}_\circ$ has the same structure as $\mathcal{P}_\circ$. But once the system reaches the set S°_{Ξ} in $\widetilde{\mathcal{P}}_\circ$, any further actions lead to a virtual "end" state with only self-loop. Thus, the left-hand side of (12) is computed by:

$$\textbf{Sat}^{\boldsymbol{\mu}_\circ}_{\mathcal{M}} = \mathbb{E}^{\boldsymbol{\pi}_\circ}_{\mathcal{P}_\circ} \left\{ \mathbb{1}_{\{\exists t < \infty, s_t \in S^{\circ}_{\Xi}\}} \right\} = \mathbb{E}^{\boldsymbol{\pi}_\circ}_{\widetilde{\mathcal{P}}_\circ} \left\{ \sum_{t=0}^{\infty} b_{s_t, S^{\circ}_{\Xi}} \right\}, \tag{13}$$

which completes the proof. □

C. Safe-return Constraints Reformulation

The safe-return constraints in (7) however have not been studied before in related work. The following theorem is essential to re-formulate such constraints.

Theorem 2. *The safe-return constraints in* (7) *for $\mathcal{M}$ under policies $\boldsymbol{\mu}_\circ$ and $\boldsymbol{\mu}_\mathrm{r}$ at stage 0 can be re-formulated as:*

$$\textbf{Safe}^{\boldsymbol{\mu}_\circ, \boldsymbol{\mu}_\mathrm{r}}_{\mathcal{M}} = \mathbb{E}^{\boldsymbol{\pi}_\circ}_{\mathcal{P}_\circ} \left\{ \sum_{t=0}^{\infty} v^\star_\mathrm{r}(s^\mathrm{r}_t) \right\}, \tag{14a}$$

$$\textit{where } v^\star_\mathrm{r}(s^\mathrm{r}_t) = \mathbb{E}^{s^\mathrm{r}_t, \boldsymbol{\pi}_\mathrm{r}}_{\widetilde{\mathcal{P}}_\mathrm{r}} \left\{ \sum_{\ell=t}^{\infty} b_{s^\mathrm{r}_\ell, S^\mathrm{r}_{\Xi}} \right\}, \tag{14b}$$

where $\widetilde{\mathcal{P}}_\mathrm{r} \triangleq (1 - b_{s,S^\mathrm{r}_{\Xi}}) \cdot \mathcal{P}_\mathrm{r}$, $b_{s,S^\mathrm{r}_{\Xi}} \triangleq \mathbb{1}_{\{s \in S^\mathrm{r}_{\Xi}\}}$ is an indicator function, and S^r_{Ξ} is defined in (11)*; $v^\star_\mathrm{r} \in [0,\, 1]$ is the value function of $\widetilde{\mathcal{P}}_\mathrm{r}$ under policy $\boldsymbol{\pi}_\mathrm{r}$; the product state of $\mathcal{P}_\circ$ at stage t is denoted by $s_t = \langle x_t, q^\circ_t\rangle$, of which the associated product state in $\widetilde{\mathcal{P}}_\mathrm{r}$ is given by $s^\mathrm{r}_t = \langle x_t, q^\mathrm{r}_0\rangle$; and $\boldsymbol{\pi}_\circ, \boldsymbol{\pi}_\mathrm{r}$ are the policies for $\mathcal{P}_\circ$ and $\mathcal{P}_\mathrm{r}$, corresponding to $\boldsymbol{\mu}_\circ, \boldsymbol{\mu}_\mathrm{r}$ for $\mathcal{M}$, respectively.*

Proof. The safe-return constraints can be expanded by *splitting* the evolution of system $\mathcal{M}$ before and after the time when the robot is requested to return, then starts following the return policy. Without loss of generality, denote by $t \geqslant 0$ this time instant. Thus, $\mathcal{M}$ evolves under the outbound policy $\boldsymbol{\mu}_\circ$ to satisfy φ between time $[0,\, t)$, then under the return policy $\boldsymbol{\mu}_\mathrm{r}$ to satisfy φ_r from time $\ell \in [t, \infty)$. Thus, the definition of safety in (7) can be expanded as follows:

$$\textbf{Safe}^{\boldsymbol{\mu}_\circ, \boldsymbol{\mu}_\mathrm{r}}_{\mathcal{M}} = \mathbb{E}^{\boldsymbol{\pi}_\circ}_{\mathcal{P}_\circ} \left\{ \sum_{t=0}^{\infty} \textbf{Sat}^{x_t, \boldsymbol{\mu}_\mathrm{r}}_{\mathcal{M}} \right\}, \tag{15}$$

where $s_t = \langle x_t, q^\circ_t\rangle$ is the product state of $\mathcal{P}_\circ$ at stage t; and $\textbf{Sat}^{x_t, \boldsymbol{\mu}_\mathrm{r}}_{\mathcal{M}}$ is defined analogously to (12) as the satisfiability of φ_r under policy $\boldsymbol{\mu}_\mathrm{r}$, for system $\mathcal{M}$ but with a modified initial state x_t at stage t. Via the same argumentation as in Theorem 1, the probability of satisfying φ_r equals to the probability of entering the union set S^r_{Ξ} of $\mathcal{P}_\mathrm{r}$ at least once as time approaches infinity. Then, consider that $\widetilde{\mathcal{P}}_\mathrm{r}$ defined above has the same structure as $\mathcal{P}_\mathrm{r}$ except that once the system reaches S^r_{Ξ}, any further actions lead immediately to a virtual "end" state with only self-loop. Thus it holds that:

$$\begin{aligned} \textbf{Sat}^{x_t, \boldsymbol{\mu}_\mathrm{r}}_{\mathcal{M}} &= \mathbb{E}^{s^\mathrm{r}_t, \boldsymbol{\pi}_\mathrm{r}}_{\mathcal{P}_\mathrm{r}} \left\{ \mathbb{1}_{\{\exists \ell \in [t,\, \infty),\, s^\mathrm{r}_\ell \in S^\mathrm{r}_{\Xi}\}} \right\} \\ &= \mathbb{E}^{s^\mathrm{r}_t, \boldsymbol{\pi}_\mathrm{r}}_{\widetilde{\mathcal{P}}_\mathrm{r}} \left\{ \sum_{\ell=t}^{\infty} b_{s^\mathrm{r}_\ell, S^\mathrm{r}_{\Xi}} \right\} = v^\star_\mathrm{r}(s^\mathrm{r}_t), \end{aligned} \tag{16}$$

where $s^\mathrm{r}_t = \langle x_t, q^\mathrm{r}_0\rangle$ is the associated initial product state of $\mathcal{P}_\mathrm{r}$ at stage t. This mapping is necessary as the states in $\mathcal{A}_\circ$ and

Algorithm 1: Baseline Solution

```
Input: $\mathcal{M}$, $(\varphi, \chi_{\circ})$, $(\varphi_{\mathrm{r}}, \chi_{\mathrm{r}})$.
Output: $(\mathcal{P}_{\mathrm{r}}, \boldsymbol{\pi}_{\mathrm{r}}, v_{\mathrm{r}}^{\star})$, $(\mathcal{P}_{\circ}, \boldsymbol{\pi}_{\circ})$.
Build product $\mathcal{P}_{\mathrm{r}}$ and its AMECs $S_{\Xi}^{\mathrm{r}}$ ;   // Def. 3
Synthesize policy $\boldsymbol{\pi}_{\mathrm{r}}$ and its value function $v_{\mathrm{r}}^{\star}$ ;
  // Eq. (17)
Build product $\mathcal{P}_{\circ}$ and its AMECs $S_{\Xi}^{\circ}$ ;   // Def. 2
Synthesize policy $\boldsymbol{\pi}_{\circ}$ given $v_{\mathrm{r}}^{\star}$ ;          // Eq. (18)
```

$\mathcal{A}_{\mathrm{r}}$ can be different; and $v_{\mathrm{r}}^{\star}$ is by definition the value function of $\widetilde{\mathcal{P}}_{\mathrm{r}}$ under policy $\boldsymbol{\pi}_{\mathrm{r}}$. Since $\boldsymbol{\mu}_{\mathrm{r}}$ optimizes the probability of satisfying φ_{r}, it holds that its value function $v_{\mathrm{r}}^{\star}(s) \in [0, 1]$, $\forall s \in S_{\mathrm{r}}$. This completes the proof. □

Theorems 1 and 2 provide theoretical guarantees on the reformulation of both constraints in Problem 1. More specifically, Theorem 1 states that the task satisfiability can be ensured by limiting the reachability of S_{Ξ}° within $\widetilde{\mathcal{P}}_{\circ}$ under policy $\boldsymbol{\pi}_{\circ}$. Theorem 2 states that the safe-return constraint can be evaluated as the expected *rewards* of $\widetilde{\mathcal{P}}_{\circ}$ under policy $\boldsymbol{\pi}_{\circ}$, where the rewards are given by the value function of $\widetilde{\mathcal{P}}_{\mathrm{r}}$ under the return policy $\boldsymbol{\pi}_{\mathrm{r}}$ by maximizing the reachability of S_{Ξ}^{r}.

D. Baseline Solution and Computational Complexity

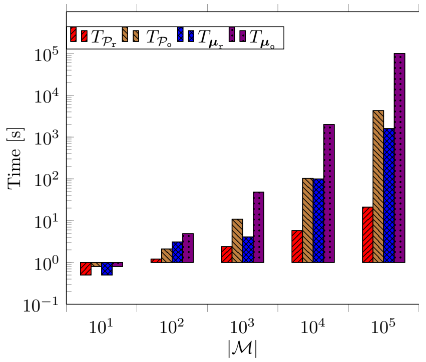
Consequently, the original problem after re-formulation can be solved by two sequential optimizations: (i) synthesize the optimal return policy $\boldsymbol{\pi}_{\mathrm{r}}$ by solving the optimization:

$$\max_{\boldsymbol{\mu}_{\mathrm{r}} \in \overline{\boldsymbol{\mu}}} \ \textbf{Sat}_{\mathcal{M}}^{\boldsymbol{\mu}_{\mathrm{r}}} = \max_{\boldsymbol{\pi}_{\mathrm{r}} \in \overline{\boldsymbol{\pi}}} \ \mathbb{E}_{\widetilde{\mathcal{P}}_{\mathrm{r}}}^{\boldsymbol{\pi}_{\mathrm{r}}} \left\{ \sum_{\ell=0}^{\infty} b_{s_{\ell}, S_{\Xi}^{\mathrm{r}}} \right\}, \tag{17}$$

i.e., to maximize the reachability of S_{Ξ}^{r} in $\widetilde{\mathcal{P}}_{\mathrm{r}}$. This amounts to solving a standard MDP problem without constraints, e.g., via LP. Since $\widetilde{\mathcal{P}}_{\mathrm{r}}$ becomes a Markov Chain under policy $\boldsymbol{\pi}_{\mathrm{r}}$, the associated value function $v_{\mathrm{r}}^{\star}$ can be easily computed, e.g., via value iteration; (ii) synthesize the optimal outbound policy $\boldsymbol{\pi}_{\circ}$ by solving the following constrained optimization:

$$\begin{aligned} \min_{\boldsymbol{\pi}_{\circ} \in \overline{\boldsymbol{\pi}}} \quad & \textbf{Cost}_{\widetilde{\mathcal{P}}_{\circ}}^{\boldsymbol{\pi}_{\circ}} \\ \text{s.t.} \quad & \mathbb{E}_{\widetilde{\mathcal{P}}_{\circ}}^{\boldsymbol{\pi}_{\circ}} \left\{ \sum_{t=0}^{\infty} b_{s_t, S_{\Xi}^{\circ}} \right\} \geqslant \chi_{\circ}, \\ & \mathbb{E}_{\widetilde{\mathcal{P}}_{\circ}}^{\boldsymbol{\pi}_{\circ}} \left\{ \sum_{t=0}^{\infty} v_{\mathrm{r}}^{\star}(s_t^{\mathrm{r}}) \right\} \geqslant \chi_{\mathrm{r}}, \end{aligned} \tag{18}$$

where the relevant notations are defined in (12) and (14). The above problem amounts to a *constrained* MDP problem [43], [44]. As a result, a methods similar to our earlier work [1] can be applied to synthesize the optimal prefix and suffix policies of $\boldsymbol{\pi}_{\circ}$. Briefly speaking, a constrained LP is formulated over the occupancy measure over all state-action pairs, such that (i) the summation of occupancy measure entering S_{Ξ}° is larger than $\chi_{\circ}$; (ii) the summation of occupancy measure multiplied by the associated value function over all states is larger than χ_{r}; and (iii) the summation of occupancy measure multiplied by the transition cost within S_{Ξ}° is minimized. The resulting outbound policy $\boldsymbol{\pi}_{\circ}$ is a stochastic policy over $\widetilde{\mathcal{P}}_{\circ}$, while the return policy $\boldsymbol{\pi}_{\mathrm{r}}$ is a stochastic policy over $\widetilde{\mathcal{P}}_{\mathrm{r}}$. We refer the readers to the supplementary material for the detailed LP formulation and policy derivation. It is worth noting that the mean cost minimization only applies to the outgoing policy $\boldsymbol{\mu}_{\circ}$ not the return policy $\boldsymbol{\mu}_{\circ}$ as formulated in (8). The above procedure is summarized in Fig. 1 and Alg. 1.



$\lvert\mathcal{M}\rvert$	$\lvert\mathcal{P}_{\mathrm{r}}\rvert$	$\lvert\mathcal{P}_{\circ}\rvert$	$T_{\boldsymbol{\mu}_{\mathrm{r}}}[s]$	$T_{\boldsymbol{\mu}_{\circ}}[\mathrm{s}]$
$(1\mathrm{e}2, 8\mathrm{e}2)$	$(5\mathrm{e}2, 4\mathrm{e}3)$	$(4\mathrm{e}3, 5\mathrm{e}4)$	0.1	0.9
$(5\mathrm{e}2, 4\mathrm{e}3)$	$(3\mathrm{e}3, 3\mathrm{e}4)$	$(2\mathrm{e}4, 2\mathrm{e}5)$	2.7	5.9
$(2\mathrm{e}3, 2\mathrm{e}4)$	$(1\mathrm{e}4, 1\mathrm{e}5)$	$(6\mathrm{e}4, 6\mathrm{e}5)$	4.1	48
$(7\mathrm{e}3, 7\mathrm{e}4)$	$(5\mathrm{e}4, 4\mathrm{e}5)$	$(2\mathrm{e}5, 3\mathrm{e}6)$	1e2	2e3
$(3\mathrm{e}4, 3\mathrm{e}5)$	$(2\mathrm{e}5, 3\mathrm{e}6)$	$(9\mathrm{e}5, 8\mathrm{e}6)$	2e3	N/A
$(1\mathrm{e}5, 2\mathrm{e}6)$	$(7\mathrm{e}5, 8\mathrm{e}6)$	N/A	N/A	N/A

Fig. 2. **Top**: Size of product $\mathcal{P}_{\mathrm{r}}$, $\mathcal{P}_{\circ}$ (measured by the number of nodes and edges); and computation time of polices $\boldsymbol{\mu}_{\mathrm{r}}$, $\boldsymbol{\mu}_{\circ}$, where $a\mathrm{e}b \triangleq a \times 10^{b}$. Note that 10^{4}s ≈ 2.7h, whereas“N/A” indicates either insufficient memory error or computation longer than 24 hours; **Bottom**: Plot of model computation time and solution time w.r.t. the size of the underlying $\mathcal{M}$.

Note that the product $\mathcal{P}_{\mathrm{r}}$ has the approximate size of $|2^{\varphi_{\mathrm{r}}}| \cdot |\mathcal{M}|$, of which the AMECs are computed in polynomial time. The optimal return policy $\boldsymbol{\pi}_{\mathrm{r}}$ can be synthesized in polynomial time also due to LP. Similar analysis holds also for $\mathcal{P}_{\mathrm{r}}$. Nonetheless, for high dimensional states, large workspaces and complex tasks considered in practice, the LPs above can become *intractable* to even construct, let alone to solve. Some examples are given below to illustrate the computation time blow-up with increasing system size, e.g., for a moderate size of $\mathcal{M}$ (around 10^{5} edges), it takes more than 24 hours to synthesize the outgoing policy.

Example 4. Consider the surveillance task in Example 2, the associated DRA has 32 states, 282 edges and 1 accepting pairs, while the DRA associated with the safe-return task in Example 3 has 7 states, 14 edges and 1 accepting pairs. Fig. 2 summarizes how $\mathcal{P}_{\mathrm{r}}$ and $\mathcal{P}_{\circ}$ increase in size, when the underlying model $\mathcal{M}$ grows. It is apparent that the baseline solution quickly becomes intractable as even formulating the LPs takes hours. For MDPs with more than 10^{6} edges, neither the return policy nor the outbound policy can be computed within reasonable amount of memory and time. ■

E. Alternative Baseline

Another straightforward but approximate method as an alternative baseline to the above baseline is to build an extended model $\widehat{\mathcal{M}}$ of the original $\mathcal{M}$ in (1) by adding another dimension, i.e., $\widehat{X} = \{\langle x, i \rangle, \forall x \in X, i \in \{0, 1\}\}$, where i indicates whether a return has been requested. More specifically, when $i = 0$, the system $\widehat{\mathcal{M}}$ evolves according to the outbound policy $\boldsymbol{\mu}_{\mathrm{o}}$; when $i = 1$, the system $\widehat{\mathcal{M}}$ evolves under the return policy $\boldsymbol{\mu}_{\mathrm{r}}$. In this way, both the outbound and return policies might be integrated into *one* policy as they act on different states in $\widehat{\mathcal{M}}$. However, since this return request is an external signal and not controlled by the robot, an *approximation* would be to modify the transition probability of $\mathcal{M}$ as follows: (i) $\widehat{p}_D(\langle x, 0\rangle, u, \langle \hat{x}, \hat{i}\rangle) = p_D(x, u, \hat{x}) \cdot 0.5$, $\forall \hat{i} \in \{0, 1\}$ and $\forall (x, u) \in D$; (ii) $\widehat{p}_D(\langle x, 1\rangle, u, \langle \hat{x}, 1\rangle) = p_D(x, u, \hat{x})$, $\forall (x, u) \in D$. In other words, the system can transit non-deterministically within level $i = 0$ or from level $i = 0$ to $i = 1$ at each time step, but not in the reversing order. Then, via the multi-objective probabilistic model checking algorithm proposed in [22] and available in PRISM [24], one common policy, denoted by $\widehat{\boldsymbol{\mu}}$, can be synthesized to satisfy both the objectives that $Pr^{\widehat{\boldsymbol{\mu}}}_{\widehat{\mathcal{M}}}(\varphi_{\mathrm{r}}) \geqslant \chi_{\mathrm{r}}$ for $i = 0$ and $Pr^{\widehat{\boldsymbol{\mu}}}_{\widehat{\mathcal{M}}}(\varphi) \geqslant \chi_{\mathrm{o}}$ for $i = 1$, while minimizing the rewards in (8). More details can be found in the supplementary material.

Since the extended model $\widehat{\mathcal{M}}$ has $2|X|$ nodes and $3|U|$ edges, the associated product automaton, denoted by $\widehat{\mathcal{P}}$, has roughly the size of $3|U| \cdot 2^{|\varphi|} \cdot 2^{|\varphi_{\mathrm{r}}|}$, see [22] for the method to construct $\widehat{\mathcal{P}}$. The computational complexity of this alternative solution can be estimated as $\mathcal{O}((3|U| \cdot 2^{|\varphi|} \cdot 2^{|\varphi_{\mathrm{r}}|})^{N_c})$, i.e., at least $(3^{N_c} \cdot 2^{|\varphi_{\mathrm{r}}|})$ times the size of the constrained optimization in (18), with $N_c > 2.3$ being the currently-known best algorithm to solve a LP. In other words, this alternative solution is at least a magnitude more expensive than the baseline solution proposed in the section. This difference can be even more significant if the specifications φ_{r}, φ are complex, see Sec. VII-C for detailed comparisons.

VI. Proposed Hierarchical Solution

To tackle the intractable complexity of the baseline solution, the main hierarchical solution is proposed in this section. As illustrated in Fig. 3, a two-step paradigm similar to the baseline solution is followed. It follows the framework of semi-MDPs and options as proposed in [29]. The major difference is that two feature-based symbolic and temporal abstraction of system $\mathcal{M}$ as semi-MDPs are constructed for the task specification and the safe-return constraints, respectively. Given these models, a hierarchical planning algorithm is proposed to synthesize both the return policy and outgoing policy as options that solve the original problem.

A. Hierarchical Planning for Safe-return Constraints

As discussed before, the safe-return policy has to be synthesized first along with its value function, which is then used to compute the outbound policy. Thus, we first describe how the safe-return policy can be synthesized hierarchically.

1) Labeled Semi-MDPs for Safe-return Constraints: Given the DRA $\mathcal{A}_{\mathrm{r}} = (Q, 2^{AP}, \delta, q_0, \mathrm{Acc}_{\mathcal{A}})$ associated with φ_{r}, we can compute the set of *effective* features:

$$\Theta_{\mathrm{r}} = \{\theta \in 2^{AP} \mid \exists q, \check{q} \in Q, \text{ s.t. } (q, \theta, \check{q}) \in \delta, \, q \neq \check{q}\}, \quad (19)$$

which includes any label inside Θ_{r} that can drive a transition within $\mathcal{A}_{\mathrm{r}}$, excluding self-transitions. Then, the feature-based abstraction of the original MDP $\mathcal{M}$ for φ_{r} as semi-MDPs, denoted by $\mathcal{M}'_{\mathrm{r}}$, can be constructed as another labeled MDP, namely:

$$\mathcal{M}'_{\mathrm{r}} \triangleq \big(X'_{\mathrm{r}}, \, U'_{\mathrm{r}}, \, p'_{\mathrm{r},D}, \, AP_{\mathrm{r}}, \, L'_{\mathrm{r}}\big), \quad (20)$$

where $X'_{\mathrm{r}} = \{x \in X \mid L(x) \cap \Theta_{\mathrm{r}} \neq \varnothing\} \subset X$ contains *only* the states that can potentially lead to a transition within $\mathcal{A}_{\mathrm{r}}$; AP_{r}, L'_{r} are defined analogously as in (1); U'_{r} is the set of macro actions that represent symbolically the underlying motion policies for each transition in X'_{r}; $p'_{\mathrm{r},D} : X'_{\mathrm{r}} \times U'_{\mathrm{r}} \times X'_{\mathrm{r}} \to [0, 1]$ is the transition probability for each transition. The derivation of $p'_{\mathrm{r},D}$ is explained below.

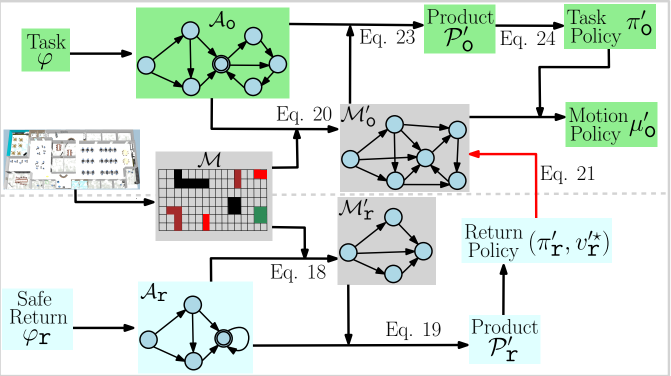

Task
φ
$\mathcal{A}_{\mathrm{o}}$
Product
$\mathcal{P}'_{\mathrm{o}}$
Eq. 23
Eq. 24
Task Policy π'_{o}
Eq. 20
$\mathcal{M}'_{\mathrm{o}}$
Motion Policy μ'_{o}
$\mathcal{M}$
Eq. 21
$\mathcal{M}'_{\mathrm{r}}$
Return Policy $(\pi'_{\mathrm{r}}, v'^{\star}_{\mathrm{r}})$
Eq. 18
$\mathcal{A}_{\mathrm{r}}$
Safe Return
φ_{r}
Eq. 19
Product
$\mathcal{P}'_{\mathrm{r}}$

Fig. 3. Illustration of the proposed hierarchical planning framework. Compared with the baseline solution in Fig. 1, two feature-based abstraction models $\mathcal{M}'_{\mathrm{o}}$, $\mathcal{M}'_{\mathrm{r}}$ are constructed first for the task specification and the safe-return constraints, respectively.

Problem 2. Determine U'_{r}, $p'_{\mathrm{r},D}$ in (20), given $\mathcal{M}$. ■

For *each* pair of states $(x_f, \, x_t) \in X'_{\mathrm{r}} \times X'_{\mathrm{r}}$, the associated macro action is symbolically denoted by $(x_f, \, x_t) \in U'_{\mathrm{r}}$. As illustrated in Fig. 4, the transition probability $p'_{\mathrm{r},D}$ is computed in three steps: (i) Construct modified MDP $\mathcal{M}'_{\mathrm{r}}(x_f, \, x_t)$ from $\mathcal{M}$, such that state x_f is the "source" state, and all the other states in X'_{r} (including x_t) are the "sink" states. Once the system enters any sink state it will stay there via self-loop with probability one and cost zero; (ii) Find the motion policy in $\mathcal{M}'_{\mathrm{r}}(x_f, \, x_t)$ that drives the system from x_f to x_t with the *maximum* probability. Denote by $\boldsymbol{\mu}'_{\mathrm{r}}(x_f, \, x_t)$ this specific motion policy, which is analogous to the *options* in [29]. Similar to the maximum reachability problem discussed in (17) before, this can be readily solved by formulating the LP over the occupancy measures of each state-action pair in $\mathcal{M}'_{\mathrm{r}}(x_f, \, x_t)$; (iii) Given the policy $\boldsymbol{\mu}'_{\mathrm{r}}(x_f, \, x_t)$, the underlying MDP $\mathcal{M}'_{\mathrm{r}}(x_f, \, x_t)$ becomes a Markov Chain (MC), of which of asymptotic behavior is fully determined. Thus, the transition probability $p'_{\mathrm{r},D}\,(x_f, \boldsymbol{\mu}'_{\mathrm{r}}(x_f, \, x_t), x_\ell)$ from the source state x_f to *any other* sink state $x_\ell \in X'_{\mathrm{r}}$ is given by the final distribution over the sink states. Note that if the LP above does not have a solution for the pair $(x_f, \, x_t)$, it means x_t cannot be reached from x_f. Consequently, the transition probability $p'_{\mathrm{r},D}\,(x_f, \boldsymbol{\mu}'_{\mathrm{r}}(x_f, \, x_t), x_\ell) = 0$ for all $x_\ell \in X'_{\mathrm{r}}$. The detailed formulation of the LP and computation of $p'_{\mathrm{r},D}$ can be found in the supplementary material.

Via the above three steps, the labeled semi-MDP $\mathcal{M}'_{\mathrm{r}}$ in (20) can be constructed fully. Assume that the DRA $\mathcal{A}_{\mathrm{r}}$ has N

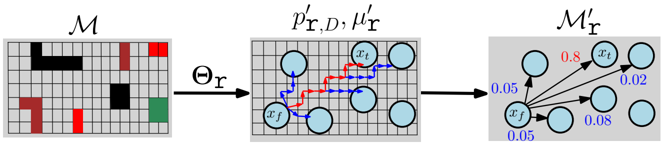

$\mathcal{M}$
$\Theta_{\mathtt{r}}$
$p'_{\mathtt{r},D}, \mu'_{\mathtt{r}}$
$\mathcal{M}'_{\mathtt{r}}$
x_t
x_f
0.8
0.05
0.02
0.08
0.05

Fig. 4. Illustration of feature-based abstraction as described in Sec. VI-A1. The associated motion policy $\boldsymbol{\mu}'_{\mathtt{r}}$ and the transition probability $p'_{\mathtt{r},D}$ should be computed for each pair of transition (x_f, x_t) in $\mathcal{M}'_{\mathtt{r}}$. Note that the ending states under the macro action (x_f, x_t) are non-deterministic, as shown in the middle and right figures.

unique labels, and within $\mathcal{M}$ there are at most M regions of interest associated with each label, then $\mathcal{M}'_{\mathtt{r}}$ has maximum $M \cdot N$ nodes and less than $M^2 \cdot N^2$ edges. For large systems with few number of interested regions, this can be significantly less than the size of $\mathcal{M}$.

2) Hierarchical Safe-Return Policy Synthesis: Given the labeled semi-MDP $\mathcal{M}'_{\mathtt{r}}$ from (20) and the DRA $\mathcal{A}_{\mathtt{r}}$, their product can be computed in the same way as in Def. 2:

$$\mathcal{P}'_{\mathtt{r}} \triangleq \mathcal{M}'_{\mathtt{r}} \times \mathcal{A}_{\mathtt{r}} = (S'_{\mathtt{r}},\, U'_{\mathtt{r}},\, E'_{\mathtt{r}},\, p'_{\mathtt{r},E},\, s'_{\mathtt{r},0},\, \mathrm{Acc}'_{\mathtt{r}}), \tag{21}$$

of which the detailed notations are similarly defined as in (9) and omitted here. Note that the transition probability $p'_{\mathtt{r},E}$ is computed based on $p'_{\mathtt{r},D}$. Given $\mathcal{P}'_{\mathtt{r}}$ above, the optimal safe-return policy $\boldsymbol{\pi}'_{\mathtt{r}}$ and the associated value function $v'^{\star}_{\mathtt{r}}$ can be computed in the same way as computing $\boldsymbol{\pi}_{\mathtt{r}}$ from $\mathcal{P}_{\mathtt{r}}$, which are described in Sec. V-D. Furthermore, since $\varphi_{\mathtt{r}}$ is given as sc-LTL formulas, the safe-return constraints are satisfied once the union set of AMECs $S'_{\mathtt{r},\Xi}$ above is reached.

B. Hierarchical Planning for Tasks

Similar to the return policy above, the outbound policy can also be synthesized in a hierarchical way. Namely, a feature-based abstraction model as semi-MDPs for the task specification is firstly constructed, which is safety-ensured as it directly incorporates the safe-return constraints. Second, a hierarchical outbound policy is synthesized given the product automaton between this abstraction model and the task automaton.

1) Safety-ensured Semi-MDPs for Tasks: Different from $\mathcal{M}'_{\mathtt{r}}$ in (20), the abstraction model for tasks as semi-MDPs should incorporate the safe-return constraints. First, the set of effective features within $\mathcal{A}_{\circ}$ is computed similarly as (19), denoted by $\Theta_{\circ}$. Then, the associated abstraction model as semi-MDPs is given by:

$$\mathcal{M}'_{\circ} \triangleq \left(X'_{\circ},\, U'_{\circ},\, p'_{\circ,D},\, AP,\, L'_{\circ},\, c'_{\circ,D}\right), \tag{22}$$

where the notations are defined analogously as in (20). However, the transition probability $p'_{\circ,D}$ and cost $c'_{\circ,D}$ are computed differently to incorporate the safe-return constraints.

Problem 3. Determine $p'_{\circ,D}$, $c'_{\circ,D}$ of $\mathcal{M}'_{\circ}$, given system $\mathcal{M}$ and the value function $v'^{\star}_{\mathtt{r}}$ of $\mathcal{P}'_{\mathtt{r}}$. ■

For *each* pair of states $(x_f,\, x_t) \in X'_{\circ} \times X'_{\circ}$, the associated macro action is symbolically denoted by $(x_f,\, x_t) \in U'_{\circ}$. Then, its transition probability $p'_{\circ,D}$ is computed in a similar procedure as $p'_{\mathtt{r},D}$ in Sec. VI-A, namely: (i) Construct the modified MDP $\mathcal{M}'_{\circ}(x_f,\, x_t)$ from $\mathcal{M}$ similar to $\mathcal{M}'_{\mathtt{r}}(x_f,\, x_t)$; (ii) Find the motion policy $\boldsymbol{\mu}'_{\circ}(x_f,\, x_t)$ as options that drives the system $\mathcal{M}'_{\circ}(x_f,\, x_t)$ from x_f to x_t with the *maximum* probability, while satisfying the safe-return constraint. Theorem 2 proves that the safe-return constraint can be re-formulated as the accumulated rewards w.r.t. the value function $v'^{\star}_{\mathtt{r}}$. Thus, the LP for computing $\boldsymbol{\mu}'_{\circ}(x_f,\, x_t)$ is revised by adding another constraint:

$$\mathbb{E}^{\boldsymbol{\mu}'_{\circ}}_{\mathcal{M}'_{\circ}(x_f,\, x_t)} \left\{ \sum_{t=0}^{\infty} v'^{\star}_{\mathtt{r}}(s^{\mathtt{r}}_t) \right\} \geqslant \chi_{\mathtt{r}}, \tag{23}$$

where $s^{\mathtt{r}}_t = \langle x_t, q^{\mathtt{r}}_0 \rangle$ is the product state of $\mathcal{P}'_{\mathtt{r}}$ derived from (21); $v'^{\star}_{\mathtt{r}}(s^{\mathtt{r}}_t)$ is the associated value function from (21); (iii) The transition probability $p'_{\circ,D}\left(x_f, \boldsymbol{\mu}'_{\circ}(x_f,\, x_t), x_\ell\right)$ is the convergent distribution of $\mathcal{M}'_{\circ}(x_f,\, x_t)$ under $\boldsymbol{\mu}'_{\circ}(x_f,\, x_t)$. Furthermore, the cost function $c'_{\circ,D}$ is determined by:

$$c'_{\circ,D}(x_f,\, x_t) = \mathbb{E}^{\boldsymbol{\mu}'_{\circ}}_{\mathcal{M}'_{\circ}(x_f,\, x_t)} \left\{ \sum_{t=0}^{\infty} c_D(x_t, u_t) \right\}, \tag{24}$$

as the expected cost of reaching x_t from x_f while following the motion policy $\boldsymbol{\mu}'_{\circ}(x_f,\, x_t)$. Note if the constrained LP for solving $\boldsymbol{\mu}'_{\circ}(x_f,\, x_t)$ *does not* have a solution, it means that x_t can not be reached from x_f while satisfying the safe-return constraints in (23). Then, the macro transition (x_f, x_t) is removed from $U'_{\circ}$ with no associated transition. The detailed formulation of the LP and derivation of $p'_{\circ,D}$, $c'_{\circ,D}$ can be found in the supplementary material.

Remark 4. The motion policy $\boldsymbol{\mu}'_{\circ}(x_f,\, x_t)$ above is an option that prioritizes the maximization of the probability of reaching x_f from x_t, instead of minimizing the expected cost $c'_{\circ,D}(\cdot)$ in (24). This could result in a loss of optimality regarding the original objective in (8). First of all, as also mentioned in our earlier work [1], there are often trade-offs when co-optimizing both terms. Second, the overall task satisfiability in (8) is not incorporated in this *local* semi-MDP, rather only in the product automaton in the sequel. Nonetheless, as shown later in the experiment, such loss is negligible, especially in contrast to the significant reduction in computation time. ■

Lemma 3. *The semi-MDP $\mathcal{M}'_{\circ}$ is safety-ensured. Namely, for each transition $(x_f, x_t) \in U'_{\circ}$, its associated motion policy $\boldsymbol{\mu}'_{\circ}(x_f, x_t)$ and the return policy $\boldsymbol{\mu}'_{\mathtt{r}}$ satisfy the safe-return constraints by* (7).

Proof. For each transition $(x_f, x_t) \in U'_{\circ}$, the safe-return constraints are enforced directly by (23) when synthesizing the motion policy $\boldsymbol{\mu}'_{\circ}(x_f, x_t)$. As shown in Theorem 2, the expected accumulative value function of all states encountered during the execution of $\boldsymbol{\mu}'_{\circ}(x_f, x_t)$ is equivalent to the safe-return probability in (7). Thus, during the execution of any transition within $\mathcal{M}'_{\circ}$, once requested, the robot can always return to the safe states with a probability larger than $\chi_{\mathtt{r}}$, by following the safe-return policy $\boldsymbol{\mu}'_{\mathtt{r}}$. □

2) Hierarchical Outbound Policy Synthesis: Given the safety-ensured semi-MDP $\mathcal{M}'_{\circ}$, its product with the DRA $\mathcal{A}_{\circ}$ can be computed via following Def. 3:

$$\mathcal{P}'_{\circ} \triangleq \mathcal{M}'_{\circ} \times \mathcal{A}_{\varphi_{\circ}}, \tag{25}$$

```
Algorithm 2: Hierarchical Planning Algorithm
  Input: $\mathcal{M}$, $(\varphi, \chi_{\mathtt{o}})$, $(\varphi_{\mathtt{r}}, \chi_{\mathtt{r}})$.
  Output: $((\mathcal{M}'_{\mathtt{r}}, \boldsymbol{\mu}'_{\mathtt{r}}), (\mathcal{P}'_{\mathtt{r}}, \boldsymbol{\pi}'_{\mathtt{r}}, v'^{\star}_{\mathtt{r}}))$,
          $((\mathcal{M}'_{\mathtt{o}}, \boldsymbol{\mu}'_{\mathtt{o}}), (\mathcal{P}'_{\mathtt{o}}, (\boldsymbol{\pi}'_{\mathtt{o,pre}}, \boldsymbol{\pi}'_{\mathtt{o,suf}})))$
  /* Safe-return policy, Sec. VI-A    */
1 Construct DRA $\mathcal{A}_{\mathtt{r}}$;
2 Build abstraction $\mathcal{M}'_{\mathtt{r}}$ and policy $\boldsymbol{\mu}'_{\mathtt{r}}$;
3 Compute product $\mathcal{P}'_{\mathtt{r}}$ in (21) ;
4 Synthesize return policy $\boldsymbol{\pi}'_{\mathtt{r}}$, and value function $v'^{\star}_{\mathtt{r}}$;
  /* Task policy, Sec. VI-B           */
5 Construct DRA $\mathcal{A}_{\mathtt{o}}$;
6 Build abstraction $\mathcal{M}'_{\mathtt{o}}$ and policy $\boldsymbol{\mu}'_{\mathtt{o}}$, given $v'^{\star}_{\mathtt{r}}$;
7 Compute product $\mathcal{P}'_{\mathtt{o}}$ in (25);
8 Synthesize outbound policy $(\boldsymbol{\pi}'_{\mathtt{o,pre}}, \boldsymbol{\pi}'_{\mathtt{o,suf}})$ ;
```

of which the detailed notations are similarly defined as in (9) and omitted here. Note that its cost function follows from (24). Now, the objective is to find the optimal outbound policy over $\mathcal{P}'_{\mathtt{o}}$ for the original planning problem.

Problem 4. Determine the policy $\boldsymbol{\pi}'_{\mathtt{o}}$ for $\mathcal{P}'_{\mathtt{o}}$ such that the long-term cost in (3) is minimized and the task satisfiability in (2) is lowered bounded. ■

Notice that different from $\mathcal{P}_{\mathtt{o}}$, the product $\mathcal{P}'_{\mathtt{o}}$ above *inherits* the safety property from the abstraction model $\mathcal{M}'_{\mathtt{o}}$ as proved in Lemma 3. Consequently, the safe-return constraints are not imposed as an additional requirement when synthesizing the outbound policy $\boldsymbol{\pi}'_{\mathtt{o}}$. Since task φ is given as general LTL formulas, $\boldsymbol{\pi}'_{\mathtt{o}}$ now consists of two parts: the prefix $\boldsymbol{\pi}'_{\mathtt{o,pre}}$ that is executed once, and the suffix $\boldsymbol{\pi}'_{\mathtt{o,suf}}$ that is executed infinite times. Without the additional safe-return constraints, the framework proposed in our earlier work [1] can be used directly to synthesize $\boldsymbol{\pi}'_{\mathtt{o}}$. For brevity, Alg. 1 in [1] is encapsulated as the following constrained optimization:

$$\min_{\{\boldsymbol{\pi}'_{\mathtt{o,pre}}, \boldsymbol{\pi}'_{\mathtt{o,suf}}\}} \quad \mathbf{Cost}_{\mathcal{P}'_{\mathtt{o}}}^{\boldsymbol{\pi}'_{\mathtt{o,suf}}} \tag{26a}$$

$$\text{s.t.} \quad \mathbf{Sat}_{\mathcal{P}'_{\mathtt{o}}}^{\boldsymbol{\pi}'_{\mathtt{o,pre}}} \geqslant \chi_{\mathtt{o}}, \tag{26b}$$

where the prefix policy $\boldsymbol{\pi}'_{\mathtt{o,pre}}$ ensures that the union of AMECs $S'_{\mathtt{o},\Xi}$ is reached with a probability larger than $\chi_{\mathtt{o}}$, while the suffix policy $\boldsymbol{\pi}'_{\mathtt{o,suf}}$ ensures that the system stays inside $S'_{\mathtt{o},\Xi}$ and the discounted cost in (8) is minimized. In the end, both policies are optimized together to find the best pair of initial state and accepting state. The detailed LP formulation and policy derivation are given in the supplementary material.

C. Algorithmic Summary and Online Execution

The complete hierarchical planning algorithm is summarized in Alg. 2 and illustrated in Fig. 3. Namely, it consists of two main parts: Lines $1-4$ to construct the semi-MDP $\mathcal{M}'_{\mathtt{r}}$ and product $\mathcal{P}'_{\mathtt{r}}$, and learn the safe-return policy $\boldsymbol{\pi}'_{\mathtt{r}}$ and the associated value function $v'^{\star}_{\mathtt{r}}$; Lines $5-8$ to construct the semi-MDP $\mathcal{M}'_{\mathtt{o}}$ and product $\mathcal{P}'_{\mathtt{o}}$, and learn the outbound policy $\boldsymbol{\pi}'_{\mathtt{o}}$. As shown in Fig. 5, the outbound policy is executed hierarchically as discussed in the sequel. The safe-return policy is activated only if the robot is requested to return.

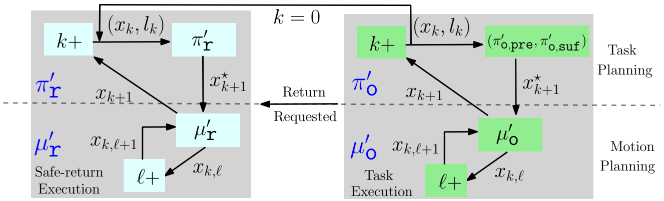


Fig. 5. Illustration of the online execution of the hierarchical outbound policy $(\boldsymbol{\pi}'_{\mathtt{o}}, \boldsymbol{\mu}'_{\mathtt{o}})$, and the hierarchical safe-return policy $(\boldsymbol{\pi}'_{\mathtt{r}}, \boldsymbol{\mu}'_{\mathtt{r}})$ if requested.

During online execution, the outbound policy is executed hierarchically as follows: Starting from the initial state $s'_{\mathtt{o},k}$ and $k=0$, the prefix policy $\boldsymbol{\pi}'_{\mathtt{o,pre}}$ is activated to determine the next action $(x_k, x^{\star}_{k+1}) \in U'_{\mathtt{o}}$. Once $x^{\star}_{k+1}$ is given, the option $\boldsymbol{\mu}'_{\mathtt{o,pre}}(x_k, x^{\star}_{k+1})$ is activated to determine the next motion $(x_k, x_{k,\ell}) \in U$, where $x_{k,\ell} \in X$ but $x_{k,\ell} \notin X'_{\mathtt{o}}$. The step ℓ is increased until one of the sink states in $X'_{\mathtt{o}}$ is reached, which is denoted by x_{k+1} and not necessarily $x^{\star}_{k+1}$ due to non-determinism. Given the actual next state x_{k+1}, the outbound policy $\boldsymbol{\pi}'_{\mathtt{o,pre}}$ is activated again to determine the next action $(x_{k+1}, x^{\star}_{k+2}) \in U'_{\mathtt{o}}$. This process repeats itself until the set $S'_{\mathtt{o},\Xi}$ of $\mathcal{P}'_{\mathtt{o}}$ is reached. Afterwards, the outbound policy switches to the suffix policy $\boldsymbol{\pi}'_{\mathtt{o,suf}}$, while the return policy $\boldsymbol{\pi}'_{\mathtt{r}}$ remains the same. Last but not least, whenever the robot is requested to return, the safe-return policy is executed hierarchically in an analogous way as above, which however terminates after the system reaches the set $S'_{\mathtt{r},\Xi}$.

Theorem 4. *The safe-return policy $\boldsymbol{\pi}'_{\mathtt{r}}$ and the outbound policy $\boldsymbol{\pi}'_{\mathtt{o}}$ fulfill the constraints in* (8) *as imposed by Problem 1, and minimizes the long-term cost in* (26a).

Proof. First, it is proven in Lemma 3 that the semi-MDP $\mathcal{M}'_{\mathtt{o}}$ is safety-ensured. In other words, during the execution of any transition within $\mathcal{M}'_{\mathtt{o}}$, once requested the robot can always return to the safe states with a probability larger than $\chi_{\mathtt{r}}$, by following the safe-return option $\boldsymbol{\mu}'_{\mathtt{r}}$. Thus, the outbound policy $\boldsymbol{\pi}'_{\mathtt{o}}$ derived by solving (26) for $\mathcal{P}'_{\mathtt{o}}$ always satisfies the safe-return constraints. Second, as proven in Theorem 6 of our earlier work [1], the optimization in (26) guarantees the task satisfiability by co-optimizing the prefix and suffix of the outbound policy $\boldsymbol{\pi}'_{\mathtt{o}}$. In particular, it shows that the prefix policy $\boldsymbol{\pi}'_{\mathtt{o,pre}}$ ensures that the union of AMECs $S'_{\mathtt{o},\Xi}$ is reached with a probability larger than $\chi_{\mathtt{o}}$, while the suffix policy $\boldsymbol{\pi}'_{\mathtt{o,suf}}$ ensures that the system stays inside $S'_{\mathtt{o},\Xi}$ and the long-term discounted cost is minimized. Lastly, as mentioned in Remark 4, the transition cost $c'_{\mathtt{o},D}$ within $\mathcal{M}'_{\mathtt{o}}$ might *not* be equivalent to the minimum cost for each transition. As a result, the discounted cost in (26a) is an approximation of the original cost in (8). The exact gap between them is not quantified in this work and remains part of our ongoing research. □

Remark 5. It is worth pointing out the algorithmic differences between the baseline method in Alg. 1 and the proposed hierarchical approach in Alg. 2. As shown in Fig. 1 and Fig. 3, the baseline method directly synthesizes first the return policy $\pi_{\mathtt{r}}$ and then the outbound policy $\pi_{\mathtt{o}}$ over the original system $\mathcal{M}$, while the proposed approach constructs first the symbolic and temporal abstractions $\mathcal{M}'_{\mathtt{r}}$, $\mathcal{M}'_{\mathtt{o}}$ as semi-MDPs and their associated polices as options. Theses semi-MDPs are then used as the respective system model for synthesizing the high-level task policy and return policy. This hierarchical approach allows planning at different spatial and temporal levels over different system models, i.e., instead of a common model as in the baseline method. ■

VII. Simulation and Experiment Study

In this section, we present two case studies to validate the proposed framework: the search-and-rescue mission in office environment after a disaster; and the planetary exploration mission similar to the DLR SpaceBot Camp [45]. All algorithms are implemented in Python3 and the main LP solver is the Google Linear Optimization Solver "*Glop*", see [46]. All computations are carried out on a laptop (3.06GHz Duo CPU and 12GB of RAM). Detailed robot model and workspace descriptions can be found in the supplementary material.

A. *Study One: Search-and-Rescue Mission*

For the first numerical study, we consider the deployment of a UGV into an office environment for a search-and-rescue mission after a disaster.

1) Workspace and Task Description: As shown in Fig. 6, a search-and-rescue UGV of size $1\text{m} \times 0.5\text{m}$ is deployed to explore one floor of an office building, approximately of size $170\text{m} \times 80\text{m}$. The system model $\mathcal{M}$ is estimated by the floor plan and the robot mobility. More specifically, the states are given by a full discretization of the floor plan in 2D (with different granularity as described later); transition probabilities and costs are estimated by a simulated robot navigation model, e.g., to the adjacent regions via 4-way movements with orientation constraints. Note that the robot can drift side-ways while moving or overshoot while rotating, yielding a probabilistic model; and the states are labeled by the corresponding rooms, while features such as victims or hazards are estimated manually. For instance, the probability of having label human ($\texttt{hm}$) in offices is 0.9, whereas 0.1 in storage rooms ($\texttt{st}$). We refer the readers to the supplementary material for detailed construction of $\mathcal{M}$. It is worth noting that given the workspace blueprint (including walls, doors and stairs) and the robot motion model, the model $\mathcal{M}$ can be constructed algorithmically without manual inputs. Moreover, there are two potential risks during the mission: first, the robot may be trapped in certain rooms due to one-way doors or some exits being blocked by debris; second, the robot may descend any stairs but not ascend too steep stairs.

More specifically, the search-and-rescue tasks include: (i) turn off switches at the maintenance room ($\texttt{mt}$); (ii) visit storage rooms ($\texttt{st}$) to check for fire hazards and gas leakage; (iii) search office rooms ($\texttt{of}$) for injured victims ($\texttt{hm}$) and bring them to the closest medical station ($\texttt{md}$). The locations of these labels are shown in Fig. 6. They are specified as the following LTL formulas:

$$\begin{aligned}\varphi = \;& \Diamond \texttt{st} \wedge (\Diamond \bigvee\nolimits_{\ell\in L} \texttt{mt}_\ell) \\ & \wedge (\Box\Diamond(\bigvee\nolimits_{i\in I} \texttt{of}_i \wedge \texttt{hm})) \wedge (\Box(\texttt{hm} \rightarrow (\neg\texttt{hm})\mathsf{U}\texttt{md})),\end{aligned} \tag{27}$$

where I and L are the sets of offices and maintenance rooms, respectively; the actions to perform at respective regions are omitted and refer to [37] for methods to combine motion and action planning. To encourage visits over different office rooms and mimic the realistic scenario of victims being relocated from offices to medical stations, the label $\texttt{hm}$ is removed from an office after it has been visited certain number of times. For a more precise modeling with robot actions and dynamic environments, readers are referred to our earlier work [8], [37]. On the other hand, the safe-return constraints require the robot to be able to return to the base station ($\texttt{bs}$) from any medical station, via one designated exit ($\texttt{ex}$), i.e.,

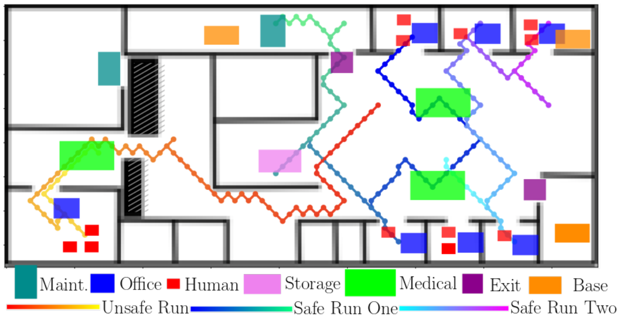


Fig. 6. Workspace model and examples of different runs under the outbound policy $\boldsymbol{\mu}'_{\circ}$ and an unsafe policy (in red) without the safe constrains.

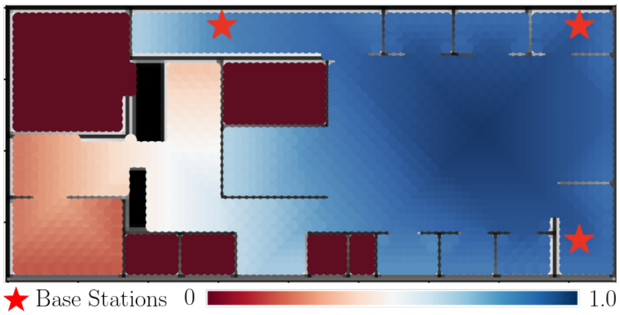


Fig. 7. Heatmap of the value function $v'^{\star}_{\mathrm{r}}$ associated with the safe-return constraints in (28). The base station is marked by the red star.

$$\varphi_{\mathrm{r}} = \Diamond(\texttt{md} \wedge \Diamond\texttt{ex}) \wedge \Diamond\Box(\bigvee\nolimits_{j\in J} \texttt{bs}_j), \tag{28}$$

where J is the set of three base stations as shown in Fig. 6. For probabilistic requirements, the satisfiability bound $\chi_{\circ}$ is set to 0.8, while the safety bound χ_{r} is set to 0.9.

2) Simulation Results: In the section, we mainly present the results for the hierarchical planning algorithm in Alg. 2. Comparison with other baselines are given in the sequel. The discretization step for constructing the underlying MDP $\mathcal{M}$ is set to 2m here, which has 1100 states and 10296 edges.

First of all, the DRA $\mathcal{A}_{\mathrm{r}}$ associated with φ_{r} in (28) is quite small with only 7 states, 14 edges and 1 accepting pair. Even though the original $\mathcal{M}$ is relatively large in size, the semi-MDP $\mathcal{M}'_{\mathrm{r}}$ in (20) contains 15 states and 210 edges. In particular, the synthesis of the low-level motion policy $\boldsymbol{\mu}'_{\circ}$ takes in average 3ms for each edge within $\mathcal{M}'_{\mathrm{r}}$. Afterwards, the product $\mathcal{P}'_{\mathrm{r}}$ in (21) is computed in 1.5s, which has 511 states and 36288 edges. Given $\mathcal{P}'_{\mathrm{r}}$, the optimal return policy $\boldsymbol{\pi}'_{\mathrm{r}}$, and its value function $v'^{\star}_{\mathrm{r}}$ are computed in 2.3s. A visualization of $v'^{\star}_{\mathrm{r}}$ is given in Fig. 7, which matches our intuition well

TABLE I
SCALABILITY RESULTS OF THREE METHODS FOR THE CASE STUDY ONE.

$\mathcal{M}$	Method[1]	$\widehat{\mathcal{M}}$, $\mathcal{M}$, $\mathcal{M}'$		$\mathcal{P}_{\mathtt{r}}$, $\mathcal{P}'_{\mathtt{r}}$		$\widehat{\mathcal{P}}$, $\mathcal{P}_{\mathtt{o}}$, $\mathcal{P}'_{\mathtt{o}}$		$\widehat{\boldsymbol{\pi}}$, $(\boldsymbol{\pi}_{\mathtt{r}}, \boldsymbol{\pi}_{\mathtt{o}})$, $(\boldsymbol{\pi}'_{\mathtt{r}}, \boldsymbol{\pi}'_{\mathtt{o}})$		
		Size[2]	Time[4][s]	Size[2]	Time [s]	Size[2]	Time [s]	Size[3]	Time [s]	Safety
(3e2, 2e3)[2]	ALT	(6e2, 6e3)	0.2	—/—	—/—	(2e5, 2e7)	3e2	1e6	29.6	0.95
	BSL	(3e2, 2e3)	**0.01**	(2e3, 3e4)	0.2	(1e4, 3e5)	21.8	6e3	9.6	0.95
	HIR	(3e2, 2e3)	0.3	(5e2, 3e4)	**0.4**	(2e3, 2e5)	**10.1**	8e3	**6.2**	0.94
(5e3, 4e4)	ALT	(1e4, 2e5)	0.5	—/—	—/—	(1e6, 2e7)	4e3	8e7	2e4	0.9
	BSL	(5e3, 4e4)	**0.3**	(4e4, 3e5)	3.6	(2e5, 2e6)	165	4e5	3e3	0.9
	HIR	(3e2, 4e3)	8.3	(5e2, 4e4)	**2.3**	(4e3, 2e5)	**18.7**	5e4	**7.1**	0.91
(2e4, 2e5)	ALT	(4e4, 6e5)	3.4	—/—	—/—	(9e6, 5e7)	1e4	N/A[4]	N/A	N/A
	BSL	(2e4, 2e5)	**1.4**	(1e5, 2e6)	15.8	(8e5, 7e6)	7e2	2e6	N/A	N/A
	HIR	(5e2, 5e3)	15.1	(5e2, 5e4)	**5.2**	(4e3, 3e5)	**22.8**	5e4	**12.8**	0.9
(8e4, 7e5)	ALT	(3e5, 2e6)	7.8	—/—	—/—	N/A[4]	N/A	N/A	N/A	N/A
	BSL	(8e4, 7e5)	**3.7**	(6e5, 7e6)	61.2	N/A	N/A	N/A	N/A	N/A
	HIR	(6e2, 7e3)	21.2	(6e2, 1e5)	**10.2**	(5e3, 4e5)	**28.2**	5e4	**17.6**	0.9

[1] ALT: The alternative approach described in Sec. V-E and PRISM 4.6 [24], which does not compute $\mathcal{P}_{\mathtt{r}}$ directly. BSL: The baseline solution in Alg. 1. HIR: The hierarchical algorithm in Alg. 2.
[2] Sizes of the MDP models are measured by the number of nodes and edges. $a\text{e}b \triangleq a \times 10^b$. Note that $1\text{e}3\text{s} \approx 16\text{min}$
[3] Sizes of the polices are measured by the number of LP variables.
[4] "N/A" indicates either insufficient memory or computation time longer than 5 hours.

that states that can *not* reach the base station have low values. Specifically, due to the narrow passage between debris, the left side of the office has relatively low value compared with the right side, while the areas that are completely inaccessible have value close to zero.

Furthermore, the safety-ensured semi-MDP $\mathcal{M}'_{\mathtt{o}}$ is computed. The DRA $\mathcal{A}_{\mathtt{o}}$ associated with task φ in (27) has 39 states, 255 edges and 1 accepting pair, which is much larger than $\mathcal{A}_{\mathtt{r}}$ earlier. It took 0.3s to construct $\mathcal{M}'_{\mathtt{o}}$ and the resulting product $\mathcal{P}'_{\mathtt{o}}$. At last, its associated outbound policy $(\boldsymbol{\pi}'_{\mathtt{o,pre}}, \boldsymbol{\pi}'_{\mathtt{o,suf}})$ is computed in 5.1s an 1.3s, respectively. The complete computation time is listed in the second row of Table I. Given the outbound policy, Fig. 6 show several runs of the online execution. It can be seen that the resulting trajectories search many different office areas and rescue the victims inside, while avoiding such areas where the value function is low and thus not safe to return to the base station. To show how the safe-return constraints and task constraints affect the trajectories, the above procedure is repeated with different values of $\chi_{\mathtt{r}}$, $\chi_{\mathtt{o}}$. More specifically, by setting these lower bounds to one of the values in $\{0.0, 0.5, 0.9\}$, the resulting trajectories and the associated costs are summarized in Table II, as partially shown in Fig. 6. Note that the return event could be requested anytime after the system starts. It can be seen that when $\chi_{\mathtt{o}}$ is small, the effectiveness of adding the safe-return constraints is not apparent as the robot simply stays around the initial state. However, when $\chi_{\mathtt{o}}$ is increased to improve task satisfiability and $\chi_{\mathtt{r}}$ increased to enforce the safety constraints, the resulting trajectories are more consistent and the robot is less likely to be trapped inside unsafe states. When $\chi_{\mathtt{r}} = 0.0$, the average cost of the resulting trajectories is much higher with more variance, due to more trajectories where the robot is trapped at an early stage. Nonetheless, as shown in the last row of Table II, simply increasing $\chi_{\mathtt{o}}$ to 0.9 can *not* achieve the same results. Namely, when $\chi_{\mathtt{o}}$ is set too high, the constrained optimization in (8) becomes infeasible and thus no solutions exist. In that case, a relaxed and maximum-satisfying policy can be synthesized as proposed in our earlier work [1], which is outside the scope of this paper. One example trajectory when $\chi_{\mathtt{r}} = 0.0$ is shown in Fig. 6, which indicates that the policy can not distinguish the areas with different value functions, thus more prone to being trapped during execution.

TABLE II
TRAJECTORY COST UNDER DIFFERENT $\chi_{\mathtt{o}}$ AND $\chi_{\mathtt{r}}$.

Traj. Cost[1]	$\chi_{\mathtt{r}} = 0.0$	0.5	0.9
$\chi_{\mathtt{o}} = 0.0$	3.0 ± 1.8	4.3 ± 1.5	5.8 ± 1.2
0.5	56.4 ± 21.2	48.8 ± 12.3	42.3 ± 5.7
0.9	N/A[2]	N/A	N/A

[1] Mean cost with standard deviation evaluated over 100 simulated runs of the proposed method with different $\chi_{\mathtt{o}}, \chi_{\mathtt{r}}$. If the robot is trapped, the cost is computed as the maximum action cost multiplied by the horizon.
[2] "N/A" indicates that no solutions can be found.

B. *Study Two: Planetary Exploration Mission*

For the second numerical study, we consider a planetary exploration mission similar to the DLR SpaceBot Camp [45]. The rover-like mobile manipulator needs to navigate to different areas to look for objects of interests, assemble and store them, while maintaining its battery level by charging often.

1) Workspace and Task Description: As shown in Fig. 8, a rover of size $1m \times 1m$ is deployed to a planetary-like environment of size $90m \times 90m$ with rough terrains. Its dynamic model within the environment is similar to the UGV described in Sec. VII-A1. States are marked by objects of interests with different probabilities. The environment consists of different types of soil surfaces that may cause slip and even trapping.

TABLE III
SCALABILITY RESULTS OF THREE METHODS FOR THE CASE STUDY TWO.

$\mathcal{M}$	Method[1]	$\widehat{\mathcal{M}}$, $\mathcal{M}$, $\mathcal{M}'$		$\mathcal{P}_{\mathtt{r}}$, $\mathcal{P}'_{\mathtt{r}}$		$\widehat{\mathcal{P}}$, $\mathcal{P}_{\mathtt{o}}$, $\mathcal{P}'_{\mathtt{o}}$		$\widehat{\boldsymbol{\pi}}$, $(\boldsymbol{\pi}_{\mathtt{r}}, \boldsymbol{\pi}_{\mathtt{o}})$, $(\boldsymbol{\pi}'_{\mathtt{r}}, \boldsymbol{\pi}'_{\mathtt{o}})$		
		Size	Time [s]	Size	Time [s]	Size	Time [s]	Size	Time [s]	Safety
(9e2, 8e3)	ALT	(2e3, 2e4)	0.1	—/—	—/—	(7e4, 1e6)	24.1	7e3	9.6	0.95
	BSL	(9e2, 8e3)	**0.03**	(6e3, 7e4)	0.6	(3e4, 2e5)	6.7	2e4	6.7	0.93
	HIR	(1e2, 2e3)	0.2	(1e2, 3e3)	**1.9**	(2e3, 1e5)	**3.5**	1e4	**2.5**	0.92
(2e4, 2e5)	ALT	(4e4, 6e5)	3.3	—/—	—/—	(8e4, 1e7)	2e3	N/A	N/A	N/A
	BSL	(2e4, 2e5)	**0.6**	(1e5, 1e6)	10.8	(4e5, 4e6)	145	3e5	2e3	0.9
	HIR	(2e2, 1e4)	8.3	(4e2, 4e4)	**2.9**	(2e3, 1e5)	**8.7**	2e4	**4.1**	0.9
(6e4, 7e5)	ALT	(2e4, 2e6)	10.4	—/—	—/—	N/A	N/A	N/A	N/A	N/A
	BSL	(6e4, 7e5)	**2.4**	(4e5, 5e6)	50.4	(2e6, 2e7)	4e2	N/A	N/A	N/A
	HIR	(3e2, 2e4)	19.1	(5e2, 4e4)	**3.2**	(2e3, 2e5)	**10.1**	2e4	**6.3**	0.9
(3e5, 3e6)	ALT	(6e5, 1e7)	1e2	—/—	—/—	N/A	N/A	N/A	N/A	N/A
	BSL	(3e5, 3e6)	**18.7**	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	HIR	(4e2, 4e4)	25.1	(5e2, 5e4)	**8.4**	(2e3, 3e5)	**18.2**	3e4	**10.6**	0.9

[1] Legends are the same as in Table I.

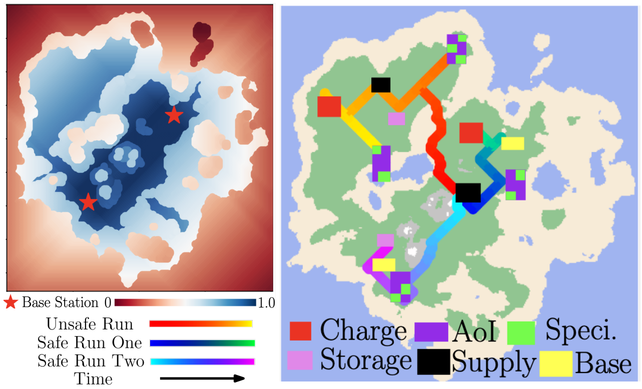


Fig. 8. **Left**: heatmap of the value function $v_{\mathtt{r}}'^{\star}$ associated with the safe-return constraints in (30). The base station is marked by the red star; **Right**: examples of different runs under the outbound policy $\boldsymbol{\mu}'_{\mathtt{o}}$ and an unsafe policy (in red) without the safe constrains.

Furthermore, there are also hills and valleys that might be too steep to descend and ascend. Once trapped, it may not be able to return safely to its base station. The system model is initialized by a depth image with a desired discretization level and manual estimation of the labels. Namely, given this depth image and the robot motion model above, the system model $\mathcal{M}$ can be constructed algorithmically, by checking the relative depth between neighboring cells.

More specifically, the exploratory mission include: (i) navigate to several potential areas of interest ($\mathtt{ar}_i$) and scope specimens ($\mathtt{ss}$); (ii) navigate to a supply area ($\mathtt{sp}$), grasp containers and put specimens in the containers; (iii) navigate to a storage area ($\mathtt{st}$) and put the containers there; finally (iv) charge at the charging areas ($\mathtt{ch}$) whenever battery is low. These tasks can be specified as the following LTL formulas:

$$\begin{aligned} \varphi = (\Box\Diamond\mathtt{ch}) \wedge \Big(&\Box\Diamond(\bigvee\nolimits_{i\in I} \mathtt{ar}_i \wedge \mathtt{ss}) \\ &\wedge \Box(\mathtt{ss} \rightarrow (\neg\mathtt{ss})\,\mathsf{U}\,\Diamond(\mathtt{sp} \wedge \Diamond\mathtt{st}))\Big), \end{aligned} \tag{29}$$

where I is the set of potential areas with specimens. Similar to the previous case, to encourage visits over different areas of interest, the label $\mathtt{ss}$ is removed from an area after it has been visited a certain number of times. The action model and the method to dynamically update the environment model are omitted and refer to our earlier work [8], [37]. On the other hand, the safe-return constraints require the rover to be able to return to the base station ($\mathtt{bs}$) after charging ($\mathtt{ch}$) and visiting the two regions, i.e.,

$$\varphi_{\mathtt{r}} = \Diamond(\mathtt{ch} \wedge \Diamond(\bigvee\nolimits_{i\in I} \mathtt{ar}_i)) \wedge \Diamond\Box(\bigvee\nolimits_{j\in J} \mathtt{bs}_j). \tag{30}$$

where J is the set of two base stations as shown in Fig. 8. Due to high uncertainties in the model, the satisfiability bound $\chi_{\mathtt{o}}$ is set to 0.6, while the safety bound $\chi_{\mathtt{r}}$ is set to 0.9.

2) Simulation Results: The results by following Alg. 2 are first presented here. By setting the discretization step to 3m, the underlying MDP $\mathcal{M}$ has 900 states and 8416 edges. Note that this model is highly *non-ergodic*, meaning that many states can be reached *from* the base station, but can not reach the base station. For instance, as shown in Fig. 8, some valleys can be reached easily by descending, but impossible to ascend back. First, given the DRA $\mathcal{A}_{\mathtt{r}}$ associated with $\varphi_{\mathtt{r}}$ in (30), the semi-MDP $\mathcal{M}'_{\mathtt{r}}$ and its motion policy $\boldsymbol{\mu}'_{\mathtt{r}}$ are constructed in 1.7s. The resulting product $\mathcal{P}'_{\mathtt{r}}$ has 119 nodes and 2628 edges, of which the optimal return policy $\boldsymbol{\pi}'_{\mathtt{r}}$, and the value function $v_{\mathtt{r}}'^{\star}$ are computed in 1.9s. The distribution of $v_{\mathtt{r}}'^{\star}$ is shown in Fig. 8. It can be noticed that the probability of returning to the base station decreases each time a valley or a rough terrain is crossed.

Second, the task DRA $\mathcal{A}_{\mathtt{o}}$ has 26 states, 190 edges and 1 accepting pair, which is slightly simpler than the search-and-rescue task. The semi-MDP $\mathcal{M}'_{\mathtt{o}}$ and its motion policy $\boldsymbol{\mu}'_{\mathtt{o}}$ are computed in 1.3s with 37 states and 1332 edges. Then, their product $\mathcal{P}'_{\mathtt{o}}$ is constructed in 3.5s, with 1898 states and 134784 edges. At last, its associated outbound policy $\boldsymbol{\pi}'_{\mathtt{o}}$ is computed in 2.5s via a LP of 10^4 variables. The detailed model size and computation time are reported in the second row of Table III. Fig. 8 shows several trajectories by following the hierarchical task policy during online execution. It can be seen that they remain mostly within central plain area, where

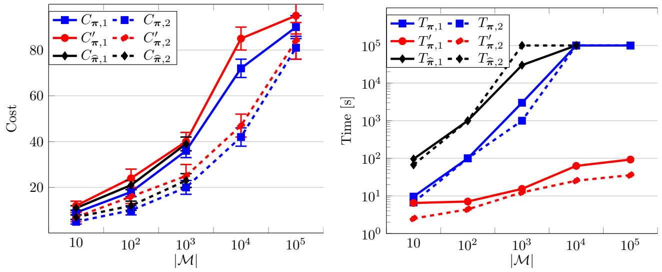


Fig. 9. **Left**: Cost distribution of 100 runs under the polices generated by the baseline solution ($C_{\boldsymbol{\pi},1},\ C_{\boldsymbol{\pi},2}$),the hierarchical algorithm ($C'_{\boldsymbol{\pi},1},\ C'_{\boldsymbol{\pi},2}$) and the alternative approach ($C_{\widehat{\boldsymbol{\pi}},1},\, C'_{\widehat{\boldsymbol{\pi}},2}$). **Right**: Evolution of solution time for both case studies w.r.t. the system size $|\mathcal{M}|$: $(T_{\boldsymbol{\pi},1}, T_{\boldsymbol{\pi},2})$ for the baseline solution, $(T'_{\boldsymbol{\pi},1}, T'_{\boldsymbol{\pi},2})$ for the hierarchical algorithm, and $(T_{\widehat{\boldsymbol{\pi}},1}, T_{\widehat{\boldsymbol{\pi}},2})$ for the alternative approach.

the value function is high. In contrast, the above procedure is repeated without the safe-return constraints. One such run is also shown in Fig. 8, which instead crosses the connecting valleys to reach the upper-left region where the value function is low, and thus more likely to get trapped and not be able to return the base station.

C. Comparison with Baselines

To further validate the computational gain and cost optimality of the proposed hierarchical approach (HIR), we compare it with the baseline method (BSL) described in Alg. 1 and the alternative method (ALT) discussed in Sec. V-E for both case studies. Regarding the ALT method, PRISM 4.6 is used [24] with the “multi-objective solution method” option set to “LP”, see [22] to compute the prefix policies, while a similar algorithm as proposed in our earlier work [1] is followed to compute the suffix policy within the AMECs. More specifically, we decrease the discretization size of the workspace such that the size of $\mathcal{M}$ increases gradually.

For the case study one, Table I summarizes the resulting size of $\mathcal{M}_{\texttt{r}}$, $\mathcal{M}_{\texttt{o}}$, $\mathcal{P}_{\texttt{r}}$ and $\mathcal{P}_{\texttt{o}}$ for Alg. 1; the size of $\widehat{\mathcal{M}}$, $\widehat{\mathcal{P}}$ for the alternative method; and the size of $\mathcal{M}'_{\texttt{r}}$, $\mathcal{M}'_{\texttt{o}}$, $\mathcal{P}'_{\texttt{r}}$ and $\mathcal{P}'_{\texttt{o}}$ for Alg. 2. Note that since the alternative method computes directly the extended model $\widehat{\mathcal{M}}$ and its product $\widehat{\mathcal{P}}$, the computation of product $\mathcal{P}_{\texttt{r}}$ does not apply. The computation time for these models and the associated polices are also reported. Both the alternative method and the baseline method take less pre-processing time to compute $\widehat{\mathcal{M}}$ and $\mathcal{M}$, compared with the abstracted model $\mathcal{M}'$. However, as the system size increases, it is clear that both methods quickly become intractable and even formulating the underlying LPs over $\mathcal{P}_{\texttt{o}}$ takes hours. For MDPs with more than 0.7 million edges, the alternative method fails to generate the product $\widehat{\mathcal{P}}$ before even computing the overall policy $\widehat{\boldsymbol{\pi}}$. This is mainly due to the fact that $\widehat{\mathcal{P}} = \widehat{\mathcal{M}} \times \mathcal{A}_{\texttt{r}} \times \mathcal{A}_{\texttt{o}}$, which leads to a drastic blow-up of the model size, compared with $\mathcal{P}_{\texttt{o}}$ or $\mathcal{P}'_{\texttt{o}}$ in the other two methods. Although the baseline method can compute the outbound and safety products $\mathcal{P}_{\texttt{r}}$, $\mathcal{P}_{\texttt{o}}$, neither the return policy nor the outbound policy can be computed within reasonable amount of time, i.e., 5 hours. In contrast, the proposed method not only can solve the same set of problems with at least 10 times less time, but also problems with much higher complexity where the baseline method failed. It is interesting to notice that the computation of the abstraction model $\mathcal{M}'_{\texttt{r}}$, $\mathcal{M}'_{\texttt{o}}$ and the associated motion policy took most of the time, while the size of the product model $\mathcal{P}'_{\texttt{r}}$ and $\mathcal{P}'_{\texttt{o}}$ remains relatively constant given a fixed task specification.

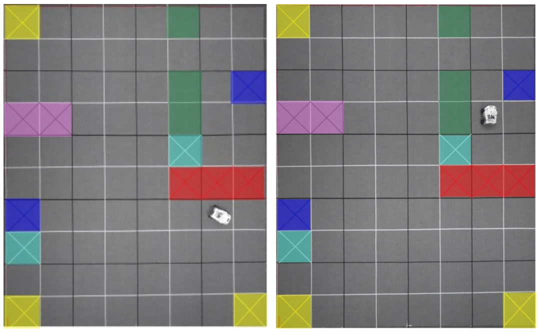


Fig. 10. Snapshots of the task execution. **Left**: the robot is transporting victims from one `bs` to one `md`. **Right**: without the safe-return constraint, the robot reaches the other `md` after crossing the narrow passage between `str`.

Similar analyses are performed for the case study two. Table III reports similar results. Namely, both the alternative method and the baseline method fail to generate a solution for systems with a large number of states and edges, while the proposed hierarchical approach is close to one order of magnitude more efficient. For the extreme case where $\mathcal{M}$ has 3 million edges, the alternative approach or the baseline solution can neither construct the products nor the linear programs given the memory and time limits, while the proposed approach can obtain the task and motion policy within 37.2s. Fig. 9 plots the solution time with respect to the system size for both cases.

Last but not least, the cost optimality of the resulting polices from all three methods are evaluated by 100 Monte Carlo simulations, for both case studies. As shown in Fig. 9, for system sizes where the baseline method can find the optimal solution, the proposed approach has a close-to-optimal cost, while the alternative method can only solve quite limited scenarios. Even for the case where $\mathcal{M}$ has around 10^5 edges, the proposed approach is within the range of 5% extra cost.

D. Hardware Experiment

The hardware experiment is carried out on an autonomous ground vehicle, within an artificial office environment, as shown in Fig. 10. The workspace has a size of 5m × 4m. Its state is monitored by a motion capture system and the communication between the state estimation, planning and control is handled by Robot Operating System (ROS).

1) Robot and Workspace Description: As shown in Fig. 10, the workspace is divided into 10×8 cells. The features and task description follow a similar setting to the search and rescue mission described in Sec. VII-A1. Similar to (27), the task is to transport victims in base stations ($\texttt{bs}_1$, $\texttt{bs}_2$) to any medical stations `md`, while avoiding the obstacles. On the other hand, the safe-return constraint is similar to (28), which requires the vehicle to return to the first base $\texttt{bs}_3$, without being trapped in the stairs. The number of offices and obstacles is smaller than the simulated case, of which the associated probabilities are shown in Fig. 11. Note that the satisfiability bound $\chi_{\texttt{o}}$ is set to 0.75 and the safe-return bound $\chi_{\texttt{r}}$ is set to 0.95. The vehicle has a navigation controller to move from any cell to

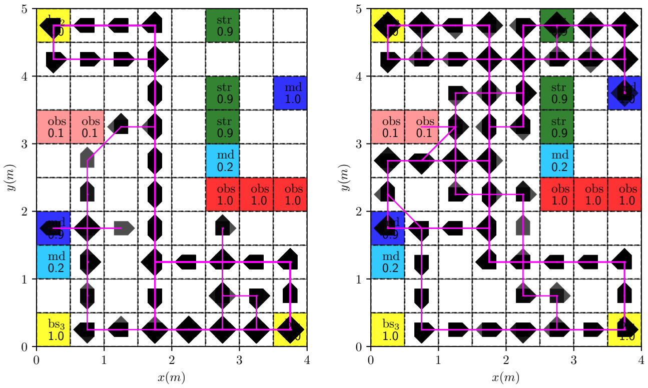


Fig. 11. Examples of robot trajectories in the plan suffix with (**left**) and without (**right**) the safe-return constraints via the proposed approach.

adjacent cell in a "turn-and-forward" fashion. It results in a similar non-determinism, i.e., it drifts side-ways while moving and overshoot while rotating, but with a smaller uncertainty compared to the simulated cases. Then, the complete model $\mathcal{M}$ is constructed automatically by composing this motion model with the labeled workspace model.

2) *Results:* In this part, we first report the results obtained by the baseline method in comparison with the case where no safe-return constraint is imposed. Then we show that the proposed hierarchical algorithm can generate the same safe-policy but with only a fraction of the planning time.

First, via the baseline method, the product $\mathcal{P}_\circ$ has 9880 states and 87048 edges, while the $\mathcal{P}_\mathrm{r}$ has 1900 states and 16740 edges. It takes 30.8s and 0.1s to compute the policy $\boldsymbol{\pi}_\mathrm{r}$ and $\boldsymbol{\pi}_\circ$, respectively. One of the resulting trajectories is shown in Fig. 11, which avoids the regions which can only be accessed via stairs. In comparison, when no such safe-return requirements as in (28) are imposed, the same product automaton $\mathcal{P}_\circ$ can be used and one resulting trajectory is also shown in Fig. 11. It can be seen that the robot reaches the medical station `md` via the narrow passage is trapped when exiting one region. Moreover, the proposed hierarchical algorithm is applied to the same problem. It took around 2.1s to construct the semi-MPDs $\mathcal{M}'_\circ$ and 0.5s for $\mathcal{M}'_\mathrm{r}$. Afterwards, the high-level task policy $\boldsymbol{\pi}'_\circ$ is synthesized in 7.3s, while the safe policy and the value function $v'^\star$ is computed in 2.6s. The total planning time is around half of the baseline solution. However, if the workspace is expanded by *four* times the size, the baseline solution takes around 97.5s while the hierarchical solution generates the optimal polices in merely 10.7s, i.e., close to one order of magnitude reduction in planning time, which is similar to the trend observed in the simulated cases.

VIII. Summary and Future Work

This work has proposed a hierarchical motion planning algorithm for mobile robots operating within uncertain environments. The proposed algorithm has taken into account not only high-level tasks but also safe-return constrains, both of which are specified as LTL formulas. It has been shown that the hierarchical planning algorithm significantly reduces the computation time compared with baseline solution, while maintaining a close-to-optimal performance. Future work includes online adaptation of the hierarchical policies.

References

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

[2] C. Belta, B. Yordanov, and E. A. Gol, *Formal Methods for Discrete-Time Dynamical Systems*. Springer, 2017, vol. 89.

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

[4] M. Hasanbeig, Y. Kantaros, A. Abate, D. Kroening, G. J. Pappas, and I. Lee, "Reinforcement learning for temporal logic control synthesis with probabilistic satisfaction guarantees," in *IEEE Conference on Decision and Control (CDC)*, 2019, pp. 5338–5343.

[5] A. K. Bozkurt, Y. Wang, M. M. Zavlanos, and M. Pajic, "Control synthesis from linear temporal logic specifications using model-free reinforcement learning," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 10 349–10 355.

[6] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[7] L. Laurenti, M. Lahijanian, A. Abate, L. Cardelli, and M. Kwiatkowska, "Formal and efficient synthesis for continuous-time linear stochastic hybrid processes," *IEEE Transactions on Automatic Control*, vol. 66, no. 1, pp. 17–32, 2020.

[8] M. Guo and D. V. Dimarogonas, "Multi-agent plan reconfiguration under local LTL specifications," *The International Journal of Robotics Research*, vol. 34, no. 2, pp. 218–235, 2015.

[9] P. Schillinger, M. Bürger, and D. V. Dimarogonas, "Simultaneous task allocation and planning for temporal logic goals in heterogeneous multi-robot systems," *The International Journal of Robotics Research*, vol. 37, no. 7, pp. 818–838, 2018.

[10] T. M. Moldovan and P. Abbeel, "Safe exploration in markov decision processes," in *29th International Coference on Machine Learning*. ACM, 2012, pp. 1451–1458.

[11] M. Turchetta, F. Berkenkamp, and A. Krause, "Safe exploration in finite markov decision processes with gaussian processes," *Advances in Neural Information Processing Systems*, vol. 29, 2016.

[12] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[13] R. Bellman, "Dynamic programming," *Science*, vol. 153, no. 3731, pp. 34–37, 1966.

[14] D. P. Bertsekas and J. N. Tsitsiklis, "Neuro-dynamic programming: an overview," in *IEEE Conference on Decision and Control (CDC)*, vol. 1, 1995, pp. 560–564.

[15] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT press, 2018.

[16] M. Lahijanian, S. B. Andersson, and C. Belta, "Formal verification and synthesis for discrete-time stochastic systems," *IEEE Transactions on Automatic Control*, vol. 60, no. 8, pp. 2031–2045, 2015.

[17] X. Ding, S. L. Smith, C. Belta, and D. Rus, "Optimal control of markov decision processes with linear temporal logic constraints," *IEEE Transactions on Automatic Control*, vol. 59, no. 5, pp. 1244–1257, 2014.

[18] V. Forejt, M. Kwiatkowska, G. Norman, D. Parker, and H. Qu, "Quantitative multi-objective verification for probabilistic systems," in *Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2011, pp. 112–127.

[19] J. Tumova and D. V. Dimarogonas, "Multi-agent planning under local LTL specifications and event-based synchronization," *Automatica*, vol. 70, pp. 239–248, 2016.

[20] Y. Kantaros, B. V. Johnson, S. Chowdhury, D. J. Cappelleri, and M. M. Zavlanos, "Control of magnetic microrobot teams for temporal micromanipulation tasks," *IEEE Transactions on Robotics*, vol. 34, no. 6, pp. 1472–1489, 2018.

[21] S. L. Smith, J. Tumova, C. Belta, and D. Rus, "Optimal path planning for surveillance with temporal-logic constraints," *The International Journal of Robotics Research*, vol. 30, no. 14, pp. 1695–1708, 2011.

[22] V. Forejt, M. Kwiatkowska, and D. Parker, "Pareto curves for probabilistic model checking," in *Automated Technology for Verification and Analysis*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 317–332.

[23] K. Etessami, M. Kwiatkowska, M. Y. Vardi, and M. Yannakakis, "Multi-objective model checking of markov decision processes," in *Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2007, pp. 50–65.

[24] M. Kwiatkowska, G. Norman, and D. Parker, "Prism 4.0: Verification of probabilistic real-time systems," in *Computer Aided Verification*. Springer, 2011, pp. 585–591.

[25] E. M. Wolff, U. Topcu, and R. M. Murray, “Robust control of uncertain markov decision processes with temporal logic specifications,” in *IEEE Conference on Decision and Control (CDC)*, 2012, pp. 3372–3379.

[26] J. Wang, X. Ding, M. Lahijanian, I. C. Paschalidis, and C. A. Belta, “Temporal logic motion control using actor–critic methods,” *The International Journal of Robotics Research*, vol. 34, no. 10, pp. 1329–1344, 2015.

[27] J. Tumova, G. C. Hall, S. Karaman, E. Frazzoli, and D. Rus, “Least-violating control strategy synthesis with safety rules,” in *International Conference on Hybrid Systems: Computation and Control*, 2013, pp. 1–10.

[28] C.-I. Vasile, J. Tumova, S. Karaman, C. Belta, and D. Rus, “Minimum-violation scLTL motion planning for mobility-on-demand,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 1481–1488.

[29] R. S. Sutton, D. Precup, and S. Singh, “Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning,” *Artificial intelligence*, vol. 112, no. 1-2, pp. 181–211, 1999.

[30] P. Tabuada, *Verification and control of hybrid systems: a symbolic approach*. Springer Science & Business Media, 2009.

[31] P.-J. Meyer and D. V. Dimarogonas, “Hierarchical decomposition of LTL synthesis problem for nonlinear control systems,” *IEEE Transactions on Automatic Control*, vol. 64, no. 11, pp. 4676–4683, 2019.

[32] P. Nilsson and N. Ozay, “Incremental synthesis of switching protocols via abstraction refinement,” in *53rd IEEE Conference on Decision and Control*, 2014, pp. 6246–6253.

[33] S. Haesaert, S. E. Zadeh Soudjani, and A. Abate, “Verification of general markov decision processes by approximate similarity relations and policy refinement,” *SIAM Journal on Control and Optimization*, vol. 55, no. 4, pp. 2333–2367, 2017.

[34] C. Baier and J.-P. Katoen, *Principles of Model Checking*. MIT press Cambridge, 2008.

[35] J. Klein, “ltl2dstar-LTL to deterministic streett and rabin automata,” `http://www.ltl2dstar.de`, 2007.

[36] P. Schillinger, M. Bürger, and D. V. Dimarogonas, “Hierarchical LTL-task MDPs for multi-agent coordination through auctioning and learning,” *The International Journal of Robotics Research*, 2019.

[37] M. Guo and D. V. Dimarogonas, “Task and motion coordination for heterogeneous multiagent systems with loosely coupled local tasks,” *IEEE Transactions on Automation Science and Engineering*, vol. 14, no. 2, pp. 797–808, 2016.

[38] Y. Kantaros, M. Malencia, V. Kumar, and G. J. Pappas, “Reactive temporal logic planning for multiple robots in unknown environments,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 11 479–11 485.

[39] G. H. Polychronopoulos and J. N. Tsitsiklis, “Stochastic shortest path problems with recourse,” *Networks: An International Journal*, vol. 27, no. 2, pp. 133–143, 1996.

[40] A. Pnueli, “The temporal semantics of concurrent programs,” *Theoretical Computer Science*, vol. 13, no. 1, pp. 45–60, 1981.

[41] MDP_TG, `https://github.com/MengGuo/P_MDP_TG`.

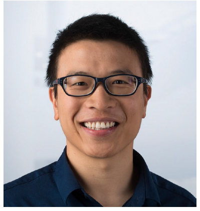
[42] X. C. Ding, S. L. Smith, C. Belta, and D. Rus, “Mdp optimal control under temporal logic constraints,” in *Decision and Control (CDC), IEEE Conference on*, 2011, pp. 532–538.

[43] E. Altman, *Constrained Markov decision processes*. CRC Press, 1999, vol. 7.

[44] D. P. Bertsekas and J. N. Tsitsiklis, “An analysis of stochastic shortest path problems,” *Mathematics of Operations Research*, vol. 16, no. 3, pp. 580–595, 1991.

[45] D. Droeschel, M. Schwarz, and S. Behnke, “Continuous mapping and localization for autonomous navigation in rough terrain using a 3d laser scanner,” *Robotics and Autonomous Systems*, vol. 88, pp. 104–115, 2017.

[46] G. L. O. Solver, `https://developers.google.com/optimization/lp`.



Meng Guo (M’11) received the M.Sc. degree (2011) in System, Control, and Robotics and the Ph.D. degree (2016) in Electrical Engineering from KTH Royal Institute of Technology, Sweden. He was a postdoctoral associate with the Department of Mechanical Engineering and Materials Science, Duke University, USA. During 2018-2021, he worked as a senior research scientist on Reinforcement Learning and Planning at the Bosch Center for Artificial Intelligence (BCAI), Germany. Since 2022, he is an assistant professor at the Department of Mechanics and Engineering Science, College of Engineering, Peking University, China. His main research interests include task and motion planning for robotic systems.

Tianjun Liao received the bachelor’s degree from the National University of Defense Technology, Changsha, China, in 2007, and Ph.D. degree from the Institute de Recherches Interdisciplinaires et de Developpements en Intelligence Artificielle, Universite Libre de Bruxelles, Brussels, Belgium, in 2013. He is currently with the Academy of Military Sciences, Beijing, China. His research interests include heuristic optimization algorithms, automated algorithm configuration, and swarm intelligence.

Junjie Wang received the B.E. degree (2020) in Automation from Beihang University, Beijing, China. He is currently a PhD candidate at the College of Engineering, Peking University, China. His research interests include formal control synthesis for multi-agent systems.

Zhongkui Li (M’11, SM’21) received the B.S. degree in space engineering from the National University of Defense Technology, China, in 2005, and his Ph.D. degree in dynamics and control from Peking University, China, in 2010. Since 2013, Dr. Li has been with the Department of Mechanics and Engineering Science, College of Engineering, Peking University, China, where he is currently a tenured Associate Professor. He has authored the book *Cooperative Control of Multi-Agent Systems: A Consensus Region Approach* (CRC press, 2014) and has published a number of journal papers. His current research interests include multi-agent systems, autonomous unmanned systems, cooperative control, motion and task planning. Dr. Li was the recipient of the State Natural Science Award of China (Second Prize) in 2015, the Yang Jiachi Science and Technology Award in 2015, and the National Excellent Doctoral Thesis Award of China in 2012. His coauthored papers received the *IET Control Theory & Applications* Premium Award in 2013 and the Best Paper Award of *Journal of Systems Science & Complexity* in 2012. He serves as an Associate Editor of *IEEE Transactions on Automatic Control*, *Nonlinear Analysis: Hybrid Systems*, and the Conference Editorial Board of IEEE Control Systems Society.