2022 IEEE International Conference on Robotics and Automation (ICRA)
May 23-27, 2022. Philadelphia, PA, USA

Temporal Logic Guided Motion Primitives for Complex Manipulation Tasks with User Preferences

Hao Wang, Haoyuan He, Weiwei Shang, and Zhen Kan

***Abstract*— Dynamic movement primitives (DMPs) are a flexible trajectory learning scheme widely used in motion generation of robotic systems. However, existing DMP-based methods mainly focus on simple go-to-goal tasks. Motivated to handle tasks beyond point-to-point motion planning, this work presents temporal logic guided optimization of motion primitives, namely PI$^{\text{BB-TL}}$ algorithm, for complex manipulation tasks with user preferences. In particular, weighted truncated linear temporal logic (wTLTL) is incorporated in the PI$^{\text{BB-TL}}$ algorithm, which not only enables the encoding of complex tasks that involve a sequence of logically organized action plans with user preferences, but also provides a convenient and efficient means to design the cost function. The black-box optimization is then adapted to identify optimal shape parameters of DMPs to enable motion planning of robotic systems. The effectiveness of the PI$^{\text{BB-TL}}$ algorithm is demonstrated via simulation and experiment.**

I. INTRODUCTION

By operating beyond structured environments, robots are moving towards applications in complex and unstructured environments, including offices, hospitals, homes [1]. These applications often require robots to be able to autonomously learn, plan, and execute a variety of challenging manipulations. As an enabling technique, dynamic movement primitives (DMPs) have emerged as a flexible trajectory learning scheme [2]–[4] for manipulators and mobile robots, such as biped walking of humanoid robots [5], collaborative manipulation of clothes [6], bimanual tasks [7]. While it is powerful, conventional DMPs are mainly limited to simple go-to-goal tasks. Another challenge that receives little attention in prior DMP-based approaches is the problem of capturing user preferences during motion and trajectory generation. Therefore, this work is particularly motivated to improve DMP-based motion generation skills for robotic systems to perform manipulations that consist of a sequence of logically organized actions with user preferences.

Related works: DMPs are encoded as a combination of simple linear dynamical systems with nonlinear components to acquire smooth movements of arbitrary shape. One common objective is to obtain the optimal parameters of nonlinear components. Due to its good scaling properties with respect to the initial/end positions, the parameter linearity, the rescaling robustness, and the continuity [8], DMPs have been widely used with policy search reinforcement learning to identify a best policy (i.e., the optimal DMP parameters). Example policy search methods include gradient based approaches [9], expectation maximization based approaches [10], and information-theoretic approaches [11]. The policy improvement with path integrals algorithm PI^2 was derived from the first principles of stochastic optimal control [12], [13]. Different with gradient-based reinforcement learning algorithms, the PI^2 algorithm avoids the curse of dimensionality and gradient estimation by using parameterized policies with probability-weighted averaging. However, the motion primitive parameters for each dimension must be optimized individually when using PI^2 algorithm, which increases the computational complexity and reduces the parameter convergence rate. A stochastic optimization algorithm, namely Policy Improvement with Black-Box Optimization PI$^{\text{BB}}$, was then developed in [14]. As a special case of PI^2, PI$^{\text{BB}}$ uses the same method for exploration and parameter updating, but differs in using black-box optimization rather than reinforcement learning for policy improvement. Since PI$^{\text{BB}}$ is based on covariance matrix adaptation through weighted averaging, it can optimize the parameters of motion primitives with multiple dimensions simultaneously for improved efficiency and convergence rate. Other representative works that exploit DMP for motion generation include [7], [12], [15]. However, limited to the form of the expect cost function, neither PI$^{\text{BB}}$ nor PI^2 can handle parameter optimization of motion primitives for manipulation tasks with complex logic and temporal constraints.

Temporal logic, as a formal language, is capable of describing a wide range of complex tasks in a succinct and human-interpretable form, and thus has been increasingly used in the motion planning of robotic systems [16]–[21]. Signal temporal logic (STL) is defined over continuous signals and its quantitative semantics, known as robustness, can measure the degree of satisfaction or violation of the desired task specification [22]. To maximize the robustness of STL, the synthesis problem is often cast as optimization problems and then solved using heuristics, mixed-integer programming or gradient methods [23]–[27]. However, STL formulas specify subtasks with explicit time bounds, while many manipulation tasks only require the subtasks to be performed in a desired sequence (e.g., open the fridge door, take the milk out, and close the fridge door). Manually assigning time bounds for subtasks might lead to the failure of finding desired policy due to unexpected environmental events. Other possible formal languages, such as BLTL [28]

H. Wang, H. He, W. Shang and Z. Kan (Corresponding Author) are with the Department of Automation at the University of Science and Technology of China, Hefei, Anhui, China, 230026. This work was supported in part by the National Natural Science Foundation of China under Grant U2013601 and Grant 62173314.

2022 IEEE International Conference on Robotics and Automation (ICRA) | 978-1-7281-9681-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICRA46639.2022.9811707

and LTL$_f$ [29], either require time bounds similar to STL or does not come with quantitative semantics. In contrast to the aforementioned methods, truncated linear temporal logic (TLTL) is a predicate temporal logic without time bounds, which is defined over finite-time trajectories of robot's states and provides a unifying and interpretable way to specify tasks [30]. In [31], TLTL was successfully used to specify a robotic cooking task. Despite its recent success, TLTL is mainly used for high-level motion planning and few effort has been devoted to extending TLTL with general trajectory generation approaches, such as DMP based methods.

Contributions: This work considers motion generation for a robotic system to perform logically and temporally structured manipulations with user preferences. Since conventional DMP based approaches (e.g., PI^2 or PI$^{\text{BB}}$) suffer from handcrafted cost function and are mainly used for simple point-to-point tasks, the first contribution is to develop the PI$^{\text{BB-TL}}$ algorithm by extending the state-of-the-art motion generation PI$^{\text{BB}}$ algorithm with TLTL. Compared with most existing works, the use of TLTL not only enables the encoding of complex tasks that involve a sequence of logically organized action plans, but also provides a convenient and effective means to design the cost function. Close to our work, LTL specifications was also incorporated in the learning of DMPs in [32]. However, the loss function designed in [32] is limited in evaluating whether or not a given LTL specification is satisfied and the log-sum-exponential approximation is over-approximated. In contrast, the weighted TLTL robustness in this work is sound that can not only qualitatively evaluate the satisfaction of LTL specifications, but also quantitively determine its satisfaction degree. Specifically, the cost function is designed based on the TLTL robustness and the smooth approximations in PI$^{\text{BB}}$ algorithm to optimize the shape parameters of DMPs, ensuring that the generated trajectory of DMPs satisfies complex tasks specified by TLTL constraints. Another contribution is to take into account user preferences in motion generation. Inspired by [26], we further extend TLTL to weighted TLTL (wTLTL) to capture sub-task with different importance or priorities. Incorporating DMPs with wTLTL ensures that the generated trajectory satisfies the given manipulation task with user specified preference. The effectiveness of PI$^{\text{BB-TL}}$ is demonstrated via simulation and experimental results.

II. PRELIMINARIES

A. Dynamic Movement Primitives

Dynamic movement primitives are a flexible representation of robot trajectories [2], which can be expressed as

$$\frac{1}{\tau}\dot{z}_t = \alpha_z\left(\beta_z\left(g - y_t\right) - z_t\right) + f_t, \tag{1a}$$

$$\frac{1}{\tau}\dot{y}_t = z_t, \tag{1b}$$

$$f_t = \varsigma_t^T\theta, \tag{1c}$$

$$\frac{1}{\tau}\dot{s}_t = -\alpha_s s_t, \tag{1d}$$

where $y_t \in \mathbb{R}$ and $\dot{y}_t \in \mathbb{R}$ represent the position and velocity, respectively, $z_t \in \mathbb{R}$ and $s_t \in \mathbb{R}$ are internal states, α_z, β_z, τ and α_s are positive scale factors, $\theta \in \mathbb{R}^L$ is the parameter vector, and $g \in \mathbb{R}$ is the goal position. The nonlinear function f_t allows the generation of arbitrary complex movements, which consists of basis functions $\varsigma_t \in \mathbb{R}^L$ represented by a piecewise linear function approximator with weighted Gaussian kernels $\varpi \in \mathbb{R}^L$ as

$$\left[\varsigma_t\right]_i = \frac{\varpi_i\left(s_t\right)\cdot s_t}{\sum_{l=1}^{L}\varpi_l\left(s_t\right)}\left(g - x_0\right), \tag{2}$$

$$\varpi_i = \exp\left(-\frac{1}{2\sigma_i}\left(s_t - c_i\right)^2\right), \tag{3}$$

where $\left[\varsigma_t\right]_i$ denotes the ith entry of ς_t, and σ_i and c_i represent the variance and mean, respectively. The core idea behind DMPs is to perturb the term of $\alpha_z\left(\beta_z\left(g - y_t\right) - z_t\right)$ by a nonlinear $\varsigma_t^T\theta$ to acquire smooth movements of arbitrary shape. Although only a 1-D system is represented in (1), multi-dimensional DMPs can be represented by coupling several dynamical systems using shared phase variable s_t, where each dimension has its own goal g and shape parameters θ. Reinforcement learning or black-box optimizations can then be used to obtain the optimal shape parameters θ.

B. Weighted Truncated Linear Temporal Logic

Truncated linear temporal logic was introduced in [30], which is able to incorporate domain knowledge and various constraints to describe complex robotic tasks. In [26], weighted signal temporal logic was developed to model user preferences. Inspired by the works of [26] and [30], we extend traditional TLTL to weighted TLTL (wTLTL) in this work to specify complex robotic missions with weights modeling relative importance and priority among temporal logic constraints (e.g., user preferences over sub-tasks).

The syntax of wTLTL is defined as

$$\begin{aligned}\varphi :=& \top \mid f\left(y_t\right) < c \mid \neg\phi \mid \wedge^w\phi_i \mid \vee^w\psi_i \mid \\ & \Diamond\phi \mid \Box\phi \mid \phi\mathcal{U}\psi \mid \phi\mathcal{T}\psi \mid \phi \Rightarrow \psi,\end{aligned} \tag{4}$$

where $\top$ is the boolean constant true, $f\left(y_t\right) < c$ is a predicate where $f\colon \mathbb{R}^n \to \mathbb{R}$ maps a system state y_t to a constant, $\neg$ (negation), $\vee$ (disjunction) and $\wedge$ (conjunction) are standard Boolean operators, $\Diamond$ (eventually), $\Box$ (always), $\mathcal{U}$ (until), $\mathcal{T}$ (then), and $\Rightarrow$ (implication) are temporal operators.

Given N conjunctions and disjunctions, the positive weight vector is denoted by $w \in \mathbb{R}^N_{>0}$, where the ith entry w_i associated with conjunction or disjunction indicates the corresponding relative importance of obligatory specifications or priorities of alternatives. Importance allows the trade-off of all specifications that need to be satisfied (i.e., conjunctions), while priorities allows the trade-off of the acceptance of alternative specifications (i.e., disjunctions). A higher value of w_i corresponds to a higher importance or priority. In the following sections, when the weight vector of a Boolean operator ($\vee$ or $\wedge$) is a unit vector, i.e. $w_i = 1$, $\forall i \in \{1, \ldots, N\}$, we omit the w in the wTLTL formula.

Note that the traditional TLTL in [30] can be considered as a special case of wTLTL where all w are unit vectors.

The semantics of wTLTL formulas is defined over finite trajectories of system states (e.g., a trajectory generated by DMP). Let $y_{t:t+k}$ denote a sequence of states from y_t to y_{t+k}. Denote by $y_{t:t+k} \vDash \phi$ if the trajectory $y_{t:t+k}$ satisfies a wTLTL formula ϕ. More expressions can be achieved by combing temporal and Boolean operators. For instance, $y_{t:t+k} \vDash \Box\phi$ indicates that ϕ is satisfied for every subtrajectory $y_{t':t+k}$, $\forall t' \in [t, t+k)$, $y_{t:t+k} \vDash \Diamond\phi$ indicates that ϕ is satisfied for at least one subtrajectory $y_{t':t+k}$ for some $t' \in [t, t+k)$, and $y_{t:t+k} \vDash \phi\mathcal{T}\psi$ indicates that ϕ is satisfied at least once before ψ is satisfied between t and $t+k$.

The wTLTL has both qualitative and quantitative semantics. The qualitative semantics indicates whether or not a trajectory satisfies a specification, while the quantitative semantics (also referred to as the robustness) quantifies the degree of satisfaction of a specification.

Definition 1 (wTLTL Robustness)**.** Given a specification φ and a trajectory $y_{t:t+k}$, the wTLTL robustness is defined as

$$\begin{aligned}
\rho^w\left(y_{t:t+k}, \top\right) &= \rho^w_{max}, \\
\rho^w\left(y_{t:t+k}, f\left(y_t\right) < c\right) &= c - f\left(y_t\right), \\
\rho^w\left(y_{t:t+k}, \neg\phi\right) &= -\rho^w\left(y_{t:t+k}, \phi\right), \\
\rho^w\left(y_{t:t+k}, \wedge^w\phi_i\right) &= \otimes^\wedge\left(w, [\rho\left(y_{t:t+k}, \phi_1\right), \ldots, \right. \\
&\qquad \left. \rho\left(y_{t:t+k}, \phi_N\right)]\right), \\
\rho^w\left(y_{t:t+k}, \vee^w\psi_i\right) &= \oplus^\vee\left(w, [\rho\left(y_{t:t+k}, \psi_1\right), \ldots, \right. \\
&\qquad \left. \rho\left(y_{t:t+k}, \psi_N\right)]\right),
\end{aligned}$$

where $\rho^w_{max} \in \mathbb{R}$ represents the maximum robustness value and $\rho\left(y_{t:t+k}, \phi_i\right)$, $i = 1, \ldots, N$, are standard robustness of TLTL as defined in [30]. The aggregation functions $\otimes^\wedge$ and $\oplus^\vee$ are associated with conjunctions and disjunctions, respectively, which satisfy $\min\left(\mathbf{x}\right) \cdot \otimes^\wedge\left(w, \mathbf{x}\right) > 0$ and $\max\left(\mathbf{x}\right) \cdot \oplus^\vee\left(w, \mathbf{x}\right) > 0$, $\forall \mathbf{x} \in \mathbb{R}^N$ and $\mathbf{x} \neq 0$, and are defined as

$$\begin{aligned}
\otimes^\wedge\left(w, \mathbf{x}\right) &= \min_{i=1:N}\left\{\left(\left(\frac{1}{2} - \bar{w}_i\right)\mathrm{Sign}\left(x_i\right) + \frac{1}{2}\right) \cdot x_i\right\}, \\
\oplus^\vee\left(w, \mathbf{x}\right) &= -\otimes^\wedge\left(w, -\mathbf{x}\right),
\end{aligned} \tag{5}$$

where $\bar{w}_i = \frac{w_i}{\sum_{j=1}^N w_j}$ is the normalized weight [26]. The definition of robustness for temporal operators ($\Diamond$, $\Box$, $\mathcal{U}$, $\mathcal{T}$, and $\Rightarrow$) is the same as the standard robustness of TLTL as defined in [30].

The wTLTL robustness in Def. 1 is *sound* in the sense that a strictly positive robustness ρ^w indicates satisfaction of the formula φ, and a strictly negative robustness ρ^w indicates violation of φ. That is, $\rho^w\left(y_{t:t+k}, \varphi\right) > 0$ implies $y_{t:t+k} \vDash \varphi$, and $\rho^w\left(y_{t:t+k}, \varphi\right) < 0$ implies $y_{t:t+k} \nvDash \varphi$. Following similar analysis in Theorem 2 of [26], it is trivial to show that the wTLTL robustness is sound. Due to its soundness property, ρ^w will be exploited in the subsequent development to facilitate the design of cost functions in PI$^{\text{BB-TL}}$ algorithm to enable complex task manipulation with specified user preferences.

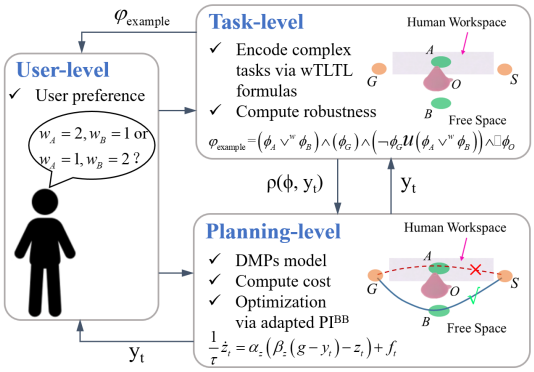

Fig. 1: The architecture of PI$^{\text{BB-TL}}$ for complex manipulation tasks with user preferences. In the task-level, the wTLTL specification encodes the complex manipulation task described in Example 1 and its robustness indicates how well the task is performed. In the planning-level, satisfactory trajectories are generated by optimizing the shape parameters of DMPs via adapted PI$^{\text{BB}}$ algorithm. User preferences are incorporated in both task and planning levels to indicate the relative importance of subtasks or priorities of alternatives.

C. Smooth Approximations

In literature, various approaches can be employed to approximate min and max. For instance, the max and min can be approximated by

$$\tilde{\min}\left(\left[a_1, a_2 \ldots, a_m\right]^T\right) = -\frac{1}{k_1}\log\left(\sum_{i=1}^m e^{-k_1 a_i}\right), \tag{6}$$

$$\tilde{\max}\left(\left[a_1, a_2 \ldots, a_m\right]^T\right) = \frac{\sum_{i=1}^m a_i e^{k_2 a_i}}{\sum_{i=1}^m e^{k_2 a_i}}, \tag{7}$$

where $k_1, k_2 > 0$ are adjustable parameters [33]. It is shown in [33] that (6) and (7) are under-approximation of the true minimum and maximum, i.e., it is always true that $\tilde{\min}\left(\mathbf{a}\right) \leq \min\left(\mathbf{a}\right)$ and $\tilde{\max}\left(\mathbf{a}\right) \leq \max\left(\mathbf{a}\right)$.

III. Problem Formulation

Consider a robotic system whose motion is represented by DMPs in (1) with known initial position y_0 and goal position g, and unknown shape parameters θ. Let $y_{0:t}$ denote a finite robot trajectory starting form y_0. The complex manipulation to be performed by the robot is specified by a wTLTL formula φ and its predicates are interpreted over the trajectory $y_{0:t}$. The goal of this work is to identify an optimal shape parameter vector of DMPs $\theta^* = \underset{\theta}{\text{argmin}} J$, where J is a cost function to be designed, such that the incurred robot motion satisfies (1) and the formula φ, i.e., $y_{0:t}\left(y_0, g, \theta^*\right) \vDash \varphi$.

To address this motion planning problem, the PI$^{\text{BB-TL}}$ algorithm is developed. The core idea of PI$^{\text{BB-TL}}$ is to incorporate DMPs and wTLTL to enable complex manipulations, where wTLTL provides interpretable task

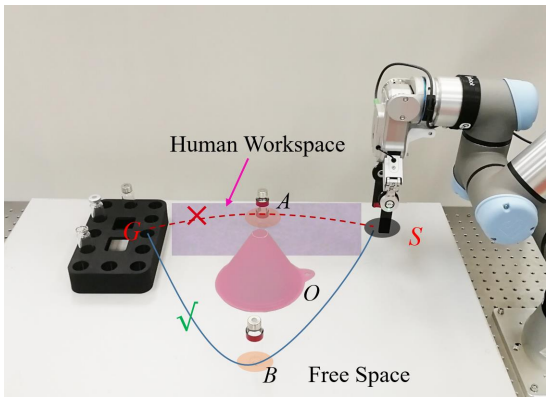

Fig. 2: A complex manipulation task with user preference, where S represents the initial position, G represents the goal position, O represents the obstacles. Human workspace is indicated by the shaded area.

specifications to instruct the robot what to do in the task-level, while DMPs are used to generate robot trajectories to instruct the robot how to do in the planning-level. User preferences are further incorporated via wTLTL and the cost function into the task and planning level, respectively, to bias the generated robotic trajectory towards user preferences. The architecture of PI$^{\text{BB-TL}}$ algorithm is illustrated in Fig. 1.

To elaborate the PI$^{\text{BB-TL}}$ algorithm, a running example is used throughout this work.

Example 1. Consider a task that requires the end effector of a manipulator to transport chemical reagent from the initial position S to the reagent rack G in Fig. 2. The manipulator is tasked to visit region A (in human workspace) or B (in free space) before reaching G, and do not visit G until A or B is visited, and always avoid obstacle O. Such a task can be encoded by a wTLTL formula $\varphi_{example} = (\phi_A \vee^w \phi_B) \wedge \phi_G \wedge (\neg\phi_G \mathcal{U} (\phi_A \vee^w \phi_B)) \wedge \Box\varphi_O$, where ϕ_A, ϕ_B, ϕ_G, ϕ_O are predicates corresponding to region A, B, G and O, respectively. It is worth pointing out that, for such a complex task $\varphi_{example}$, conventional DMP-based approaches [3], [12] are no longer applicable. There are two main challenges to perform $\varphi_{example}$. The first challenge is how conventional DMP can be extended to learn optimal shape parameters of motion primitives for complex tasks at a time. Another challenge is how the optimal shape parameters in DMP can be learned while considering user preferences. For instance, suppose there are two feasible plans, i.e., the dashed and solid lines in Fig. 2. Even the path via A to the goal G is shorter, the path via B is more favorable since it avoids human workspace to reduce the potential collision of chemical reagent with human operator. Such a preference should be encoded in the motion planning strategy.

IV. Cost Function Design and Policy Improvement

This section first presents the cost function design in Sec. IV-A, which exploits wTLTL robustness to guide motion planning for complex tasks. Section IV-B then presents how the black-box optimization algorithm can be adapted to solve the synthesis problem.

A. TLTL Robustness Guided Cost Function Design

Due to the soundness property of $\rho^w(y_{0:t}, \varphi)$, the goal is to design a cost function J such that minimizing J is equivalent to maximizing the satisfaction degree of the wTLTL specification φ. Optimization methods can then be employed to identify the optimal shape parameters θ^* that minimize the cost function J. However, the traditional robustness only considers satisfaction of a TLTL formula at the most extreme sub-formulas, hindering the optimization to find feasible solutions. To address this issue, inspired by [26], [33], we refine wTLTL robustness by accumulating and averaging the robustness over all sub-formulas.

Definition 2 (Smoothed wTLTL Robustness)**.** Given a specification φ and a state trajectory $y_{t:t+k}$, the smoothed wTLTL robustness $\widetilde{\rho}$ is defined as follows:

$$
\begin{aligned}
\widetilde{\rho}(y_{t:t+k}, \top) &= \widetilde{\rho}^w_{max}, \\
\widetilde{\rho}(y_{t:t+k}, f(y_t) < c) &= c - f(y_t), \\
\widetilde{\rho}(y_{t:t+k}, \neg\phi) &= -\widetilde{\rho}(y_{t:t+k}, \phi), \\
\widetilde{\rho}(y_{t:t+k}, \Phi \wedge \Psi) &= \tilde{\min}(\rho^w(y_{t:t+k}, \Phi), \rho^w(y_{t:t+k}, \Psi)), \\
\widetilde{\rho}(y_{t:t+k}, \Phi \vee \Psi) &= \tilde{\max}(\rho^w(y_{t:t+k}, \Phi), \rho^w(y_{t:t+k}, \Psi)), \\
\widetilde{\rho}(y_{t:t+k}, \Box\phi) &= \tilde{\min}_{t' \in [t, t+k)} (\rho^w(y_{t:t+k}, \phi)), \\
\widetilde{\rho}(y_{t:t+k}, \Diamond\phi) &= \tilde{\max}_{t' \in [t, t+k)} (\rho^w(y_{t:t+k}, \phi)), \\
\widetilde{\rho}(y_{t:t+k}, \phi \mathcal{U} \psi) &= \tilde{\max}_{t' \in [t, t+k)} \Big(\tilde{\min}\left(\rho^w(y_{t':t+k}, \psi)\right), \\
&\qquad \tilde{\min}_{t'' \in [t, t']} (\rho^w(y_{t'':t'}, \psi)) \Big), \\
\widetilde{\rho}(y_{t:t+k}, \phi \mathcal{T} \psi) &= \tilde{\max}_{t' \in [t, t+k)} \Big(\tilde{\min}\left(\rho^w(y_{t':t+k}, \psi)\right), \\
&\qquad \tilde{\max}_{t'' \in [t, t']} (\rho^w(y_{t'':t'}, \psi)) \Big), \\
\widetilde{\rho}(y_{t:t+k}, \phi \Rightarrow \psi) &= \tilde{\max}(-\rho^w(y_{t:t+k}, \phi), \rho^w(y_{t:t+k}, \psi)).
\end{aligned}
\tag{8}
$$

where $\Phi = \wedge^w \phi_i$ and $\Psi = \vee^w \psi_i$.

In Def. 2, $\tilde{\min}$ and $\tilde{\max}$ are smoothed version of the conventional $\min$ and $\max$ operators which are calculated by (6) and (7) as discussed in Section II-C.

Theorem 1. *Given a wTLTL specification φ and a state trajectory $y_{t:t+k}$, for any pre-defined error bound $\epsilon > 0$, there exist $\bar{k}_1$ and $\bar{k}_2$ such that $0 \leq \rho^w(y_{t:t+k}, \varphi) - \widetilde{\rho}(y_{t:t+k}, \varphi) \leq \epsilon$ for all $k_1 \geq \bar{k}_1$, $k_2 \geq \bar{k}_2$, where k_1 and k_2 are tuning parameters defined in (6) and (7).*

The proof of Theorem 1 is omitted, since it is a trivial extension of Theorem 2 in [33]. Theorem 1 indicates that the smoothed wTLTL robustness $\widetilde{\rho}$ is an under-approximation of ρ^w, and thus it provides a sufficient condition for the satisfaction check of wTLTL specification φ. That is, if

$\widetilde{\rho}(y_{t:t+k},\varphi)>0$, it is always true that $\rho^{w}(y_{t:t+k},\varphi)>0$, which implies $y_{t:t+k}\vDash\varphi$. Theorem 1 also indicates that the smoothed wTLTL robustness $\widetilde{\rho}$ gradually approaches the true robustness ρ as k_1 and k_2 increase, which means that the approximation will not hide any potential solutions if k_1 and k_2 are sufficiently large.

By Theorem 1, we design the cost function J as

$$J\left(\widetilde{\rho}\left(y_{0:t},\varphi\right)\right)=\begin{cases}-\widetilde{\rho}\left(y_{0:t},\varphi\right), & \text{if } \widetilde{\rho}\left(y_{0:t},\varphi\right)<0,\\ 0, & \text{otherwise,}\end{cases} \tag{9}$$

which indicates that minimizing the cost function (i.e., $J\left(\widetilde{\rho}\left(y_{0:t},\varphi\right)\right)\to 0$) can lead to $\widetilde{\rho}(y_{0:t},\varphi)\geq 0$, resulting in that $y_{0:t}\vDash\varphi$ due to the soundness property of wTLTL robustness.

Based on (9), the problem considered in Sec. III can then be formally formulated as

$$\begin{aligned}\theta^{*}=&\underset{\theta}{\operatorname{argmin}} J\left(\widetilde{\rho}\left(y_{0:t},\varphi\right)\right),\\ &\text{s.t. } (1) \text{ are satisfied,}\end{aligned} \tag{10}$$

where the constraint $y_{0:t}\left(y_0,g,\theta^{*}\right)\vDash\varphi$ is implicitly encoded in the the cost function J via the smoothed wTLTL robustness. Note that the goal of (10) is to identify a satisfactory trajectory, rather than an optimal trajectory with largest robustness, with respect to logical and temporal specifications and user preferences. Ongoing research is to leverage the wTLTL robustness in the design of the cost function to facilitate the identification of the optimal trajectory.

B. TLTL Guided Black-Box Optimization

After designing the optimization problem in (10), the next step is to identify the optimal shape parameters θ^{*} with respect to J. Two widely used approaches to perform this optimization are reinforcement learning (RL) and black-box optimization (BBO). As discussed in [1] and [34], PI^2 is a RL algorithm which uses reward information during exploratory policy executions, while PI^{BB}, a variant of PI^2 with constant exploration and without temporal averaging, is a BBO algorithm that uses the total reward during execution, which enables that the utility function can be treated as a black box. Despite the lack of theoretical guarantees, strong empirical evidence shows that PI^{BB} has equal or superior performance than PI^2 in terms of convergence speed and robustness of policy improvement. In addition, as a special case of covariance matrix adaptation evolutionary strategy, PI^{BB} is a global optimization method [34], [35].

Motivated by the discussion above, this section presents how the stochastic optimization algorithm PI^{BB} can be adapted in our $\text{PI}^{\text{BB-TL}}$ to solve the optimization problem[1] in (10). Specifically, the $\text{PI}^{\text{BB-TL}}$ algorithm is outlined in Alg. 1, and the shape parameters are updated following the subsequent steps:

[1]In contrast to the traditional TLTL robustness in [30], the smoothed wTLTL robustness $\widetilde{\rho}$ is differentiable, which makes it also suitable for gradient-based optimization methods, e.g., RL-based optimization methods [33]. Since BBO based optimization in general outperforms RL based optimization, this work focuses on adapted stochastic optimization algorithm PI^{BB}.

Algorithm 1 wTLTL Guided $\text{PI}^{\text{BB-TL}}$ Algorithm for DMPs

1: **procedure** INPUT: (the initial shape parameter θ^{init}; the wTLTL specification φ; the exploration levels λ^{init}, λ^{min}, and λ^{max}; the sample size M; the eliteness parameter h)
Output: the optimal shape parameters θ^{*}
Initialization: set $\theta=\theta^{init}$, $\Sigma=\lambda^{init}I$
2: **while** cost $J\left(\widetilde{\rho}\left(y_{0:t},\varphi\right)\right)$ not converged **do**
3: Exploration: sample parameters and compute costs
4: **for** each $m\in M$ **do**
5: $\theta_m\sim\mathcal{N}\left(\theta,\Sigma\right)$
6: Generate a trajectory $y_{0:t}^{m}$ based on current θ_m according to (1)
7: Compute the $\widetilde{\rho}\left(y_{0:t}^{m},\varphi\right)$ according to Def. 1 and Def. 2
8: Compute the cost J_m according to (9)
9: **end for**
10: $J^{min}=\min\left\{J_m\right\}_{m=1}^{M}$
11: $J^{max}=\max\left\{J_m\right\}_{m=1}^{M}$
12: Evaluation: compute weight for each sample
13: **for** each $m\in M$ **do**
14: $P_m=\frac{exp\left(-h\frac{J_m-J^{min}}{J^{max}-J^{min}}\right)}{\sum_{l=1}^{M}exp\left(-h\frac{J_l-J^{min}}{J^{max}-J^{min}}\right)}$
15: **end for**
16: Update: weighted averaging over M samples
17: $\Sigma\leftarrow\sum_{m=1}^{M}\left[P_m\left(\theta_m-\theta\right)\left(\theta_m-\theta\right)^{T}\right]$
18: $\Sigma^{new}\leftarrow\text{boundcovar}\left(\Sigma,\lambda^{min},\lambda^{max}\right)$
19: $\theta^{new}\leftarrow\sum_{m=1}^{M}\left[P_m\theta_m\right]$
20: **end while**
21: **end procedure**

1) Initialization (line 1): Set the mean and covariance $\langle\theta,\Sigma\rangle$ to $\left\langle\theta^{init},\lambda^{init}I\right\rangle$, where I is an identity matrix with appropriate dimension.
2) Exploration (line 3-9): Sample M parameter vectors θ_m, $m=1,\dots M$, from $\mathcal{N}\left(\theta,\Sigma\right)$ according to $\left\{\theta_m=\theta+\epsilon_m\right\}_{m=1}^{M}$, where ϵ_m is sampled from a zero-mean Gaussian distribution with variance Σ. Compute the cost $J_m=J\left(\widetilde{\rho}\left(y_{0:t},\varphi\right)\right)$ of each sample.
3) Evaluation (line 10-15): Compute the weight $P\in\mathbb{R}^{M}$ for the M samples, where the mth entry $P_m\in[0,1]$ and $\sum_{m=1}^{M}P_m=1$. Specifically, using the normalized exponentiation function, P_m is computed as

$$P_m=\frac{\exp\left(-h\frac{J_m-J^{min}}{J^{max}-J^{min}}\right)}{\sum_{r=1}^{M}\exp\left(-h\frac{J_r-J^{min}}{J^{max}-J^{min}}\right)}, \tag{11}$$

where $J^{min}=\min\left\{J_m\right\}_{m=1}^{M}$, $J^{max}=\max\left\{J_m\right\}_{m=1}^{M}$, and h is the eliteness parameter. If a large h is used, only a few samples will contribute to the weighted averaging. If $h=0$, no learning would occur since all given samples have equal weight independent of the cost. In general, low-cost samples have higher weights, and vice versa.
4) Update (line 16-19): Update the parameters $\langle\theta,\Sigma\rangle$ to $\left\langle\theta^{new},\Sigma^{new}\right\rangle$ with weighted averaging according to

$$\Sigma^{new}\leftarrow\text{boundcovar}\left(\Sigma,\lambda^{min},\lambda^{max}\right) \tag{12}$$

and

$$\theta^{new}=\sum_{m=1}^{M}\left[P_m\theta_m\right], \tag{13}$$

where

$$\Sigma = \sum_{m=1}^{M} \left[P_m \left(\theta_m - \theta\right) \left(\theta_m - \theta\right)^T \right] \tag{14}$$

and boundcovar $\left(\Sigma, \lambda^{min}, \lambda^{max}\right)$ is a function that restricts the eigenvalues λ of the covariance matrix Σ within $\left[\lambda^{min}, \lambda^{max}\right]$, i.e., if $\lambda < \lambda^{min}$, then $\lambda = \lambda^{min}$; if $\lambda > \lambda^{max}$, then $\lambda = \lambda^{max}$. The low-cost samples contribute more to the update since they have higher weights, leading to that θ moves towards θ^*.

The visualization of one parameter update using PI$^{\text{BB-TL}}$ in a 2D search space can be found in [36]. Given that the sample size is M and the number of updates required for cost convergence is Λ, the computational complexity of PI$^{\text{BB-TL}}$ is $O(\Lambda M)$ according to Alg. 1.

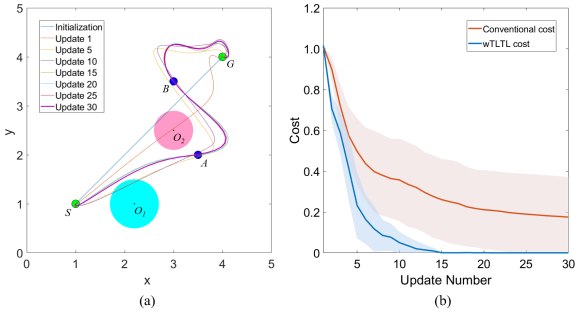

Fig. 3: (a) The evolution of generated trajectories using PI$^{\text{BB-TL}}$ algorithm in Case 1 for a sequential goal reaching task with obstacle avoidance. (b) Learning curves of the wTLTL guided cost and the conventional cost trained with PI$^{\text{BB}}$ in Case 1, respectively. The solid line shows the mean of the cost and the shaded area represents the corresponding standard deviation of 20 training results.

V. Case Studies

The effectiveness of the PI$^{\text{BB-TL}}$ algorithm is demonstrated via two simulation cases and experiment. In Case 1, to show the capability of generating trajectories for complex manipulations, the conventional cost function from [12] is treated as a baseline and compared with the developed wTLTL guided cost function. Consider a wTLTL specification $\varphi_{case1} = (\phi_A \mathcal{T} \phi_B) \wedge (\neg\phi_B \mathcal{U} \phi_A) \wedge \Box\varphi_O$ in Case 1, where $\varphi_O = \wedge_{i=1,2} d_{O_i} > r_{O_i}$ with d_{O_i} representing the Euclidean distance between the robot and the obstacle O_i and r_{O_i} representing the radius of obstacle O_i. In English, φ_{case1} means "visit A then B, do not visit B until A is visited, and always avoid the obstacles O_1 and O_2". Fig. 3 (a) shows the generated trajectories after updates using PI$^{\text{BB-TL}}$ and the evolution of the costs are shown in Fig. 3 (b). To show the capability of handling user preferences in PI$^{\text{BB-TL}}$, we consider a similar case in [26] in Case 2, and the wTLTL formula $\varphi_{case2} = (\phi_A \vee^w \phi_B) \wedge \phi_G \wedge (\neg\phi_G \mathcal{U} (\phi_A \vee^w \phi_B)) \wedge \Box\varphi_O$, where $\varphi_O = d_O > r_O$, d_O is the Euclidean distance between the trajectory and obstacle O, r_O is the radius of obstacle O_i. In English, φ_{case2} means "visit region A or B, then visit region G, and do not visit G until either A or B is visited, while always avoiding obstacle O". The generated trajectories are shown in Fig. 4 (a) and (b) with different weights, respectively. To evaluate practical performance, the developed PI$^{\text{BB-TL}}$ is validated on a physical UR5e robot in a 3D workspace. The experimental setup is shown in Example 1 and the task to be performed by the manipulator is the same as φ_{case2}. Fig. 5 shows the snapshots of the motion of the end effector of the UR5e manipulator. The experiment video is provided.[2] Since the sequential goal reaching task can also be solved by the PI^{2}SEQ algorithm in [12], the performance of PI^{2}SEQ and PI$^{\text{BB-TL}}$ are compared. Table I shows, under the same conditions, the number of DMPs and the number of optimization parameters required in Case 1, Case 2 and UR5e experiment for PI^{2}SEQ and PI$^{\text{BB-TL}}$, respectively. More detailed results about the environment settings, disscussion of case study and computational complexity of the algorithm can be found in [36].

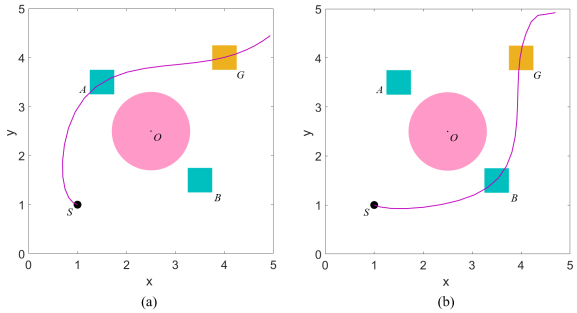

Fig. 4: Simulated trajectories of φ_{case2} with $w_A = 2, w_B = 1$ in (a) and $w_A = 1, w_B = 2$ in (b).

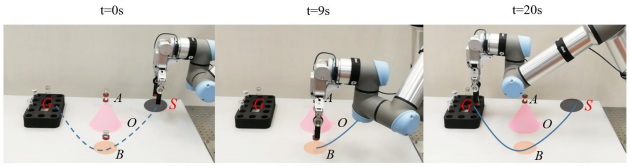

Fig. 5: Snapshots of a manipulation task with complex logic and temporal constraints. The end effector of UR5e transports chemical reagent from S to the reagent rack G via region B with user preference.

TABLE I: Comparison of the number of DMPs and the number of optimization parameters using PI^{2}SEQ and PI$^{\text{BB-TL}}$

	Number of DMPs			Number of Parameters		
	Case 1	Case 2	UR5e	Case 1	Case 2	UR5e
PI^{2}SEQ	6	4	4	60	40	40
PI$^{\text{BB-TL}}$	2	2	2	20	20	20

VI. Conclusion

A temporal logic guided PI$^{\text{BB-TL}}$ algorithm is developed to generate desired motions for complex manipulation tasks with user preferences. Ongoing research will consider online adaptive optimization and reactive temporal logic planning for dynamic workspace with mobile obstacles and time-varying missions.

[2]https://www.youtube.com/watch?v=IGnVKqC-T-A

REFERENCES

[1] K. Chatzilygeroudis, V. Vassiliades, F. Stulp, S. Calinon, and J.-B. Mouret, "A survey on policy search algorithms for learning robot controllers in a handful of trials," *IEEE Trans. Robot.*, vol. 36, no. 2, pp. 328–347, 2020.

[2] S. Schaal, "Dynamic movement primitives-a framework for motor control in humans and humanoid robotics," in *Adaptive motion of animals and machines*. Springer, 2006, pp. 261–280.

[3] C. L. Bottasso, D. Leonello, and B. Savini, "Path planning for autonomous vehicles by trajectory smoothing using motion primitives," *IEEE Trans. Control Syst. Technol.*, vol. 16, no. 6, pp. 1152–1168, 2008.

[4] S. Dutta, L. Behera, and S. Nahavandi, "Skill learning from human demonstrations using dynamical regressive models for multitask applications," *IEEE Trans. Syst., Man, Cybern., Syst.*, 2018.

[5] M. Raković, B. Borovac, M. Nikolić, and S. Savić, "Realization of biped walking in unstructured environment using motion primitives," *IEEE Trans. Robot.*, vol. 30, no. 6, pp. 1318–1332, 2014.

[6] A. Colomé and C. Torras, "Dimensionality reduction for dynamic movement primitives and application to bimanual manipulation of clothes," *IEEE Trans. Robot.*, vol. 34, no. 3, pp. 602–615, 2018.

[7] A. Gams, B. Nemec, A. J. Ijspeert, and A. Ude, "Coupling movement primitives: Interaction with the environment and bimanual tasks," *IEEE Trans. Robot.*, vol. 30, no. 4, pp. 816–830, 2014.

[8] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: learning attractor models for motor behaviors," *Neural Comput*, vol. 25, no. 2, pp. 328–373, 2013.

[9] J. Peters and S. Schaal, "Reinforcement learning of motor skills with policy gradients," *J. Neural Netw.*, vol. 21, no. 4, pp. 682–697, 2008.

[10] ——, "Policy gradient methods for robotics," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots*. IEEE, 2006, pp. 2219–2225.

[11] C. Daniel, G. Neumann, O. Kroemer, J. Peters *et al.*, "Hierarchical relative entropy policy search," *J. Mach. Learn. Res.*, vol. 17, pp. 1–50, 2016.

[12] F. Stulp, E. A. Theodorou, and S. Schaal, "Reinforcement learning with sequences of motion primitives for robust manipulation," *IEEE Trans Robot*, vol. 28, no. 6, pp. 1360–1370, 2012.

[13] E. Theodorou, J. Buchli, and S. Schaal, "A generalized path integral control approach to reinforcement learning," *J. Mach. Learn. Res.*, vol. 11, pp. 3137–3181, 2010.

[14] F. Stulp and P.-Y. Oudeyer, "Proximodistal exploration in motor learning as an emergent property of optimization," *Dev. Sci.*, vol. 21, no. 4, p. e12638, 2018.

[15] S. Calinon, "A tutorial on task-parameterized movement learning and retrieval," *Intelligent service robotics*, vol. 9, no. 1, pp. 1–29, 2016.

[16] C. Baier and J.-P. Katoen, *Principles of model checking*. MIT press, 2008.

[17] M. Kloetzer and C. Mahulea, "LTL-based planning in environments with probabilistic observations," *IEEE Trans. Autom. Sci. Eng.*, vol. 12, no. 4, pp. 1407–1420, 2015.

[18] M. Cai, H. Peng, Z. Li, and Z. Kan, "Learning-based probabilistic LTL motion planning with environment and motion uncertainties," *IEEE Trans. Autom. Control*, vol. 66, no. 5, pp. 2386–2392, 2020.

[19] M. Cai, S. Xiao, B. Li, Z. Li, and Z. Kan, "Reinforcement learning based temporal logic control with maximum probabilistic satisfaction," in *Proc. Int. Conf. Robot. Autom.* Xi'an, China: IEEE, 2021, pp. 806–812.

[20] M. Cai, M. Hasanbeig, S. Xiao, A. Abate, and Z. Kan, "Modular deep reinforcement learning for continuous motion planning with temporal logic," *IEEE Robot. Autom. Lett.*, vol. 6, no. 4, pp. 7973–7980, 2021.

[21] A. K. Bozkurt, Y. Wang, M. M. Zavlanos, and M. Pajic, "Control synthesis from linear temporal logic specifications using model-free reinforcement learning," in *Int. Conf. Robot. Autom.*, 2020, pp. 10 349–10 355.

[22] O. Maler and D. Nickovic, "Monitoring temporal properties of continuous signals," in *Proc. Formal Techn. Model. Anal. Timed Fault Tolerant Syst.* Springer, 2004, pp. 152–166.

[23] S. Saha and A. A. Julius, "An milp approach for real-time optimal controller synthesis with metric temporal logic specifications," in *Proc. IEEE Amer. Control Conf.*, 2016, pp. 1105–1110.

[24] V. Raman, A. Donzé, M. Maasoumy, R. M. Murray, A. Sangiovanni-Vincentelli, and S. A. Seshia, "Model predictive control with signal temporal logic specifications," in *Proc. IEEE Conf. Decis. Control*, 2014, pp. 81–87.

[25] C. Belta and S. Sadraddini, "Formal methods for control synthesis: An optimization perspective," *Annu. Rev. Control Robot. Auton. Syst.,*, vol. 2, pp. 115–140, 2019.

[26] N. Mehdipour, C. I. Vasile, and C. Belta, "Specifying user preferences using weighted signal temporal logic," *IEEE Control Syst. Lett.*, vol. 5, no. 6, pp. 2006–2011, 2021.

[27] A. G. Puranic, J. V. Deshmukh, and S. Nikolaidis, "Learning from demonstrations using signal temporal logic in stochastic and continuous domains," *IEEE Robot. Autom. Lett.*, vol. 6, no. 4, pp. 6250–6257, 2021.

[28] T. Latvala, A. Biere, K. Heljanko, and T. Junttila, "Simple bounded ltl model checking," in *Int. Conf. Form. Method. Computer-Aided Des.* Springer, 2004, pp. 186–200.

[29] G. De Giacomo and M. Y. Vardi, "Linear temporal logic and linear dynamic logic on finite traces," in *Proc. Int. Jt. Conf. Artif. Intell.* Association for Computing Machinery, 2013, pp. 854–860.

[30] X. Li, C.-I. Vasile, and C. Belta, "Reinforcement learning with temporal logic rewards," in *IEEE Int. Conf. Intell. Robot. Syst.*, 2017, pp. 3834–3839.

[31] X. Li, Z. Serlin, G. Yang, and C. Belta, "A formal methods approach to interpretable reinforcement learning for robotic planning," *Sci. Robot.*, vol. 4, no. 37, 2019.

[32] C. Innes and R. Ramamoorthy, "Elaborating on learned demonstrations with temporal logic specifications," in *Robot.: Sci. Syst.*, 2020.

[33] Y. Gilpin, V. Kurtz, and H. Lin, "A smooth robustness measure of signal temporal logic for symbolic control," *IEEE Control Syst. Lett.*, vol. 5, no. 1, pp. 241–246, 2020.

[34] F. Stulp and O. Sigaud, "Policy improvement methods: Between black-box optimization and episodic reinforcement learning," 2012.

[35] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evolutionary computation*, vol. 9, no. 2, pp. 159–195, 2001.

[36] H. Wang, H. He, W. Shang, and Z. Kan, "Temporal logic guided motion primitives for complex manipulation tasks with user preferences," *arXiv preprint arXiv:2202.04375*, 2022.