A Survey of Optimization-Based Task and Motion Planning: From Classical to Learning Approaches

Zhigen Zhao, Shuo Cheng, Yan Ding, *Student Member, IEEE*, Ziyi Zhou, Shiqi Zhang, *Member, IEEE*, Danfei Xu, and Ye Zhao, *Senior Member, IEEE*

***Abstract*—Task and motion planning (TAMP) integrates high-level task planning and low-level motion planning to equip robots with the autonomy to effectively reason over long-horizon, dynamic tasks. Optimization-based TAMP focuses on hybrid optimization approaches that define goal conditions via objective functions and are capable of handling open-ended goals, robotic dynamics, and physical interaction between the robot and the environment. Therefore, optimization-based TAMP is particularly suited to solve highly complex, contact-rich locomotion and manipulation problems. This survey provides a comprehensive review on optimization-based TAMP, covering first, planning domain representations, including action description languages and temporal logic, second, individual solution strategies for components of TAMP, including AI planning and trajectory optimization (TO), and finally, the dynamic interplay between logic-based task planning and model-based TO. A particular focus of this survey is to highlight the algorithm structures to efficiently solve TAMP, especially hierarchical and distributed approaches. In addition, the survey emphasizes the synergy between the classical methods and contemporary learning-based innovations, such as large language models. Furthermore, the future research directions for TAMP is discussed in this survey, highlighting both algorithmic and application-specific challenges.**

***Index Terms*—AI planning, large language models (LLMs), robot learning, task and motion planning (TAMP), temporal logic, trajectory optimization (TO).**

Received 2 April 2024; revised 19 June 2024; accepted 15 August 2024. Recommended by Technical Editor S. Zhuang and Senior Editor H. Gao. This work was supported in part by the Office of Naval Research (ONR) under Grant 0023 N000142312223, in part by the National Science Foundation (NSF) under Grant 0023 IIS-1924978, Grant 0023 CMMI-2144309, and Grant 0023 FRR-2328254, and in part by the USDA under Grant 0023 2023-67021-41397. *(Corresponding author: Ye Zhao.)*

Zhigen Zhao, Ziyi Zhou, and Ye Zhao are with the The Laboratory for Intelligent Decision and Autonomous Robots (LIDAR), Georgia Institute of Technology, Atlanta, GA 30318 USA (e-mail: zhigen.zhao@gatech.edu; zhouziyi@gatech.edu; yzhao301@gatech.edu).

Shuo Cheng and Danfei Xu are with the Robot Learning and Reasoning Lab (RL2), Georgia Institute of Technology, Atlanta, GA 30318 USA (e-mail: shuocheng@gatech.edu; danfei@gatech.edu).

Yan Ding and Shiqi Zhang are with the Autonomous Intelligent Robotics (AIR) Group, Binghamton University, Binghamton, NY 13902 USA (e-mail: yding25@binghamton.edu; zhangs@binghamton.edu).

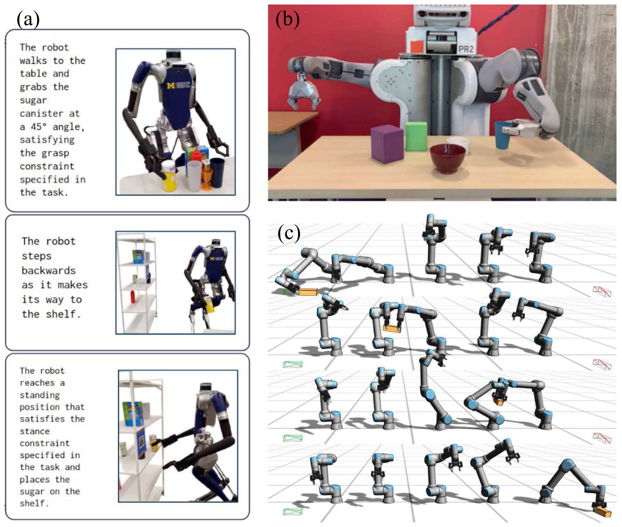
Color versions of one or more figures in this article are available at https://doi.org/10.1109/TMECH.2024.3452509.

Digital Object Identifier 10.1109/TMECH.2024.3452509

I. Introduction

IN RECENT years, robotic systems are rapidly transitioning from structured factory floors to unstructured human-centric environments. To this end, the demand continues to grow for a planning system that enables robots to efficiently perform complex, long-horizon tasks, as exemplified in Fig. 1. To achieve this level of autonomy, robots must be capable of generating and executing feasible and efficient motion plans that allow them to interact with their environment and complete assigned tasks. This complex problem is often framed as robot task and motion planning (TAMP), which breaks a complex, often intractable planning problem into a hybrid symbolic search and a set of local motion planning problems, where each subproblem is tractable to solve.

The main research focus in TAMP is to develop appropriate problem representations and algorithms that efficiently synthesize both symbolic and continuous components of the planning problem [2]. In the existing literature, there are three mainstream classes of TAMP methods: 1) constraint-based TAMP [4], [5], 2) sampling-based TAMP [6], [7], and 3) optimization-based TAMP [8], [9]. Constraint- and sampling-based TAMP characterizes the problem as a set of goal conditions. The solutions are typically found via constraint satisfaction or sampling-based approaches [10], which satisfy the defined goal conditions, but often cannot evaluate or compare the quality of the generated plan or the final state due to the lack of objective functions. In many robotics problems, goals are often expressed as an objective function rather than an explicitly defined set of states. For example, “given a number of rectangular blocks on the table, build a stable structure that is as tall as possible with minimal robot control effort.” This is challenging for traditional sampling-based methods, which often require explicit goal definition and do not have mechanisms to compare plan qualities. As an exception, a specific class of sampling-based motion planning [11] have been proposed to address optimal planning using RRT* and PRM* [12], [13]. However, the complexity and expressiveness of the objective functions are often limited to simple costs, such as path length, time, and energy



The robot walks to the table and grabs the sugar canister at a 45° angle, satisfying the grasp constraint specified in the task.

The robot steps backwards as it makes its way to the shelf.

The robot reaches a standing position that satisfies the stance constraint specified in the task and places the sugar on the shelf.

Fig. 1. Optimization-based TAMP enables dynamic locomotion and manipulation behaviors in complex environments: (a) bipedal robot loco-manipulation [1]; (b) mobile robot table-top manipulation [2]; (c) long-horizon multi-agent collaboration [3].

TABLE I
COMPARISON BETWEEN OPTIMIZATION-BASED VS SAMPLING-BASED TAMP METHODS

Aspect	**Optimization-Based TAMP**	**Sampling-Based TAMP**
Optimality	Capable of converging to optimal or near-optimal solutions	Asymptotically optimal but no guarantee on convergence speed
Task Definition	Expressive definitions via objective functions & constraints	Less expressive due to requirement for explicit goal definition, the objective function is limited to simple forms
Robot Dynamics	Naturally incorporates dynamics in model-based TO	Costly to handle dynamics via steering or forward propagation
Constraint Handling	Handle complex constraints explicitly in model-based TO	Rely on relaxation or simplification to handle manifold constraints
Robustness	Sensitive to initialization and problem setup, potentially trapped by local optima	Robust to variations in initial conditions due to random sampling
Completeness	No mechanism to determine infeasibility	Typically probabilistically complete

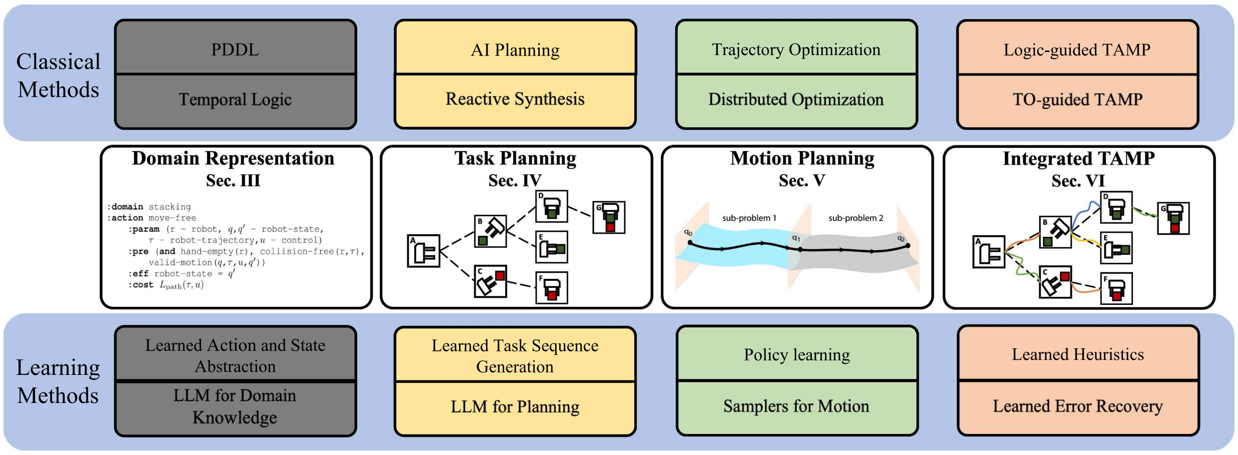
consumption [14]. A comparison between optimization and sampling-based TAMP methods is presented in Table I. For clarification, the scope of this survey focuses on optimization-based TAMP, which naturally defines an objective function for representing the plan quality, in addition to task- and motion-level constraints. This framework enables us to represent and solve a broad range of tasks with complex objective functions.

Optimization-based TAMP optimizes the objective function while adhering to constraints imposed by the robot kinematics and dynamics at the motion planning level and the discrete logic at the task planning level. This motivates the formulation of optimization-based TAMP as a hybrid optimization problem. Optimization-based TAMP naturally incorporates model-based trajectory optimization (TO) methods in motion planning, which allow the planning framework to encode complex robot dynamics, leading to not only feasible but also natural, efficient, and dynamic robot motions. This is especially important for contact-rich applications, such as long-horizon robot manipulation [15] of objects with complex geometry and frictional properties [16], [17], and dynamic locomotion over uneven terrains [18], [19], [20], [21]. In addition, optimization-based TAMP allows the inclusion of more complex objective functions and constraints (e.g., nonlinear and nonconvex ones), enabling the robot to achieve various robot behaviors, thereby enhancing the applicability of robotic systems in real-world deployments.

However, the hybrid optimization problems formed by optimization-based TAMP are often computationally intractable. A successful planning algorithm needs to simultaneously overcome the combinatorial complexity at the task planning level, and the numerical complexity at the motion planning level. As such, a common theme in optimization-based TAMP is to tradeoff between the complexity of the optimization and comprehensiveness of the information included in the planning problem. Either extreme of this tradeoff tends to degrade either the quality or the computational efficiency of the resulting robot plans. In addition, optimization-based TAMP faces several limitations comparing to sampling-based methods as follows:

1) it is sensitive to the initial and goal conditions of the problem setup, which can lead to failures in complex environments, such as complex obstacle geometry or difficult terrain, where certain initial and goal conditions can make it particularly challenging to find the optimal solution;
2) the optimization results can be dependent on the initialization of decision variables, which might cause the planner to get stuck in local optima;
3) optimization-based methods are not complete, meaning they cannot discover infeasible problems.

Therefore, the challenge remains to improve the robustness of optimization-based TAMP and bridge the gap between planning for long-horizon tasks [22], [23] and generating highly dynamic robot behaviors, showcased in model-based optimal control strategies [24], [25], [26]. The integration of learning-based approaches in TAMP has become a significant research trend, as learning-based approaches offer considerable promise for enhancing the scalability and generalizability of classical TAMP methods. Leveraging learning as heuristics improves the efficiency of classical methods. For example, action feasibility checks during the task sequence search process can be accelerated by a neural feasibility classifier [27], [28]. As an alternative method, generative models offer promising avenues to effectively replace certain components within the classical methods, as illustrated by learned task sequence generation from visual input [29] and the use of large language models (LLMs) for domain knowledge representation and planning [30], [31], [32]. Along another line of research, reinforcement learning



```
:domain stacking
:action move-free
    :param (r - robot, q,q' - robot-state,
        τ - robot-trajectory,u - control)
    :pre (and hand-empty(r), collision-free(r,τ),
        valid-motion(q,τ,u,q'))
    :eff robot-state = q'
    :cost L_path(τ,u)
```

Fig. 2. Overview of the problem structures and related algorithms in optimization-based TAMP discussed in this survey paper.

(RL)-based skill learning has been studied in conjunction with the symbolic interface of a task planner, resulting in reusable skill learning that is generalizable across long-horizon tasks [33], [34].

A. Survey Goals and Roadmap

This work is inspired by and builds upon previous surveys in TAMP [2], [10], [35], [36], but carries the unique overarching goal of reviewing the historical background and state-of-the-art optimization-based TAMP, and illustrating the connection between classical methods and the recent development in learning methods. Portions of this work are inspired by recent surveys in other relevant areas, such as logic programming [23], formal methods [37], distributed optimization [38], and TO for legged locomotion [24]. In addition, research contributions originated from 19 countries are highlighted to provide a global research landscape on TAMP innovations (see Fig. 3).

We aim to provide discussion, promising solutions, and future trends in the following questions.

1) *Q1:* Why are optimization-based methods important for TAMP? What are the benefits?
2) *Q2:* How will solutions for each individual component of optimization-based methods inform the strategies to solve integrated TAMP?
3) *Q3:* What common structures are observed in optimization-based TAMP problems, and which tools and strategies can exploit these structures to efficiently generate long-horizon, dynamic robot plans?
4) *Q4:* How to leverage machine learning algorithms to enable robust, and generalizable TAMP frameworks?

Q1 and Q2 motivate us to explore the key components and critical features of optimization-based TAMP, including problem formulation (see Section II), domain representation (see Section III), task planning (see Section IV), and motion planning (see Section V). Q3 seeks to present the current strategies and remaining challenges of optimization-based TAMP (see Section VI), and inspire improved TAMP frameworks that efficiently manage complex task structures and robot dynamics.

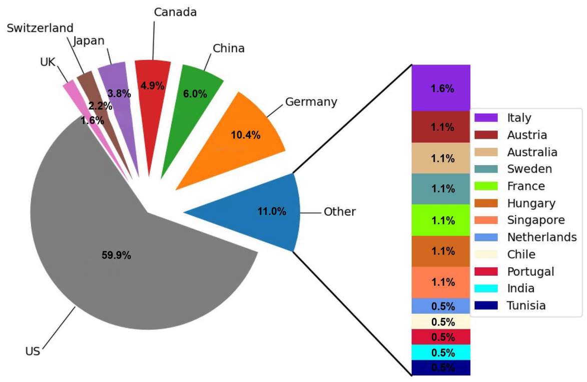


Fig. 3. Percentage statistics sorted by countries of origin. In total, 182 references that directly address TAMP are included.

A particular focus of the discussion is on the interaction between task planning and motion planning layers. Q4 addresses advancements in learning-based methods with the intent of synthesizing these elements for the enhancement of TAMP frameworks. The discussion on Q4 is interleaved with the classical methods to place the learning approaches into proper context. An overview of the structure of the survey is illustrated in Fig. 2.

Q1–Q3 serve as an effective introduction for early-stage researchers new to the TAMP field, but also provide background information for experts in one or more of the TAMP components looking to explore an integrated optimization-based TAMP framework. Q4 provides important context for machine learning experts on how to combine learning techniques with classical TAMP. For the research groups currently exploring classical TAMP, this survey provides a systematic overview of the recent works in learning methods.

Finally, we offer our outlook on the potential future research directions in TAMP (see Section VII), including the challenges in incorporating LLMs and skill learning in TAMP, as well as under-explored application areas, such as loco-manipulation and human–robot collaboration (HRC).

II. Preliminaries

In this section, we present the assumptions and definitions involved in optimization-based TAMP, as well as a motivating example that will be used throughout the survey.

A. Assumptions

The following assumptions, adapted from [22], are made to formulate a basic definition for optimization-based TAMP:

1) *A1. Deterministic Transitions:* If a symbolic action is applicable to a symbolic state, applying the action brings the deterministic symbolic transition system to a *single* other symbolic state, similarly for the application of continuous control.
2) *A2. Known Models and Objectives:* The planner has complete knowledge about the continuous state transition system and the continuous dynamic system, as well as the objective function *before* the planning process begins.
3) *A3. Fully Observable Environments:* The planner has *complete knowledge* about the symbolic and continuous states.
4) *A4. Sequential Plans:* The solutions to a TAMP problem are two *linearly ordered* finite sequences of symbolic actions and continuous controls, respectively.

In some extensions of the optimization-based TAMP problems, certain assumptions may not be satisfied. For example, in planning problem with probabilistic operators [39], the symbolic transitions are not deterministic, relaxing A1; in RL-based TAMP, a prior world model is unavailable, relaxing A2; in a TAMP framework incorporating visual input [29], the mapping function between observation and state is often learned implicitly or explicitly, relaxing A3. These variants present unique additional challenges due to the relaxation of certain assumptions. Nevertheless, the principles and patterns underscored throughout this survey retain their relevance and applicability, even in these more complex scenarios.

B. TAMP as Joint Optimization

The optimization-based TAMP problem can be viewed as a joint optimization between task planning and motion planning. The optimization at two different levels are interconnected by constraints in both decision variables and cost functions.

The task planning domain is defined as $\mathcal{D}^t$, with a set of symbolic states $\mathcal{S}$, and a set of actions $\mathcal{A}$. Each symbolic state $s \in \mathcal{S}$ is defined by the values of a fixed set of discrete variables; each action $a \in \mathcal{A}$ specifies a state transition $s_{k+1} \in \gamma(s_k, a_k)$, where $k = 1, \ldots, K$ is the index of the discrete mode of the task planner. A task planning problem is represented by a task planning domain $\mathcal{D}^t$ an initial state $s^{\text{init}} \in \mathcal{S}$, and a set of goal states $\mathcal{S}^{\text{goal}} \subseteq \mathcal{S}$. A task plan consists of a symbolic state-action sequence of length K: $\langle \mathbf{S}, \mathbf{A} \rangle = \langle s_0, a_0, s_1, a_1, \ldots, a_{K-1}, s_K \rangle$, where $s_0 = s^{\text{init}}$, $s_K \in \mathcal{S}^{\text{goal}}$.

The motion planning domain is defined as $\mathcal{D}^m$. The continuous robot state at the tth knot point of the trajectory is represented by $\mathbf{x}_t = [\mathbf{q}_t, \dot{\mathbf{q}}_t] \in \mathbb{R}^{2n}$, where $\mathbf{q}_t, \dot{\mathbf{q}}_t \in \mathbb{R}^n$ represent the generalized configuration and velocity of the robot. The control input is $\mathbf{u}_t \in \mathbb{R}^m$. The discretized dynamics of the robot is denoted as $\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t)$. The cost function at time t is $L(\mathbf{x}_t, \mathbf{u}_t) \to \mathbb{R}$, which maps the state-control pair $\langle \mathbf{x}_t, \mathbf{u}_t \rangle$ to a real number. In addition, $\langle \mathbf{x}_t, \mathbf{u}_t \rangle$ is constrained by various factors, such as joint limits, torque limits, and robot collision. These constraints are denoted as $g(\mathbf{x}_t, \mathbf{u}_t) \leq \mathbf{0}$.

The planning domain of the TAMP problem is jointly defined by the task planning domain $\mathcal{D}^t$ and the motion planning domain $\mathcal{D}^m$. Each symbolic state $s \in \mathcal{S}$ represents a manifold $\mathcal{X}^s$ in the continuous state space, which is specified by the state mapping function $M : \mathcal{X}^s = M(s)$. A symbolic state transition $\langle s_k, a, s_{k+1} \rangle$ corresponds to a continuous trajectory representing the robot motion: $\langle \mathbf{X}_k, \mathbf{U}_k \rangle = \langle \mathbf{x}_{k,0}, \mathbf{u}_{k,0}, \mathbf{x}_{k,1}, \mathbf{u}_{k,1}, \ldots, \mathbf{u}_{k,T_k-1}, \mathbf{x}_{k,T_k} \rangle$. To achieve the symbolic state transition, the entire trajectory must lie within the manifold indexed by s_k: $\mathbf{x}_t \in \mathcal{X}^{s_k}\ \forall\, t \in [0, T_k]$, while the final state of the kth trajectory should lie on the intersection between the manifolds indexed by s_k and s_{k+1}: $\mathbf{x}_{T_k} \in \mathcal{X}^{s_k} \cap \mathcal{X}^{s_{k+1}}$ and trigger the mode transition.

Given the planning domains $\langle \mathcal{D}^t, \mathcal{D}^m \rangle$, the initial states $\langle s^{\text{init}}, \mathbf{x}_0^{\text{init}} \rangle$ and goal states $\langle \mathcal{S}^{\text{goal}}, \mathbf{x}_K^{\text{goal}} \rangle$, the optimization-based TAMP problem is formulated as a joint optimization of the task-level decisions and the motion-level trajectory segments

$$\min_{\langle \mathbf{S}, \mathbf{A}, \mathbf{X}_{1:k}, \mathbf{U}_{1:k} \rangle} \sum_{k=0}^{K-1} \sum_{t=0}^{T_k-1} L_{\text{path}}(\mathbf{x}_{k,t}, \mathbf{u}_{k,t}) + L_{\text{goal}}(\mathbf{x}_{k,T_k})$$

$$\text{s.t.} \quad s_0 = s^{\text{init}},\ s_K \in \mathcal{S}^{\text{goal}} \tag{1a}$$

$$\forall k \in \{1, \ldots, K-1\} \quad \forall t \in \{0, \ldots, T_k - 1\}$$

$$a_k \in \mathcal{A},\ s_{k+1} = \gamma(s_k, a_k) \tag{1b}$$

$$\mathbf{x}_{k,t+1} = f(\mathbf{x}_{k,t}, \mathbf{u}_{k,t}) \tag{1c}$$

$$\mathbf{x}_{k,0} = \mathbf{x}_k^{\text{init}},\quad \mathbf{x}_{k,T_k} = \mathbf{x}_k^{\text{goal}} \tag{1d}$$

$$\mathbf{x}_{k,t} \in \mathcal{X}^{s_k},\quad \mathbf{x}_{k,T_k} \in \mathcal{X}^{s_k} \cap \mathcal{X}^{s_{k+1}} \tag{1e}$$

$$g_k(\mathbf{x}_{k,t}, \mathbf{u}_{k,t}) \leq \mathbf{0}, h_k(\mathbf{x}_{k,t}, \mathbf{u}_{k,t}) = \mathbf{0}. \tag{1f}$$

In this formulation, task planning and motion planning inform each other as they contain different subsets of the TAMP problem. Symbolic states in TAMP correspond to manifold constraints in the continuous domain, while symbolic actions define transitions and constraints for motion planning. The sequence of actions, or the plan skeleton, guides the trajectory planning process by defining the sequence of mode transitions to be achieved. Conversely, motion planning informs task planning by providing geometric information, action feasibility, and cost evaluations, ensuring that task decisions are realizable at the motion level.

C. Motivating Example

We introduce a tabletop manipulation task as a representative example to illustrate the formulations and algorithms discussed in this survey. As illustrated in Fig 4, the task involves a robot manipulator, denoted as R, and three distinct movable objects labeled as A, B, C. The primary objective of this task is to stack

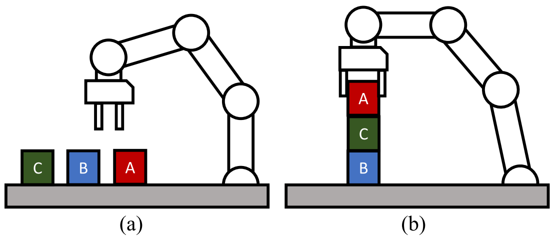


Fig. 4. Illustration for the table top manipulation example: (a) initial state and (b) one possible final state.

the objects such that the final height of object A is maximized. In addition, the robot should exert a minimal amount of control effort to achieve this task.

To quantitatively evaluate the performance of the manipulator in executing the task, we define an objective function. This function encompasses two distinct components: the path cost, which quantifies the control effort exerted by the robot throughout the task execution; the terminal cost, which measures the final elevation achieved by object A at the conclusion of the task. The objective function is then constructed as a weighted sum of these individual costs, providing a holistic measure of the task's efficiency and effectiveness.

Throughout the survey, we will enrich the initial tabletop manipulation task with various extensions to demonstrate the practical considerations of the discussed algorithms and formulations.

III. Planning Domain Representation

In real-world scenarios, planning domain representation demands formulating declarative knowledge about environments, robots, objects, their interrelationships, and task goals, alongside integrating continuous-domain knowledge, such as robot configurations and object placement. Converting this knowledge into an optimization-based formulation requires a standardized interface, which ensures seamless integration by transforming varied input knowledge into an encoded form that can be effectively utilized by optimization algorithms. Therefore, this interface bridges the gap between real-world complexities and optimization-based TAMP.

Traditional methods for planning domain representation have been adopted from both the AI planning and temporal logic communities [23]. These methods generally involve the use of logic. Section III-A presents AI planning techniques that often employ domain-independent action description languages, such as the planning domain definition language (PDDL), which are widely interfaced with state-of-the-art task planners. Temporal logic approaches (see Section III-B), utilizing formalisms, such as linear temporal logic (LTL) [40], [41], signal temporal logic (STL) [42], and metric temporal logic (MTL) [43], have been extensively used to express time-dependent behaviors and constraints.

One drawback of these traditional logic-based formalisms is that the domain representations are generally hand-specified by expert users. Therefore, a recent trend is to use learning-based methods to automatically encode domain knowledge for TAMP (see Sections III-C and D). These methods include the learning of symbolic operators, which can model the preconditions and effects of actions based on previous experiences. Furthermore, LLMs have been explored to process and interpret natural language inputs, providing a novel method for encoding the planning domain in a more intuitive and accessible format.

A. AI Planning

The task planning problem with discrete planning domains has long been the focus of the AI planning community. Ghallab et al. [22] provided a comprehensive discussion of task planning representations and algorithms in the AI planning perspective.

PDDL [44], [45] is a standard language extensively used in the AI planning community for encoding a task planning problem. It offers a compact and domain-independent syntax that aids in the clear delineation and representation of the task planning problem. An action $a \in \mathcal{A}(\mathcal{S})$ in PDDL is expressed as a tuple consisting of five components: $\langle \texttt{name}(a), \texttt{param}(a), \texttt{pre}(a), \texttt{eff}(a), \texttt{cost}(a) \rangle$ as follows.

1) `name`: name of the action.
2) `param`: discrete and continuous parameters involved to evaluate $\texttt{pre}(a)$ and $\texttt{eff}(a)$.
3) `pre`: a set of predicates that represent a set of facts that must be satisfied before the action can be applied.
4) `eff`: a set of predicates that represent a set of facts that must be satisfied after the action is applied.
5) `cost`: cost of the action represented by a positive scalar.

Classically, PDDL only supports a deterministic, discrete, and nontemporal world model [45]. Historically speaking, multiple versions and extensions of PDDL have been developed to improve its expressiveness. Numerical expressions, plan metrics, and temporal planning are introduced in PDDL2.1 [46]. The latest official version is PDDL3.1 [47], which includes more elements of modern planning problems, such as state-trajectory constraints, soft constraints, and object-fluents. Among the PDDL extensions, hybrid system planning is handled in PDDL+ [48]; probabilistic operators are introduced in PPDDL [49]; and multiagent planning is included in MA-PDDL [50].

Within the context of TAMP, PDDL undergoes certain modifications to accommodate the inherent complexity of the domain. For robotics problems, additional continuous parameters are often introduced into the planning domain. The values of predicates in both the precondition, $\texttt{pre}(a)$, and the effect, $\texttt{eff}(a)$, can be functions of these continuous variables, such as robot poses or the continuous trajectory taken by the robot. Furthermore, the cost of an action may be defined as a function of the continuous trajectory. This expanded use of PDDL allows for a more detailed and nuanced representation of planning problems, enabling the bridging of symbolic task planning and continuous motion planning.

Example: The block stacking example can be represented by a hybrid AI planning `stacking` domain with five distinct actions, as seen in Fig. 5. The `move-holding` and `move-free` actions allows the manipulator to move along a collision free trajectory with or without holding an object. The `pick` action

```
:domain stacking
:action move-free
    :param (r - robot, $q,q'$ - robot-state,
        $\tau$ - robot-trajectory,$u$ - control)
    :pre (and hand-empty(r), collision-free(r,$\tau$),
        valid-motion($q,\tau,u,q'$))
    :eff robot-state = $q'$
    :cost $L_{\text{path}}(\tau,u)$
:action move-hold
    :param (r - robot, o - object,
        $q,q'$ - robot-state, $\tau$ - robot-trajectory,
        $u$ - control, $\phi$ - object-trajectory)
    :pre (and holding(r,o),
        collision-free(r,$\tau$), collision-free(o,$\phi$),
        valid-motion($q,\tau,u,q'$), kinematics(r,o,$\tau,\phi$))
    :eff robot-state = $q'$
    :cost $L_{\text{path}}(\tau,u)$
:action pick
    :param (r - robot, o - object,
        $q$ - robot-state, $p$ - object-state)
    :pre (and hand-empty(r), clear(o), ontable(o),
        kinematics(r,o,$q,p$))
    :eff (and (not hand-empty(r)), holding(r,o),
        (not ontable(o)))
:action place
    :param (r - robot, o - object,
        $q$ - robot-state, $p$ - object-state)
    :pre (and holding(r,o), stable(o,$p$),
        kinematics(r,o,$q,p$))
    :eff (and hand-empty(r), (not holding(r, o)),
        ontable(o), clear(o))
:action stack
    :param (r - robot, x - object, y - object,
        $q$ - robot-state, $p_x$ - object-state, $p_y$ -
            object-state)
    :pre (and holding(r,x), kinematics(r,x,$q,p$),
        stable-stack(x,y,$p_x,p_y$))
    :eff (and hand-empty(r), (not holding(r,o)),
        on(x,y), (not clear(y)), clear(x))
```

Fig. 5. Hybrid block stacking problem expressed in a PDDL-style action description language.

allows the robot to pick up an object that is on the table. The `place` action allows that robot to place the object it is currently holding onto the table. The `stack` action allows the robot to stack one object on top of another. Note that the actions are represented in a templated manner, which provides a compact representation of the planning problem. At the planning time, the actions are instantiated into grounded representations associated with specific robots and objects.

B. Temporal Logic

Temporal logic formalism provides concise expressions for temporal relations between symbolic expressions. One of the most popular classes of temporal logic in robotic applications is LTL [40], [41], which assumes a linear sequence of event, as opposed to the more complex nonlinear temporal logic (e.g., computation tree logic [51]). The syntax of LTL contains a set of propositional variables AP, boolean operators $\neg$ (negation), $\wedge$ (conjunction), $\vee$ (disjunction), and a collection of temporal operators. The most common temporal operators are as follows:

1) `eventually` $\Diamond\varphi$: φ will hold true at some point in the future;
2) `next` $\bigcirc\varphi$: φ is true at the next time step;
3) `always` $\Box\varphi$: φ has to be true for the entire path;
4) `until` $\varphi_1\mathcal{U}\varphi_2$: φ_1 has to hold true at least until φ_2 becomes true;
5) `release` $\varphi_1\mathcal{R}\varphi_2$: φ_2 holds true until φ_1 becomes true.

One limitation of LTL formula is that only boolean variables and discrete time evaluation is allowed. Several extensions of LTL have been proposed to enable real-time and real-valued expressions. MTL [43] extends LTL to real-time applications by allowing timing constraints. STL [42] further extends MTL to allow formula evaluation over continuous real-valued signals, which enrich the temporal logic formalism to specify hybrid planning problems in TAMP.

For STL, let $\mathbf{y}:\mathbb{R}_{\geq 0}\to\mathbb{R}^n$ be a signal and $t\in\mathbb{R}_{\geq 0}$ be a time. Let $(\mathbf{y},t):=(\mathbf{y},[t,\infty))$ denote the suffix of the signal. Let π represent an atomic predicate of the form $\mu^\pi(\mathbf{y})\geq 0$. The satisfaction of an STL formula φ at time t for signal $\mathbf{y}$ is defined as follows:

1) $(\mathbf{y},t)\models\pi \iff \mu^\pi(\mathbf{y}(t))\geq 0$;
2) $(\mathbf{y},t)\models\neg\varphi \iff (\mathbf{y},t)\not\models\varphi$;
3) $(\mathbf{y},t)\models\varphi_1\wedge\varphi_2 \iff (\mathbf{y},t)\models\varphi_1$ and $(\mathbf{y},t)\models\varphi_2$;
4) $(\mathbf{y},t)\models\varphi_1\vee\varphi_2 \iff (\mathbf{y},t)\models\varphi_1$ or $(\mathbf{y},t)\models\varphi_2$;
5) $(\mathbf{y},t)\models\Box_{[t_1,t_2]}\varphi \iff \forall t'\in[t_1,t_2],(\mathbf{y},t')\models\varphi$;
6) $(\mathbf{y},t)\models\Diamond_{[t_1,t_2]}\varphi \iff \exists t'\in[t_1,t_2],(\mathbf{y},t')\models\varphi$;
7) $(\mathbf{y},t)\models\varphi_1\mathcal{U}_{[t_1,t_2]}\varphi_2 \iff \exists t'\in[t_1,t_2],(\mathbf{y},t')\models\varphi_2\wedge\forall t''\in[t_1,t'],(\mathbf{y},t'')\models\varphi_1$.

The robustness degree $\rho(\mathbf{y},\varphi,t)$ of STL is often used to quantify how well a given signal satisfies or violates an STL specification. The mathematical definition of robustness degree can be found in [52].

Encoding STL formula as mixed-integer constraints: The STL specification can be encoded into mixed-integer constraints using the big-M method [53]. The overall idea is that for each predicate π, a binary variable z_t^π is created at time t, where 1 corresponds to true and 0 corresponds to false. Using the big-M method, the robustness degree ρ can be represented with the inequality

$$\mu^\pi(\mathbf{y}(t))-M_t(1-z_t)\geq\epsilon_t,\mu^\pi(\mathbf{y}(t))-M_tz_t\leq\epsilon_t \quad (2)$$

where M_t is a sufficiently large constant for all predicates at time t, $M_t\geq\max_\pi\mu^\pi(\mathbf{y}(t))$, and ϵ_t is a sufficiently small positive constant that bounds $\mu^\pi(\mathbf{y}(t))$ away from 0. Using the big-M method, the boolean operations, such as disjunction and conjunction are represented by the following:

$$z=\bigwedge_{i=1}^{n_z}z_i\Longrightarrow z\leq z_i,\quad i=1,\ldots,n_z \quad (3)$$

$$z=\bigvee_{i=1}^{n_z}z_i\Longrightarrow z\geq\sum_{i=1}^{n_z}z_i. \quad (4)$$

Kurtz and Lin [54] proposed a tree structure for STL formulas, resulting in a more efficient encoding that uses fewer binary variables. In comparison to the big-M method, the smoothed approximation approaches, introduced in Section VI-B, represent the STL specifications as continuous constraints via the robustness degrees.

Example: To express the block stacking problem in STL, we first define the signals and predicates and then express the planning domain using STL formulas. The continuous states and controls in the planning domain are represented as signals

The following predicates can be generated from the signals to express the state of the objects.

- H_A, H_B, H_C: whether the manipulator is holding an object A, B, or C, $\|\mathrm{FK}(\mathbf{q}(t)) - \mathbf{p}_A\| = 0 \wedge g(t) \le 0.5 \models H_A$, similarly for H_B and H_C;
- $S_{AB}, S_{AC}, S_{BC}, S_{BA}, S_{CA}, S_{CB}$: whether an object is stacked on top of another. For example, S_{AB} signifies A is stacked directly on top of B, i.e., $\big((\mathbf{p}_{A,x}(t) = \mathbf{p}_{B,x}(t)) \wedge (\mathbf{p}_{A,y}(t) = \mathbf{p}_{B,y}(t)) \wedge (\mathbf{p}_{A,z}(t) = \mathbf{p}_{B,z}(t) + h_A)\big) \models S_{AB}$, where h_A is the the height of the object.

The STL formulation for the blocking domain is then represented in the following formulas:

- Robot can hold only one object at a time: $\square_{[0,T]}(H_A \Rightarrow \neg H_B \wedge \neg H_C) \wedge (H_B \Rightarrow \neg H_A \wedge \neg H_C) \wedge (H_C \Rightarrow \neg H_A \wedge \neg H_B)$;
- Objects cannot be free-floating: $\square_{[0,T]}(\mathbf{p}_{A,z}(t) > 0 \Rightarrow H_A \vee S_{AB} \vee S_{AC}) \wedge (\mathbf{p}_{B,z}(t) > 0 \Rightarrow H_B \vee S_{BA} \vee S_{BC}) \wedge (\mathbf{p}_{C,z}(t) > 0 \Rightarrow H_C \vee S_{CA} \vee S_{CB})$

Fig. 6. Tabletop manipulation example expressed in STL.

in STL. We define signals $\mathbf{p}_A(t), \mathbf{p}_B(t), \mathbf{p}_C(t)$ to be the 3-D positions of objects A, B, C, and signal $\mathbf{y}(t) = [\mathbf{q}(t); \mathbf{u}(t)]$ to be the joint angles and torques of the robot manipulator at time t. In addition, the gripper state is represented by signal $g(t) \in \{0, 1\}$, where 0 means the gripper is closed and 1 means it is opened. Note that, STL has to instantiate each object individually, which is different from the templated representation in PDDL. The following example in Fig. 6 provides one instantiation for each type of predicate.

C. Learning Operators and State Abstractions

To facilitate the search for task plan in solving TAMP problems, researchers propose learning symbolic operators, where probabilistic transition models are evaluated. In addition, learning state abstractions studies the intrinsic structure of the task, such as hierarchical structure and object importance, in order to help decompose the large search space into two or more levels of abstractions.

Learning operators: Silver et al. [39] proposed to learn symbolic operators for TAMP using a relational learning method, where the demonstration data is first converted to symbolic transitions with defined predicates, and then the effects and preconditions are discovered by grouping transitions with similar effects. To alleviate the burden of hand-engineered symbolic predicates, Silver et al. [55] further proposed to learn the symbolic predicates and the operators jointly from the demonstration data by optimizing a surrogate objective that relates to planning efficiency. To improve the generalization over novel objects, Chitnis et al. [56] introduced neuro-symbolic relational transition models, where high-level planning is achieved through symbolic search, and the learned action sampler and transition models are used to generate continuous motion.

Learning state abstractions: State abstractions have also been studied to further improve the efficiency and generalization of TAMP systems. Chitnis et al. [57] introduced a method for acquiring context-specific state abstractions. This approach focuses on considering only task-relevant objects, streamlining the planning process and improving adaptability across different scenarios, Silver et al. [58] developed a graph neural network (GNN)-based framework to predict object importance, thus allowing the planner to efficiently search for a solution while only considering the objects that are relevant to the task goal. Zhu et al. [59] proposed a hierarchical framework that constructs the symbolic scene graph and geometric scene graph from visual observations for representing the states, which are used for generating task plans and motion plans. Wang et al. [60] suggested utilizing extensive datasets to enhance generalization. They adopt a two-step approach, commencing with the pretraining of visual features through symbolic prediction tasks and semantic reconstruction tasks. Subsequently, they employ the latent feature derived from this pretraining to learn abstract transition models, which in turn aid in guiding the task plan search process.

D. Generating Domain Knowledge by LLMs

Generating domain knowledge for planning methods, including *action descriptions* and *goal specifications*, typically requires manual input from human experts using specific declarative languages like PDDL. Manually encoding action description knowledge for task planners can be a tedious process. It requires extensive domain knowledge from human experts and must be regularly maintained to adapt to domain changes. It is a long-standing challenge of generating domain knowledge for autonomous agents (including robots) with minimum human involvement. Recent advances in LLMs have demonstrated the great potential of automating this process across various planning scenarios.

Generating action description by LLMs: The strategy for generating action descriptions can be divided into two categories. The first involves LLMs revising existing action descriptions to adapt them to different domains and situations. For instance, Ding et al. dynamically enrich original domain knowledge with task-oriented commonsense knowledge extracted from LLMs [61]. The second category involves LLMs directly creating new action descriptions for planning. Here, researchers may employ various prompting methods to enhance generation performance. Examples of such methods include specifying detailed prompts that guide the generative model toward producing outputs that are more aligned with the desired outcome [31], [62], and integrating structured data through programming languages to provide a clear context or framework for the generation [63]. A major challenge in this area is ensuring the practicality of these generated descriptions in real planning systems, given the variability of LLM outputs. To address this, researchers deploy various evaluation methods, including simulations [64], comparison against predefined actions [32], or human assessments, to filter the most viable outcomes [65].

Generating goal description with LLMs: Existing studies aim to translate objectives stated in natural language into specific formats, such as PDDL [62], [66] or LTL [67]. This process is challenging, requiring understanding of context, adherence to syntax to avoid errors that may lead to failure, and alignment with the particular domains and problems. Like action description

Problem Description:
The problem description is: You have 3 blocks. B is on top of C. C is on top of A. A is on the table. B is clear. Your arm is empty. Your goal is to move the blocks, so that the final height of block A is as high as possible. Provide me with the problem PDDL file that describes the planning problem directly without further explanations.

Problem File:

```
(define (problem blocks-problem)
(:domain blocks)
(:objects A B C - block)
(:init (clear B) (on B C) (on C A) (on-table A) (arm-
empty) (= (height A) 0) (= (height B) 0) (= (height C)
0) )
(:metric (height A)) )
```

Fig. 7. Tabletop manipulation example expressed in LLM prompt.

generation, this requires various prompting methods, such as including examples in the prompts [62], [66]. Learning methods are also used to translate natural language directives into LTL specifications, with minimal need for human-annotated training data [67]. Unlike others, AutoTAMP [68] employs LLMs to translate task requirements into goals, which can be applicable at both task and motion levels. Fig. 7 illustrates an example of translating natural language into goal specification.

IV. Task Planning

Task planning focuses on determining sequences of actions to achieve specific goals using symbolic method. Traditionally, classical AI planning methods address this using graph search algorithms with specialized heuristics. Alternatively, temporal-logic-based techniques, especially those using LTL, employ automata theory and reactive synthesis to generate discrete decision sequences. However, the aforementioned classical AI planning and temporal-logic-based approaches are not without limitations. One major challenge is the combinatorial complexity that arises when dealing with large-scale planning problems. This complexity can severely hamper the scalability and efficiency of planning algorithms.

To address these challenges, recent advancements in the field aim to bypass the combinatorial bottleneck by leveraging learned models to guide the task sequence search. These approaches utilize insights from learned models, incorporating task decompositions, action affordances, and the effects of skills. Notably, the advent of LLMs has introduced new methodologies. LLM-native planning derives strategies directly from data, while LLM-aided techniques synergize these models with established planning systems. The fusion of classic algorithms with state-of-the-art machine learning encourages a promising evolution in task planning algorithms.

A. Classical Task Planning

Classical task planning, as described by Ghallab et al. [22], refers to the problem of planning for a deterministic, static, finite, and fully observable state-transition system with restricted goals and implicit time. The most straightforward task planning algorithms are state-space search methods. In this paradigm, the search space is a subset of the state-space itself, where each node in the search represents a state, and each edge symbolizes a transition. The state-space search typically results in a sequential path traversing the state space, effectively detailing the progression from an initial state to a goal state. State-space search is particularly relevant to the field of TAMP, as the underlying motion planning algorithm inherently operates on state space. The key considerations for algorithm design include the identification of appropriate search space, the selection of efficient algorithms, and the determination of suitable heuristics to guide the search process.

The search heuristics in classical AI planning can be seen as the relaxation of the exact search problem. In practice, the heuristics design often involves a tradeoff between computational cost and informativeness of the heuristics. The works in [69] and [70] employ heuristics based on the idea of state reachability relaxation, where the heuristics are computed by constructing a relaxed planning graph starting at state s, and all negative effects of operators are ignored when growing the graph. Therefore, the resulting planning graph has the properties of monotonic increase in the number of propositions with respect to the depth of the graph. A simple, computationally-cheap heuristics based on the relaxed planning graph is the goal distance function [71]. Let the distance to goal $h^*(s)$ be defined as the minimum number of operators needed to reach the goal. The lower bound estimation of $h^*(s)$ can be easily calculated by the minimum depth of the node containing all the goal propositions within the relaxed planning graph. As an alternative approach, the fast downward-based [72], [73] planning systems uses hierarchical decomposition of planning tasks to compute a causal graph heuristic, which uses the causal dependencies in a relaxed causal graph to guide the forward state-space search.

In comparison to state-space search, other AI planning methods, such as hierarchical task network [74], attempt to conduct search on plan-space. However, these methods are not often used in TAMP scenarios due to the difficulty in interfacing plan-space search with motion planners.

For temporal logic-based formulations, such as LTL, automata-based approaches [75], such as reactive synthesis [76], [77], [78], are often used to generate a reactive system that ensures the system meets a desired specification irrespective of external inputs.

Note that this survey assumes the readers have basic backgrounds of classical task planning and intentionally keeps this section brief. For more information, readers are referred to [22], [23], and [37].

B. Learning Models for Task Planning

A key challenge of improving the scalability of TAMP is the combinatorial complexity of the discrete planning problem and the large number of motion planning problems to be solved. A promising approach to circumvent this challenge is to use learning methods to guide the high-level task plan search. Pasula et al. [79] proposed to learn probabilistic, relational planning rule representations to model the action effects, which can be used to generate the task plan through search. Similarly, Amir and Chang [80] developed a method that learns the deterministic action models in partially observable domains. To allow dealing with uncertain representation and probabilistic plans, Konidaris et al. [81] proposed to replace the sets and logical operations by probability distributions and probabilistic operations, and develop a framework that enables autonomous learning of the probabilistic symbols from continuous environments. To address the challenge of goal-directed planning involving a set of predefined motor skills, Konidaris et al. [82] presented a framework that directly acquires symbolic representations, abstracting the low-level transitions for effective utilization in planning tasks.

More recently, deep learning techniques have been explored to learn the models from large-scale datasets. Ames et al. [83] proposed to learn preconditions, action parameters, and effects from execution results of parameterized motor skills, which are then used to construct symbolic models for efficient planning. Neural task programming [84] proposes to learn neural models that recursively decompose a task demonstration video into robot executable action primitives. To further improve the generalization on long-horizon tasks, neural task graphs [85] learns neural networks for generating conjugate task graphs, where the actions are represented as nodes and the dependencies between actions are modeled by edges, better exploring the compositionality. Regression planning networks [86] learns a neural model to iteratively predict the intermediate subgoals in a reverse order based on the current image observation and the final symbolic goal. Ceola et al. [87] proposed to utilize deep RL to train neural models for generating discrete actions. Deep affordance foresight [88] learns the long-term affordance of actions and the latent transition models to guide the search, and thereby informs the robot of the best actions to achieve the final task goal. Similarly, Liang et al. [89] proposed to learn skill effect models that generate future terminal states of each parameterized skill, and then leverage these models to aid search-based task planning.

C. LLMs for Task Planning

Traditionally, optimizing task plans for robots involves minimizing either the number of actions or the total plan cost, depending on whether action costs are considered. The emergence of LLMs, such as Google's Bard, OpenAI's ChatGPT [90], and Meta's LLaMA [91], have reshaped the landscape of AI, including task planning for robots [92]. We categorize the LLM-based planning methods into the two groups: *LLM-native planning methods* and *LLM-aided planning methods*, where the former does not rely on external knowledge and the latter does, as shown in Fig. 8. Comparing to regular learning-based methods, LLMs are typically trained on a large amount of out-of-domain data that contains a great deal of commonsense knowledge. While LLMs are not strong in numerical reasoning (and hence optimization) [93], [94], the incorporation of LLMs improves the capabilities of natural language understanding, the acquisition of world knowledge, and commonsense reasoning. Such capabilities enable LLM-based planners to reason about symbolic information, such as spatial relationships between objects [95]

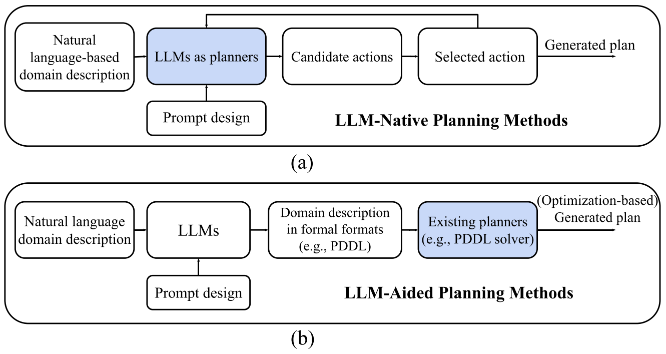

Natural language-based domain description
LLMs as planners
Candidate actions
Selected action
Generated plan
Prompt design
LLM-Native Planning Methods
(a)
Natural language domain description
LLMs
Domain description in formal formats (e.g., PDDL)
Existing planners (e.g., PDDL solver)
(Optimization-based) Generated plan
Prompt design
LLM-Aided Planning Methods
(b)

Fig. 8. Two methodologies for task planning using LLMs, with the key difference lies in the role of LLMs: (a) LLM-native planning methods use LLMs for planning, while (b) LLM-aided planning methods LLMs to generate domain descriptions for existing task planning methods, such as PDDL and temporal logic.

and symbolic correctness of a task sequence [30], without prior interaction with the robot environment. Therefore, LLMs as a task and domain agnostic reasoning module has the potential to enhance the scalability and generalizability of robot planning.

LLM-native planning methods often incorporate additional components like RL to enhance planning by choosing better actions. Conversely, LLM-aided planning methods can be integrated with classical optimization strategies, ensuring satisfactory planning efficiency and practicality. These two approaches are compatible with optimization methods, while integrating LLMs enhances the overall planning capabilities.

LLM-native planning methods: One method to leverage LLMs for task planning involves directly generating plans from LLMs by providing a domain description [see Fig. 8(a)]. This can be done either in a one-shot way or iteratively. These methods primarily focus on prompt design for effective communication with LLMs, and the grounding to specific domains and robot skills. Multiple systems have made efforts in this field. Huang et al. [32] proposed to generate candidate actions and design tools to improve their executability, such as enumerating all permissible actions and mapping the model's output to the most semantically similar action. Building upon this, SayCan [96] enables robotic planning using affordance functions that determine action feasibility and respond to natural language requests, such as "deliver a Coke." An advanced approach, named Inner Monologue, developed by Huang et al. [97], integrated environmental feedback for task planning and situation handling. Previously, methods typically generate task plans in text forms. Singh et al. [63] developed a system, called ProgPrompt, which employs programmatic LLM prompts to generate task plans and manage situations, by verifying the preconditions of the plan and reacting to failed assertions with suitable recovery actions. Employing code as the framework for high-level planning provides significant benefits. It allows for the expression of complex functions and feedback loops. These loops effectively process sensory outputs and enable the parameterization of control primitives within application programming interface (APIs) [98]. Apart from planning for robots, there is also research on whether LLMs can act as universal planners. They could potentially create programs that efficiently generate plans for various tasks within the same domain [31]

LLM-aided planning methods: Prior to the development of LLMs, various tools existed for robot task planning, but they had scalability limitations. For example, defining domain knowledge in PDDL demands significant time from human experts [see Fig. 8(b)]. The advent of LLMs offers a way to augment these traditional planners by supplementing knowledge, thereby improving their performance and enabling more natural language interaction. There are a few ways of integrating LLMs and classical task planners. First, a series of studies are conducted to explore the conversion of natural language descriptions of planning tasks into standardized languages like PDDL or temporal logic. LLMs complete these transformations, playing a crucial role in the process. These translated specifications are then used in existing planning systems. For example, Xie et al. [66] created optimality-based task-level plans with the PDDL planner, translating natural language inputs into PDDL problems. Second, one can dynamically extract commonsense knowledge from LLMs, enhancing PDDL's action knowledge for planning and situational handling [95]. Third, Zhao et al. [64] utilized LLMs to build world models, and perform heuristic policy in search algorithms, such as Monte Carlo tree search, which uses the common-sense knowledge provided by LLMs to generate possible world states, thus facilitating efficient decision-making as well as underlying motion planning.

The optimization of those LLM-based planning methods occurs in the interaction with the LLMs, in the plan generation of classical task planners, or both. The prompting strategy of LLM-native planning methods encourage behaviors toward maximizing the overall task completion rate, where the optimization usually occurs in an implicit way (i.e., there is no objective function explicitly specified). By comparison, the LLM-Aided planning methods compute optimal plans with or without plan cost in consideration, where the optimality is conditioned on the external knowledge provided by LLMs, and the optimization process is embedded within the deployed task planning system.

V. Optimization-Based Motion Planning

Optimization-based motion planning is an important component in robot planning. It aims to generate a continuous robot motion path and a control sequence that optimizes an objective function subject to a set of kinematics and/or dynamics constraints. Numerous methods have been proposed [99] to TO[1]. Notable TO techniques include direct methods that transcribe TO into nonlinear programs (NLPs), and indirect methods that leverage the optimality conditions.

In the meantime, with the increasing complexity and diversity of environments that robots operate in and tasks that the robot are required to accomplish, there is an imperative need to enhance the scalability of these TO strategies, especially in handling robot dynamics, complex constraints in physical contact problems, and higher dimensional state spaces in multirobot scenarios. To this end, distributed optimization techniques have been introduced, with consensus alternating direction method of multipliers (ADMM) being a notable methodology [100].

In conjunction with model-based TO approaches, recent advancement in combining data-driven approaches and TO has shown capabilities in predictively generating trajectories by imitating offline-generated optimized paths solved by model-based TO techniques [101], [102]. These learned methods hold significant promise in enhancing the efficiency and adaptability of motion planning processes, especially in environments with dynamic and unforeseen challenges.

A. Trajectory Optimization

A motion planning problem is specified by a motion planning domain $\mathcal{D}^m$, an initial state $\mathbf{x}^{\text{init}} \in \mathbb{R}^{2n}$, and a goal state $\mathbf{x}^{\text{G}} \in \mathbb{R}^{2n}$. A motion plan consists of a state-control trajectory with T knot points: $\langle \mathbf{X}, \mathbf{U} \rangle = \langle \mathbf{x}_0, \mathbf{u}_0, \mathbf{x}_1, \mathbf{u}_1, \ldots, \mathbf{u}_{T-1}, \mathbf{x}_T \rangle$, where $\mathbf{x}_0 = \mathbf{x}^{\text{init}}$ and $\mathbf{x}_T = \mathbf{x}^{\text{G}}$.

The motion planning problem can be formulated as a constrained NLP

$$\min_{\mathbf{X},\mathbf{U}} \sum_{t=0}^{T-1} L_{\text{path}}(\mathbf{x}_t, \mathbf{u}_t) + L_{\text{goal}}(\mathbf{x}_T) \tag{5a}$$

$$\text{s.t.} \quad \mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t) \tag{5b}$$

$$\mathbf{x}_0 = \mathbf{x}^{\text{init}}, \ \mathbf{x}_T = \mathbf{x}^{\text{G}} \tag{5c}$$

$$g(\mathbf{x}_t, \mathbf{u}_t) \leq \mathbf{0} \tag{5d}$$

where the dynamics equation in (5b) and inequality constraint in (5d) are defined in Section II-B.

Direct collocation [103], [104] offers a straightforward transcription where both controls and states are treated as decision variables, and complex state constraints can be easily expressed. General-purpose NLP solvers, such as IPOPT [105] and SNOPT [106], can be adopted to solve for optimal solutions. Alternatively, motivated by the real-time computation requirement for many robotics applications, researchers start to devise problem-specific solvers for reliably solving the above NLP. Notably, differential dynamic programming (DDP) [107] is a shooting method that efficiently explores the problem structure through Riccati recursion and handles nonlinear dynamics, but limited to unconstrained TO. More recently, variants of DDP algorithms have been proposed to handle diverse state and control constraints [26], [108], [109], [110], [111], [112], [113]. Readers are referred to [99] and [114] for a comprehensive overview on the numerical TO methods.The recent survey paper [24] offers insights into contemporary applications of TO in legged locomotion with an emphasis on handling complex dynamic and contact constraints.

Example: For the tabletop manipulation task, we consider the motion planning problem for a single task of a manipulator moving from a free position to pick up an object A. Let forward kinematics function $FK(\cdot)$ denote the end-effector position of the manipulator and $\mathbf{p}_A$ denote the position of object A. The running cost consists of a position tracking term and regularization

[1]In this survey, we interchangeably use the terms of "TO" and "optimization-based motion planning."

terms for state and control

$$L_{\text{path}}(\mathbf{x}_t, \mathbf{u}_t) = w\|FK(\mathbf{x}_t) - \mathbf{p}_A\|^2 + \mathbf{x}_t^\top Q\mathbf{x}_t + \mathbf{u}_t^\top R\mathbf{u}_t. \tag{6}$$

The goal cost only concerns whether the final configurations of the robot achieves the desired final end-effector position

$$L_{\text{goal}}(\mathbf{x}_T) = w\|FK(\mathbf{x}_T) - \mathbf{p}_A\|^2. \tag{7}$$

The dynamics in (5b) is represented by the numerical integration of the rigid-body dynamics equation [115]

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{u} + \mathbf{J}(\mathbf{q})\boldsymbol{\lambda} \tag{8}$$

where $\mathbf{M} \in \mathbb{R}^{n\times n}$ represents is inertia matrix; $\mathbf{C} \in \mathbb{R}^n$ is the gravitational, centrifugal, and Coriolis forces; $\mathbf{J}$ represents the Jacobian matrix; and $\boldsymbol{\lambda}$ is the contact forces at the end-effector.

The following inequality constraints are involved:

$$\textbf{state limit}: \underline{\mathbf{x}} \leq \mathbf{x}_t \leq \overline{\mathbf{x}} \tag{9}$$

$$\textbf{control limit}: \underline{\mathbf{u}} \leq \mathbf{u}_t \leq \overline{\mathbf{u}} \tag{10}$$

$$\textbf{collision avoidance}: d_i(\mathbf{x}) \geq d_{\min} \tag{11}$$

where $d_i(\mathbf{x})$ represents the distance between the ith collision pair at state $\mathbf{x}$, and $d_{\min}$ denotes the minimum allowable distance to avoid collisions.

B. Distributed Optimization

Many TO problems have intrinsically distributed structures. Such distributed structures are often formulated and solved via alternating optimization approaches, such as ADMM [100], in order to improve the efficiency of TO. However, the distributed structure might not be immediately apparent and the optimization problems often need to be reformulated into an explicitly distributed format. We focus here on the consensus ADMM as a representative distributed formulation, where copies of decision variables and additional consensus constraints are often introduced to reveal the distributed structure of the TO problems. Consider a optimization problem where the objective is the sum of N functions

$$\min_{\mathbf{X}} \quad \sum_{i}^{N} J_i(\mathbf{X}). \tag{12}$$

The optimization can be reformulated in the consensus ADMM format

$$\min_{\overline{\mathbf{X}}, \mathbf{X}_1, \ldots, \mathbf{X}_N} \quad \sum_{i}^{N} J_i(\mathbf{X})$$
$$\text{s.t.} \quad \mathbf{X}_i = \overline{\mathbf{X}} \quad \forall i \in \{1, \ldots, N\} \tag{13}$$

where $\overline{\mathbf{X}}$ is a set of global decision variables. Practically, one critical aspect to achieve a satisfactory consensus performance is built upon appropriate selection of ADMM parameters through principled mechanisms, such as over-relaxation [116], varying-penalty parameters [117], and Nestorov acceleration method [118]. In the following, we focus on the discussion of three structures that can be effeciently solved using consensus ADMM commonly seen in TO problems.

Spatial structure: The spatial structure of the system can be exploited when the subsystems and their dynamics are separable. This property often exists in multirobot systems, where the planning of each robot can be treated as a subproblem. The decision variables are possibily coupled through the objective functions or the constraints (e.g., collision avoidance between robots). Local copies of the full state variables can be created for each robot to decouple the problem [119], as seen in Fig. 9(a). Readers are referred to [38] and [120] for a detailed review of the multirobot ADMM. Robustness is further studied in [121] given a multirobot motion planning problem via ADMM. Amatucci et al. [122] accelerated TO for loco-manipulation tasks by modeling a quadruped robot with an articulated arm as three subrobots.

Temporal structure: TO problems are often formulated in their discretized form. In most cases, the discrete formulation involves a set of decoupled objective and constraint terms that are functions of robot state and control *at a single timestep* [e.g., (5c) and (5d)], and dynamics constraints that couples the states and control trajectory *across consecutive timesteps* [e.g., (5b)].

For single-timestep objectives or constraints that are computationally expensive (e.g., complementarity constraints for contact [25]), it is beneficial to accelerate the optimization process by leveraging the temporal structure of the problem and parallelizing single-timestep objectives and constraints of interest in a distributed fashion [see Fig. 9(b)].

A constraint can be decoupled in a similar fashion by moving the constraint into objective using indicator functions or projection operators. Examples include [123], where the linear complementarity constraints are independent temporally, and [124], where a $L1$ objective on the control are decoupled. Similarly, in [125] and [126], box constraints are handled separately through a projection operator.

System structure: The system structure of TO can be exploited when the system dynamics can be characterized by two or more interacting subsystems, i.e., dynamic models with different complexities. ADMM is used to separate the full optimization problem into sub-problems, each of which corresponds to a subsystem [see Fig. 9(c)]. This separation often applies to systems with complex robot dynamics with high degrees of freedom. Different from the spatial structure, the system structure often involves nonlinear mapping from one subsystem to another one, e.g., a mapping from centroidal dynamics to whole-body dynamics for locomotion problems, as introduced in the next paragraph.

In legged locomotion, there is often a hierarchy of model abstractions, where a whole-body TO and a reduced-order TO are both solved over the planning horizon [127], [128]. This hierarchy of model abstractions can be effectively handled via the dynamic splitting strategy of ADMM. The original rigid body dynamics can be split into centroidal dynamics and whole-body kinematics [129] or dynamics [130], [131]. Although the authors in [129] and [130] do not explicitly use ADMM, they iteratively feed optimized trajectory from one subsystem to the other one as the reference trajectory. Empirically, decent results

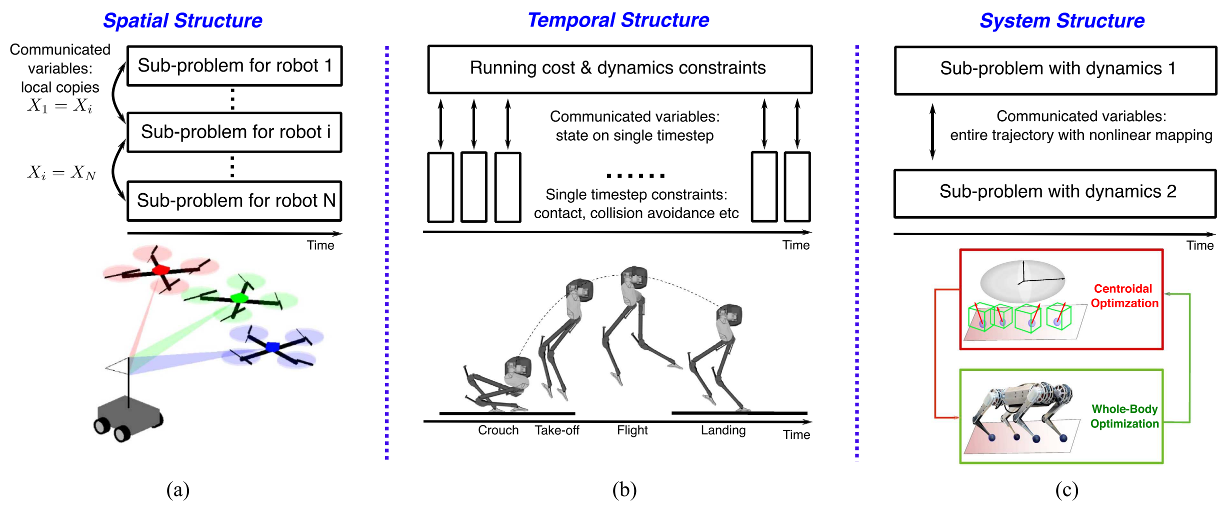

Spatial Structure
Communicated variables: local copies $X_1 = X_i$
$X_i = X_N$
Sub-problem for robot 1
Sub-problem for robot i
Sub-problem for robot N
Time
(a)
Temporal Structure
Running cost & dynamics constraints
Communicated variables: state on single timestep
Single timestep constraints: contact, collision avoidance etc
Time
Crouch
Take-off
Flight
Landing
Time
(b)
System Structure
Sub-problem with dynamics 1
Communicated variables: entire trajectory with nonlinear mapping
Sub-problem with dynamics 2
Time
Centroidal Optimzation
Whole-Body Optimization
(c)

Fig. 9. Three common structures in distributed TO: (a) spatial structure, illustrated by a multiagent drone system; (b) temporal structure, illustrated by a Cassie robot jumping over a gap; (c) system structure, illustrated by an alternating centroidal and whole-body optimization for a quadruped.

have been reported for converging to local minima [132]. A potentially accelerated ADMM updating scheme is also proposed in [133].

C. Learning Methods for Motion Planning

Despite the improvements in the efficiency of classical TO methods, it remains challenging to achieve real-time TO in many use cases. Moreover, problem-specific objectives and constraints within TO often need to be manually designed, limiting the generalizability of TO approaches. Consequently, learning methods have been extensively explored to facilitate motion generation by: 1) learning objectives and constraints to guide the TO, 2) learning physical models for integration into TO, and 3) learning end-to-end policies that imitates the trajectories generated by TO.

Learned objectives and constraints for TO: Objective functions and constraints can be learned from trajectory demonstrations and other task specification inputs, such as natural language. Guided cost learning [134] recovers cost function by adaptively sampling trajectories generated by TO using policy optimization. For constrained TO scenarios, inverse karush-kuhn-tucker (KKT) [135] learns the cost function and KKT conditions of the underlying constrained optimization problem. Janner et al. [102] proposed to view RL as a generic sequence modeling problem, and then develop a transformer-based architecture to model the distribution of the trajectories, and utilize beam search to solve the planning problem. To allow more flexible task specifications, Sharma et al. [136] proposed to learn neural networks for mapping natural language sentences to transformations of cost functions, which are then used for optimizing the motion trajectories. Along another line of research, LLM has shown promises as a interface to motion planning by describing robot motions and translating desired robot motions into reward functions [137] to guide the optimization of control policy. VoxPoser [138] leverages LLM to generate cost maps based on task specifications, and then utilizes search algorithms to derive the robot motion trajectory.

Learned physical models for TO: Complexities in physical models, especially the discontinuities in contact models can cause significant numerical challenges for TO. These challenges have spurred the development of learned differentiable contact models, despite the noted difficulty in accurately capturing the behavior of stiff contacts [139]. ContactNets [140] proposes to learn interbody distances and contact Jacobians using a smooth, implicit parameterization, which can potentially be integrated with TO. The work in [141] extends upon [140] to simultaneously learn continuous and contact dynamics using residual networks. For object manipulation problems, Cleac'h et al. [142] build a dynamic augmented neural object model that simulates the geometry and dynamics of an object as well as a differentiable contact model. Driess et al. [143] proposed to learn object representations as signed distance fields, which are particularly suitable for optimization-based planning approaches.

End-to-end policy learning guided by TO: To address the inefficiencies encountered in TO and the obstacles associated with executing TO in real-time, research efforts have been made to learn neural policies that imitates the trajectory examples generated by offline TO. Guided policy search [144], [145] iteratively trains policy on distributions over guiding samples generated by DDP. In comparison, the works in [101], [146], and [147] propose to use ADMM to achieve consensus between neural network policy and trajectory examples provided by TO. OracleNet [148] recovers the motion plans sequentially with learned recurrent neural networks. To address the motion planning problems with task constraints, CoMPNet [149] first encodes the task descriptions and environment into latent space, with a recurrent neural network and CNNs, and then sequentially generates the intermediate robot configurations based on the feature embedding, initial configuration, and goal configuration. Similarly, Radosavovic et al. [150] developed a transformer-based framework for tacking the humanoid locomotion task, where the model is first trained in simulation for generating actions in an autoregressive way, and directly deployed in the real world. To handle the multimodal action distribution of

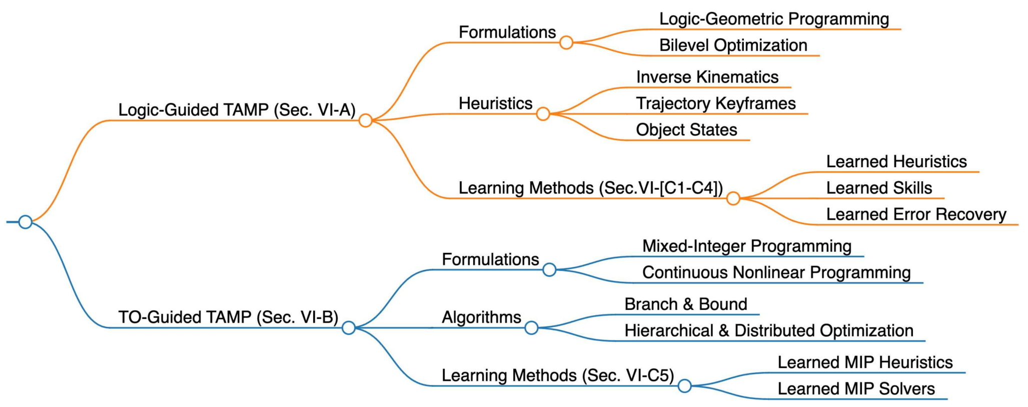

Logic-Guided TAMP (Sec. VI-A)
Formulations
Logic-Geometric Programming
Bilevel Optimization
Heuristics
Inverse Kinematics
Trajectory Keyframes
Object States
Learning Methods (Sec.VI-[C1-C4])
Learned Heuristics
Learned Skills
Learned Error Recovery
TO-Guided TAMP (Sec. VI-B)
Formulations
Mixed-Integer Programming
Continuous Nonlinear Programming
Algorithms
Branch & Bound
Hierarchical & Distributed Optimization
Learning Methods (Sec. VI-C5)
Learned MIP Heuristics
Learned MIP Solvers

Fig. 10. Organization and main topics of Integrated TAMP in Section VI. Two main types of optimization-based TAMP are introduced: *logic-guided TAMP* and *TO-guided TAMP*. For each type, the formulations, considerations in classical methods, and relevant learning methods are discussed.

TABLE II
OVERVIEW OF CLASSICAL APPROACHES FOR OPTIMIZATION-BASED TAMP

Papers	Formulation	Mathematical Program	Dynamics/Kinematics	Applications
[16]	LGP	STRIPS + NLP	Cartesian dynamics	tabletop manipulation
[8]	LGP	MCTS + KOMO	joint dynamics	tabletop manipulation
[155], [15]	LGP	MBTS + KOMO	joint dynamics	tabletop manipulation
[125]	BO	A* + DDP	joint dynamics	tabletop manipulation
[156]	PDDL or ASP	Heuristics Search + RRT	kinematics	navigation
[17]	BO	A* + NLP	joint dynamics	dyadic manipulation
[157]	LTL + MBO	MILP	aerial and mobile robot dynamics	quadrotor and car navigation
[158]	Hybrid Automata	MILP	kinematics	truck and drone delivery
[159]	Hybrid Automata	MILP	Cartesian dynamics	tabletop manipulation
[160]	LTL + MLD	MILP	Cartesian dynamics	manipulation
[1]	PDDL+	MICP	kinematics	unified loco-manipulation
[161]	MTL	MILP + gradient descent	Cartesian dynamics	manipulation
[153]	MLD	ADMM	joint dynamics	manipulation and locomotion
[162]	STL	NLP	aerial dynamics	quadrotor
[9]	STL	NLP	Cartesian dynamics	manipulation
[154]	STL	NLP	joint dynamics	locomotion

Abbreviations: MCTS=Monte Carlo Tree Search, MBTS=Multi-bound Tree Search, ASP=Answer Set Programming, KOMO=K-order Markov Path Optimization, BO=Bilevel Optimization, MBO=Multibody Optimization, MLD=Mixed Logical-Dynamical System

low-level skills, diffusion policy [151] iteratively refine the noise into action sequence through a learned gradient field that is conditioned on the observations, which provides stable training and accommodates high-dimensional action sequences. For legged locomotion, Viereck and Righetti [152] proposed to learn a neural network that generates the desired centroidal motion real-time, which is subsequently integrated with a whole-body controller.

VI. INTEGRATED TAMP

Integrated TAMP presents a holistic approach that contrasts with others separately handling task planning (see Section IV) and motion planning (see Section V). In optimization-based TAMP, the plan is not merely required to be feasible but also expected to approximate the global optimality. The crucial consideration of integrated TAMP lies in the interdependence between task planning and motion planning. This interplay forms the cornerstone of the design of efficient TAMP algorithms and represents an area of active research.

The optimization-based formulations for TAMP often involve a hybrid optimization of discrete symbolic-level decisions and continuous motion-level trajectories, as shown in Section II-B. To this end, we identify two general approaches to formulate and solve the hybrid optimization: *logic-guided TAMP* and *TO-guided TAMP*. While both approaches inherently solve the hybrid optimization problems, they fundamentally differ in formulations and algorithms, particularly in the definition of discrete variables and the selection of search spaces. Fig. 10 shows the overall organization of this section and main topics discussed. Table II presents a representative set of classical approaches for optimization-based TAMP, highlighting their formulations, algorithms, and whether dynamics or kinematics are considered in the application.

Logic-guided TAMP (see Section VI-A) is formulated based on symbolic languages, such as PDDL, with the continuous variables and constraints for motion planning embedded as continuous-level realization of symbolic planning (referred to as *refinement* hereafter). A notable formulation in logic-guided TAMP is logic-geometric programming (LGP) [8], where logic at the symbolic level governs the constraints imposed on TO, i.e., the motion planner. The algorithm structures for logic-guided TAMP typically involves a state-space search-based task planner, with hand-designed heuristics specific to the planning problem (see Section VI-A1). The motion planner is typically interleaved with the task planner to refine the plan skeleton generated by the task planner (see Section VI-A2).

TO-guided TAMP (see Section VI-B) is formulated as a single TO problem with binary variables that represent discrete decisions. This formulation often views the hybrid optimization problem of TAMP as mixed-integer programming (MIP). Frequently, TO-guided TAMP is derived from temporal logic representations as introduced in Section III-B. The methods to solve TO-guided TAMP typically employs general-purpose numerical solvers, such as B&B, without problem-specific heuristics. Unlike logic-guided TAMP, the algorithm's search space is defined not by the explicit state space of the planning problem, but the solution space of the underlying numerical program (see Section VI-B1). In addition, efforts have been made to improve the scalability of TO-guided TAMP by splitting the MIP into subproblems [153] (see Section VI-B2) or formulating the planning problem as a fully continuous optimization [9], [154] (see Section VI-B3).

Despite the progress made in classical optimization techniques for integrated TAMP, these methods still typically have limited scalability due to the combinatorial nature of task planning and numerical complexity of motion planning. One current trend of research is to explore the use of learning-based techniques to improve the efficiency of TAMP algorithms. For logic-guided TAMP, learning methods have been utilized in the interaction between task planning and motion planning layers, for example, learned action feasibility (see Section VI-C1) and search guidance (see Section VI-C2). Along a different line of research, reusable motion skill acquisition has been explored, which facilitates the efficiency improvement for motion generation in long-horizon tasks (see Section VI-C3). For TO-guided TAMP, integrating learning-based techniques to reduce the computational burden of MIP problem has been an active area of research (see Section VI-C5).

A. Logic-Guided TAMP

In logic-guided TAMP, the approach to solving the hybrid planning problem can be conceptualized as constructing a trajectory tree. In this trajectory tree representation, each node corresponds to a symbolic state and each edge represents a trajectory segment. Given the intertwined nature of TAMP, the determination of a symbolic state's feasibility and its associated cost is influenced by a combination of symbolic and continuous domains.

A naive approach to solve for logic-guided TAMP is to impose a strict hierarchical structure [2], where task planning precedes, followed by motion planning to refine the proposed plan skeleton in continuous domain. This approach hinges on the downward refinement property [163], which posits that for every plan skeleton generated by the task planner, a corresponding continuous motion plan exists. However, the downward refinement property does not hold in most real-world scenarios. This necessitates mechanisms for replanning or backtracking at the task planning level upon realizing that a current plan skeleton becomes infeasible in the motion planning level.

On the other extreme, a fully intertwined algorithm for TAMP might require a call to the motion planner every time a new node in the search tree is expanded, in order to validate the feasibility of the selected symbolic action sequence and to generate a feasible and low-cost continuous motion plan. This method fully determines each symbolic state's reachability and its associated cost is influenced by a combination of planning at symbolic and continuous domains. However, each motion planner call is often computationally expensive, and a majority of symbolic states expanded and trajectory segments solved are unused in the final solution. This often makes the fully intertwined approach computationally intractable.

Therefore, the main research question is how to effectively interface between task planning and motion planning layers in order to curtail the size of the search tree and minimize the number of calls to the motion planner, while still effectively solving for feasible and ideally optimal solutions. Fig. 11 illustrates the overall algorithm structure that is commonly seen in logic-guided TAMP.

1) Search Heuristics in TAMP:

Many search heuristics in TAMP attempt to solve a relaxation of the underlying motion planning problem, in order to obtain an estimation of the feasibility and cost of the action. For example, in TAMP for navigation problems [156], Euclidean distance in 2-D space serves as an admissible and easily computable heuristic function, which improves planning efficiency while guaranteeing task-level optimality. However, in planning domains with high-dimensional configuration spaces, it is often difficult to generate an analogous distance measure that estimates the action costs. One intuitive approach is to evaluate the action feasibility and cost based on the initial and final configurations of the robot or object while ignoring the intermediate trajectories. For example, inverse kinematics (IK) is commonly used to reason about the feasibility of the initial and final robot poses in the action without generating the full trajectory [125], [164]. For collaborative robot manipulation, the angular displacement of a manipulated object is often used as heuristics [17]. Agostini and Piater [165] proposed an object-centric representation of manipulation constraints that unifies TAMP into a single heuristic search that is amenable to existing AI planning heuristics. Toussaint [8] proposed a multistage method to solve the TAMP: 1) optimizing over the final configuration given an action sequence, 2) optimizing over all kinematics configurations at symbolic state transitions, and 3) optimizing over the entire trajectory. The first two stages effectively act as heuristics during tree search to check the geometric feasibility of a given symbolic action sequence, while the costly TO is only conducted in the final stage.

Heuristics presented so far consider only the initial and final states in a symbolic action. Therefore, no feasibility or cost information about the intermediate states along the trajectory is available, which makes the heuristics easy-to-compute but less informative. This is insufficient to solve more complicated problems, where the path feasibility of the actions plays an important role in the planning process. In comparison, [155], extended from [8], proposes to use a TO with a very coarse time resolution (2 time steps per symbolic action). These heuristics

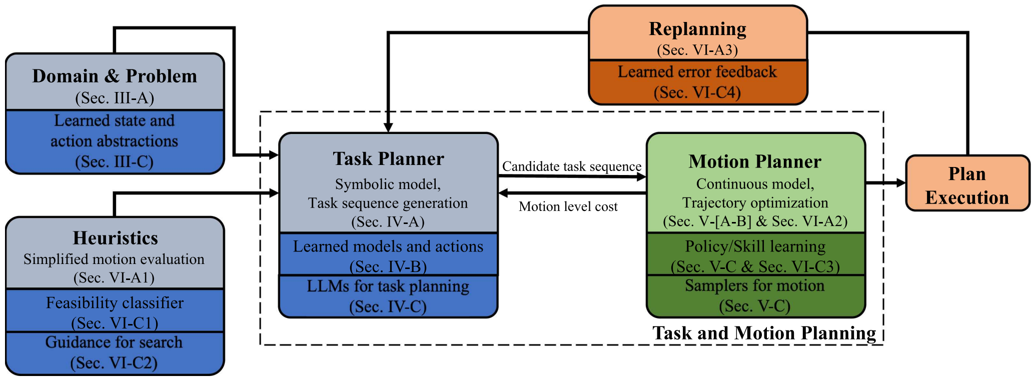


Fig. 11. Schematic overview of *logic-guided TAMP* and associated learning approaches: blue blocks illustrate domain representation and task planning level; green blocks denote the motion planning level; orange blocks represent plan execution and failure recovery mechanisms.

incorporate some path feasibility information while remaining relatively fast to compute, effectively achieving a different level of informativeness-relaxation tradeoff.

Additional work has been done to discover and prune infeasible actions before the corresponding node is reached. If an action is determined to be infeasible by a heuristic function or a motion planner during the search, the same action that exists on other branches of the tree would also be infeasible if no other actions are taken to modify the states relevant to the infeasible action. Srivastava et al. [166] proposed a planner-independent task-motion interface layer, where additional `infeasible` predicates are introduced to the task planning domain when an infeasibility is found by the motion planner.Toussaint and Lopes [155] extended this method to operate in conjunction with Monte Carlo tree search in an optimization-based TAMP formulation.

2) Multimodal Motion Planning (MMMP) Solved by TO: After a complete or partial plan skeleton is generated by the heuristics search process at the task planning level, the plan skeleton is refined into a continuous trajectory by TO. The problem of TO over a given plan skeleton is akin to the conventional MMMP problem proposed in the sampling-based planning community [167], [168]. TO incorporates the mode transitions and mode constraints derived from the symbolic decisions to form a MMMP problem. Mode constraints are predominantly expressed as manifold constraints in TO. Furthermore, transitions at the symbolic level, often called "symbolic switches," are often represented as continuity constraints between trajectory segments.

Two main strategies arise to solve the MMMP using TO. The first paradigm emphasizes segment-wise optimization, wherein trajectory segments associated with individual actions in the symbolic sequence are solved independently. For example, LGP-based formulation typically uses kth order motion optimization [169] as the underlying motion planner. The authors in [15] and [170] further extends LGP to incorporate dynamics constraints and predicates. Migimatsu and Bohg [16] proposed an object-centric TO formulation based on LGP. Similarly, Zhao et al. [125] solved the hybrid NLP as trajectory tree, but aims to improve the efficiency of the solver by using ADMM to handle the constraints of the trajectory segments in a distributed manner. Another line of research attempts to solve for the full motion trajectory as a whole given the symbolic sequence: Zimmermann et al. [171] proposed a multilevel optimization framework that exploits the implicit differentiation method to solve the switch conditions and full trajectories holistically. Phoon et al. [172] used a multiphased TO approach that optimizes the entire motion sequence simultaneously.

3) Receding Horizon TAMP: The real-world application of TAMP for long-horizon dynamic tasks is often hindered by failures in plan execution due to changes in the environment, interaction with human, or noisy sensor inputs. Receding horizon TAMP has been explored to mitigate this issue via online replanning. Receding horizon TAMP is analogous to model predictive control (MPC) [173], where the planning problem is solved iteratively over a receding time window. The specific challenge in receding horizon TAMP is to appropriately define the finite time horizon over the hybrid planning domain. The works in [125] and [174] rely on task-specific decomposition, where each receding horizon planning iteration achieves the goal for a subtask. The authors in [175] and [176] proposed to plan over a fixed action-horizon, where a full task plan is generated in each iteration, while the motion plan is only computed for a predefined number of actions. Chen et al. [177] developed branch-MPC, where the objective function is optimized over a scenario tree, which is constructed by enumerating the predicted environmental responses.

Example: To address the tabletop manipulation scenario through TAMP, the initial step involves the task planner generating a plan skeleton using tree search. This process involves constructing a tree where nodes represent potential states of the environment and edges represent actions, such as moving or stacking the objects A, B, C. The objective is to find a sequence of actions that leads to the desired configuration with maximal height for object A. An example plan skeleton can be seen in Fig. 12.

During the tree search, IK is employed as a heuristic function to check the feasibility of actions. This involves determining whether the robot can physically reach and manipulate the objects as required by the actions in the plan skeleton. The

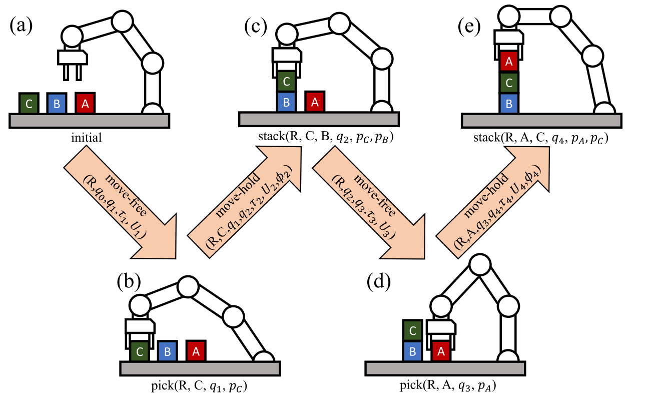

(a)
initial
move-free (R, q_0, q_1, τ_1, U_1)
(b)
pick(R, C, q_1, p_C)
move-hold (R, C, q_1, q_2, τ_2, U_2, ϕ_2)
(c)
stack(R, C, B, q_2, p_C, p_B)
move-free (R, q_2, q_3, τ_3, U_3)
(d)
pick(R, A, q_3, p_A)
move-hold (R, A, q_3, q_4, τ_4, U_4, ϕ_4)
(e)
stack(R, A, C, q_4, p_A, p_C)

Fig. 12. Illustration of a plan skeleton for the tabletop manipulation example. (a)–(e) Plan to stack block A on top of C and B using `pick` and `stack` actions; the arrows denote the associated `move-free` and `move-hold` actions and their required continuous decision variables.

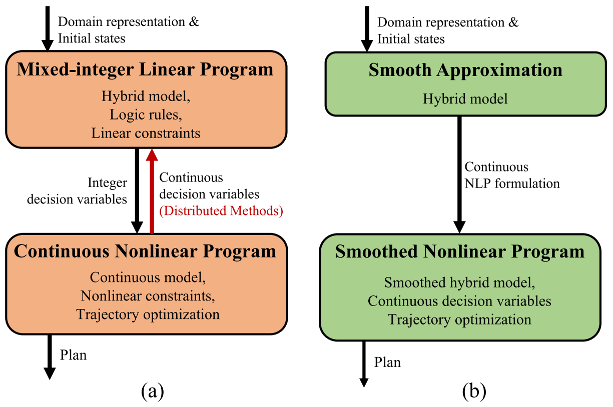

Domain representation & Initial states
Mixed-integer Linear Program
Hybrid model, Logic rules, Linear constraints
Integer decision variables
Continuous decision variables (Distributed Methods)
Continuous Nonlinear Program
Continuous model, Nonlinear constraints, Trajectory optimization
Plan
(a)
Domain representation & Initial states
Smooth Approximation
Hybrid model
Continuous NLP formulation
Smoothed Nonlinear Program
Smoothed hybrid model, Continuous decision variables Trajectory optimization
Plan
(b)

Fig. 13. Example algorithm structures for *TO-guided TAMP:* (a) hierarchical and distributed methods and (b) smoothed approximation methods.

use of IK as a heuristic aids in efficiently pruning the search tree by quickly eliminating infeasible actions, thereby focusing the search on promising solution paths. Once a preliminary plan skeleton is generated, it is passed to a multimodal motion planner, which refines this skeleton into a detailed, executable plan. This process is repeated iteratively until the optimal plan is reached or the allocated planning time elapses.

B. TO-Guided Tamp

In contrast with the approaches discussed in Section VI-A, where the interactions between task planning and motion planning are expressed *explicitly*, the common methods to solve TO-guided TAMP typically rely on internal features of numerical algorithms, such as branch-and-bound (B&B) and ADMM, to achieve interplay between the discrete and continuous decision variables *implicitly*.

1) B&B Methods: One typical approach to solve MIP is B&B-based algorithms [178], [179]. This method partitions the solution space into smaller subsets (branching) and uses bounds on the objective function to eliminate regions that do not contain an optimal solution. Initially, integer constraints are relaxed to provide an initial bound. The algorithm then branches based on fractional integer variable values, constructing a search tree. By assessing bounds for each subproblem and pruning branches that cannot improve the current best solution, B&B converges to the global optimum after multiple iterations. However, MIP is classified as a NP-hard problem [180], therefore several branching heuristics are commonly used [181] to improve scalability, analogous to the state-space search heuristics discussed in Section VI-A. For example, strong branching heuristics [182] aims to produce a small B&B tree by selecting the variable to branch that will result in the best improvement of the objective function. Alternatively, local neighborhood search [183] attempts to improve upon existing feasible solutions by local search.

B&B-based algorithms are widely implemented in commercial solvers, such as Gurobi [184], Mosek [185], and Matlab [186]. However, many off-the-shelf implementations are only able to efficiently solve mixed integer linear programming (MILP) or mixed integer convex programming (MICP). Therefore, one commonly adopted strategy is to formulate the TAMP problems as MILP or MICP in order to effectively leverage the commercial MIP solvers.

From the formal control community, such as temporal logic, [157] avoid reactive synthesis by directly encoding LTL formula as mixed-integer linear constraints on nonlinear systems, and aim to find an optimal control sequence. Chen et al. [158] encoded the LTL-based hybrid planning problem as an MILP by fixing the number of automaton runs and reasoning over temporally concurrent goals. Kogo et al. [159] integrated an existing TAMP model with collision avoidance using an MILP formulation with hard constraints on collision and soft constraints on goal positions. Katayama et al. [160] proposed an object-oriented MILP formulation for dual-arm manipulation by representing the LTL formulas, robot end effector dynamics, and object dynamics as a mixed logical dynamical (MLD) system. Bredu et al. [1] proposed grounded task planning as mixed integer programming, which builds a hybrid funnel graph (HFG) from the hybrid planning problem description in PDDL+, and encodes the HFG as an MICP.

2) Hierarchical and Distributed Optimization Methods: For systems subject to nonlinear dynamics, the optimization formulation extends to mixed integer nonlinear programming (MINLP). However, the computational burden associated with MINLP is often prohibitive, making them impractical for many real-world applications. In order to manage the computational complexity, the MINLP are often reformulated by decomposing it into solvable subproblems in a hierarchical or distributed fashion [see Fig. 13(a)].

For the hierarchical methods, Saha and Julius [161] proposed a hierarchical framework that uses MTL to express specifications for object manipulation tasks and encodes them into an MILP at the high level to solve for task sequence and manipulation poses. Meanwhile, it employs a gradient-descent-based optimization at the low level to compute collision-free robot trajectory. Similarly, Funk et al. [187] solved robot assembly discovery problem via a tri-level hierarchical planning structure, where the high level solves an MILP for object arrangement. The work in

[188] and [189] address bipedal locomotion problem in partially observable environment with a LTL-based task planner and a reduced-order model motion planner. The main drawback of these hierarchical frameworks is that the low-level TO only attempts to refine the high-level candidate solutions with an detailed continuous-level trajectory, but cannot influence the high-level MILP to achieve a better discrete solution.

To address the communication issue between the high and low level in hierarchical methods above, the distributed methods [153], [190] use ADMM to convert a MINLP into a consensus problem between a MICP, which involves the logical rules and discrete variables, and a continuous NLP that involves the nonlinear kinematics and dynamics. The MICP and NLP shares mutual information through the consensus constraints. The ADMM-based algorithm for MINLP is demonstrated to be effective in a modular robot climbing [153] and manipulation task [190]. However, ADMM relies on augmented Lagrangian method, which assumes all decision variables are continuous. Consequently, the presence of integer variables in TAMP can impede ADMM's convergence. To circumvent this limitation, modifications to the ADMM algorithm are required. One potential solution involves the direct copying of integer solutions across subproblems, effectively bypassing the primal-dual update process for integer variables [153].

3) Smooth Approximation Methods: Different from the hierarchical and distributed methods above, another line of research to circumvent the combinatorial complexity of MIP is reformulate the hybrid optimization into a continuous NLP with a specific cost function or constraint representing the smooth approximation of the discrete task planning [see Fig. 13(b)]. Such a formulation can be solved more efficiently with gradient-based solvers. Recent works in STL utilize smooth approximations of a task specification formula and encode the corresponding robustness degrees into the NLP cost functions [162], [191], [192], [193]. The work in [9] builds on the smooth approximation approach but focuses on handling multiple dynamic modes in robot manipulation tasks. The authors in [154] and [194] applied the STL specifications and robustness degrees on the push recovery scenarios for bipedal locomotion. Envall et al. [3] proposed a differentiable scheme for multiarm manipulation problems by treating robot task assignment implicitly as continuous constraints that associate the states of robots and objects. Analogous to receding horizon TAMP discussed in Section VI-A3, MPC-based methods have been developed based on STL to enable formal guarantees or reasoning about robustness of the task satisfaction in an online fashion [194], [195], [196], [197], [198].

C. Learning for Combined TAMP

Classical TAMP frameworks [2], [199] require accurate, special-purpose perception systems and hand engineered manipulation skills, rendering these approaches less effective while handling novel problems. To overcome this issue, in recent years, there has been extensive exploration of learning techniques within TAMP community. Data-driven approaches allow robots to make informed decisions based on prior examples and experiences, which enhance flexibility and generalizability. Furthermore, the scalability of classical TAMP methods is often limited by the problem size of the tree search for complex problems and the computational cost to evaluate heuristics and optimal trajectories. Learning-based methods show potential to accelerate or replace some of the computationally expensive components of classical methods, such as feasibility checking, search guidance, and skill learning. Categorizing the works based on the roles and functionalities of the learned components, we primarily classify them into the following five categories. Note that, although some of the low-level motion planners used by works cited in this section might not be optimization-based, the methodologies discussed here are fundamental and highly relevant to optimization-based TAMP. A list of representative learning approaches is presented in Table III.

1) Learning Feasibility Classifier: Traditionally, TAMP methods leverage the geometric and dynamic information in the continuous domain to determine the feasibility of tasks during discrete search. However, it can be challenging to incorporate this feasibility check mechanism into a discrete planner. A single feasibility check might involve computationally expensive operations, such as collision checking, IK, or even TO. In addition, the selection of an action or associated geometric parameters can have long-horizon implications on the feasibility of the plan.

To address these issues, Wells et al. [200] proposed to train a classifier for evaluating feasible motions and use the classifier as a heuristic for discrete task plan search. Driess et al. [29] proposed to learn a neural model for evaluating the hypothesized discrete actions based on visual images. Noseworthy et al. [201] leveraged active learning to efficiently collect the data for training the plan feasibility classifier, and then utilize the learned classifier to guide the planning and execution. Xu et al. [27] similarly train a feasibility classifier with a neural network, which estimates the feasibility of proposed TAMP actions from images of the robot's workspace. To avoid exhaustively reevaluating infeasible motion-level actions, Sung et al. [205] proposed learning backjumping heuristics to identify infeasible actions for efficient backtracking during the discrete search. Alternatively, Yang et al. [28] developed a transformer-based framework for directly predicting the feasibility of finding motion trajectories for the given task plan conditioned on the environment state. Curtis et al. [206] learned to predict the affordances of actions from color and depth images, which helps the TAMP solver generalize to environments with unknown object models.

2) Learning Search Guidance: When dealing with planning challenges with extended continuous state-action spaces, relying on random uniform sampling of action parameters without guidance until a path to a goal is discovered proves to be extremely inefficient. In addition, gradient-based methods frequently struggle when the optimization manifold of a specific problem lacks smoothness. To tackle these challenges, some researchers propose to learn samplers to speed up the TAMP solver for sequential manipulation. Wang et al. [207], [208] proposed to jointly learn the action samplers and the conditions of the models. Kim et al. [209] proposed to learn an action sampling distribution with adversarial training to guide the search toward the task goal, then they develop a score space representation and leverage it for transferring constraints to novel situations [210],

TABLE III
OVERVIEW OF LEARNING TECHNIQUES FOR PLANNING

Papers	Learned Components	Data	Collection Strategy	Application
[200]	Feasibility Classifier	object states; motion feasibility	offline	tabletop manipulation
[201]	Feasibility Classifier	abstract action sequence; plan feasibility	online	tabletop manipulation
[28]	Feasibility Classifier	abstract action sequence, goal descriptions, images, and object states; plan feasibility	offline	mobile manipulation
[39]	State and Action Abstractions	low-level transitions with object states and actions; symbolic predicates	offline	mobile manipulation
[60]	State and Action Abstractions	low-level transitions with images, segmentation masks, and actions; symbolic predicates	offline	mobile manipulation
[88]	State and Action Abstractions	low-level transitions with images and actions; skill feasibility	offline + online	tabletop manipulation
[89]	State and Action Abstractions	low-level transitions with object states and actions; skill execution cost	offline + online	tabletop manipulation
[33]	Control Policy	low-level transitions with object states and actions; rewards	online	tabletop manipulation
[202]	Control Policy	low-level transitions with object states and actions	offline	tabletop manipulation
[203]	Control Policy; State and Action Abstractions	low-level transitions with object states and actions; symbolic predicates	offline	tabletop manipulation
[204]	Combinatorial Decisions	MICP solutions	offline	free-flyer & dexterous grasping

which facilitates the speed-up of the search in TAMP. Chitnis et al. [211] proposed to formulate the task plan refinement as a Markov decision process and leverage RL to learn a policy to guide the task plan search. Similarly, Kim et al. [212] developed an abstract representation of states and goals, and learn a value function using graph neural networks to guide TAMP. Ortiz et al. [213] proposed to represent the problem as constraint graphs and break the overall problem into smaller sequential sampling problems, which are solved by learning assignment orders with Monte Carlo tree search, then they propose to utilize generative models for learning to sample solutions on constraint manifolds [214].

For complex scenarios involving heterogeneous multiagent systems, the planning framework must effectively handle task allocation and scheduling. Traditionally, these problems are addressed using heuristics-based approaches, but the combinatorial growth of the search space makes them computationally demanding. To tackle this challenge, RL methods have been widely explored for both task allocation [215], [216] and task scheduling [217]. These studies demonstrate RL's capability to significantly improve the time and memory requirements for multiagent planning.

3) Learning Skill Policies: In recent advancements, some researchers have proposed the concept of acquiring foundational skills within the operational framework of TAMP systems. Notably these methods differ from conventional motion planning by learning skill policies, which are capable of generating trajectories through rollout and can be viewed as implicit motion plans. Learning skill policies at motion planning level has shown significant potential to improve the overall efficiency of TAMP due to the reusability and adaptability of the skills. McDonald et al. [34] proposed to distill the knowledge of a TAMP system into a hierarchical RL policy, where they first leverage the TAMP solver to generate supervision data for imitation learning, and then the learned control policies are utilized to further speed up the TAMP solver. LEAGUE [33] proposes to learn RL policies with the guidance of a task planner, where the acquired skills are reused in the TAMP system to accelerate the learning of new skills, which progressively grows its capability for solving long-horizon manipulation tasks in a more efficient manner. The work in [218] extends [33] to leverage LLMs to guide skill learning. HITL-TAMP [202] develops an efficient teleoperation system that leverages TAMP to reach the beginning state of the demonstration phase, where the low-level control policies are learned with the collected data and integrated into the system in the testing stage. Meng et al. [219] proposed to learn control policies for satisfying the long-term tasks specified with STL, where the learned models are used to generate trajectories via MPC during execution. Silver et al. [203] proposed to jointly learn the symbolic operators and low-level skill policies with demonstration data, the extracted operators are first used to generate abstract task plans, and the learned policies are then invoked for generating motions to achieve subgoals.

4) Learning for Error Recovery: Robots tend to make mistakes during the execution phase of TAMP in unstructured environments. Therefore, robust policies are critical to both prevent and recover from such errors. Pan et al. [220] proposed a TAMP framework that accounts for potential failures during execution, enabling the robot to calculate and perform necessary actions to achieve the goal despite potential failures. The strength of this framework lies in its continuous reassessment and adjustment of the basic beliefs associated with actions, minimizing the likelihood of execution failures. Wang and Kroemer [221] used multimodal information to create a system to improve the robustness of robot manipulation tasks in unstructured environments. This procedure involves developing a multimodal state transition model, grounded on task contact dynamics and observed transitions. Similar to [221], Luo et al. [222] used learning-from-demonstration techniques to enable error recovery in robots.

Recently, vision language models (VLMs) and LLMs have been utilized in error recovery. For instance, Zhang et al. [223] developed a system, called task planning via visual question answering (TPVQA). This system leverages VLMs to identify action failures and validate action possibilities, thereby increasing the likelihood of successful plan execution. REFLECT [224] uses LLMs to autonomously identify and explain robot failures. It creates a hierarchical summary of past robot experiences, utilizing multisensory data, systematically formulates plans to correct failures, and effectively managing complex tasks. SayPlan [225] offers an error recovery strategy via an iterative replanning process. It produces an initial plan tested in a simulator, and any unexecutable actions detected are reported back to the LLM for plan refinement. This iterative feedback system, a method of error recovery learning, enables the LLM to steadily boost its planning efficiency, adapting to expansive, complex environments. Huang et al. [97] leveraged three types of environmental feedback in LLMs to generate task plans and handle failures.

5) Learning for MIP: One main limitation of MIP-based methods for TAMP is that MIP is expensive to solve if a large number of integer variables are involved. Recent works in the optimization literature have shown promises to learn the branching heuristics and initializations for MIP. The authors in [226] and [227] provided detailed surveys for learning-based MIP solvers. Alvarez et al. [228] first proposes to imitate the strong branching heuristics via supervised learning. The work in [229] similarly imitates strong branching but develops a framework to solve MIP in an instance-specific manner. Gasse et al. [230] further extends the imitation learning method to incorporate GNN-based models and represent the MIP problem as a bipartite graph. Along another line of research, the work in [231], [232], and [233] combined RL with local neighborhood search for MIP and learn to determine the neighborhood selection, initialization and search neighborhood, and neighborhood size, respectively.

Aside from the advancements in learning-based general MIP solvers, recent works also specifically explore learning for MIP in TAMP tasks. [234] trains a neural network offline that imitate the solution of MILP solvers as a warm-start to online multi-robot planning tasks. Cauligi et al. [204], [235] speeds up MICP for robot planning and control by solving a dataset of MICP offline and learning the combinatorial decisions via a strategy classifier. The classifier is used to predict the discrete variables and the remaining convex program is solved online during execution. Deits et al. [236] learned the optimal value function from mixed-integer optimizations to guide policy search.

VII. Future Challenges and Opportunities

Foundation Models for TAMP: The integration of LLMs and VLMs in robot planning is emerging but faces a few challenges. In robot planning, it is crucial to break down complex task specifications into actionable steps suited for particular environments [64], [237], [238], [239], [240]. However, LLMs and VLMs often struggle with this. Their plans may be too abstract, failing to consider the practical constraints of the physical world. This presents a significant issue for robots that need concrete and feasible instructions for physical operations. Furthermore, the current capabilities of LLMs and VLMs are limited, which in turn restricts their effectiveness in robot planning. For instance, spatial reasoning is essential in robot planning, yet LLMs and VLMs may not accurately understand physical spaces and dynamic environments [241], [242], [243]. Robot planning also requires considering historical data and long-term goals. The limited short-term memory of LLMs could result in information loss during sequential or multistage tasks, impacting the coherence and efficiency of planning [244], [245], [246]. Although recent developments aim to enhance the capabilities of LLMs and VLMs, they seem not to fundamentally solve these limitations [241], [247]. In addition, in open-world settings such as homes, malls, and hospitals, robots also need the ability to adapt to new, unforeseen tasks. Despite the complexities, progress in developing LLMs and VLMs for these purposes is emerging [248], [249].

Diffusion Models for TAMP: The use of diffusion models in motion planning, such as diffuser [250], decision diffuser [251], and diffusion policies [151], [252], has been explored for their flexibility and composability. Pan et al. [253] proposed a model-based diffusion planner that solves TO using the diffusion process without external data. In the context of TAMP, diffusion models can serve as trajectory samplers for individual skills, offering a robust method for generating diverse and feasible motion plans. Mishra et al. [254] presented generative skill chaining, where short-horizon skill-centric diffusion models are learned and a compositional framework is established to directly generate long-horizon plans given a plan skeleton. Fang et al. [255] integrated diffusion skill samplers into a classical TAMP method, which adapts the framework to partially observable planning domains. Promising future research can explore the synergy between LLMs and diffusion models to develop generative multitask models and end-to-end TAMP frameworks.

Multimodal Sensing for TAMP: Currently, most TAMP frameworks rely predominantly on visual sensing. However, integrating multimodal sensing, such as visual, tactile, and acoustic modules, can significantly improve the robots' capabilities for contact-rich tasks in imitation learning [256], [257], [258], since each sensing modality provides unique and useful contact information related to the manipulation task that covers a wide range of geometric scales and frequency bandwidths. Future research opportunities include further integrating multi-modal sensing with TAMP by advanced sensor fusion as well as contact information representation, extraction, and utilization. Given these potentials, we can potentially enhance the performance of TAMP with better heuristics for task planning and more accurate contact models for TO.

Policy Learning in and for TAMP: In the realm of RL-based policy learning, key obstacles revolve around sample inefficiency (the cost of trial and error) and the reliance on meticulously crafted dense reward functions. These challenges are exacerbated in the realm of complex, long-term tasks and often mandate the initial training of RL algorithms within simulators, thereby adding complexity to the transfer of acquired behaviors from simulation to real world [33]. While tapping into robot teleoperation data directly from real-world environments could alleviate generalization issues, this strategy demands careful system design to ensure seamless robot-human handover and maintain efficient data collection [202]. Moreover, constructing diverse task configurations that encompass an adequate range of scenarios in terms of object geometries, spatial arrangements, and lighting conditions is nontrivial, presenting additional challenges for acquiring generalizable models in long-horizon tasks. In addition, the design of observation and action representations is pivotal in enhancing the generalization and reusability of acquired skills.

TAMP for Locomotion and Manipulation: Although the duality between locomotion and manipulation [259] means that they can be viewed as equivalent problems, the distinct nature of locomotion and manipulation presents challenges in applying TAMP more complex robotic systems, e.g., humanoid robots with loco-manipulation capabilities. From the TAMP perspective, both involve hybrid planning for dynamic contact interactions between the robot and the environment but differ in the representation and frequency of the hybrid events. Manipulation tasks are usually object-centric [2] and occur at lower frequencies for contact switching, focusing on precise interactions with objects. In contrast, locomotion involves robot-centric motions with higher frequency contact switching, typically handled through a hierarchical approach of contact planning followed by trajectory generation [24], often facilitated by a centroidal trajectory planning approach [260], [261]. This difference in representation and operational frequency raises important research questions in developing a unified TAMP framework for loco-manipulation, requiring a balanced integration of these aspects. Sleiman et al. [19] showed promises in integrating graph search and TO for long-horizon loco-manipulation tasks by handling the object-centric and robot-centric tasks separately and transferring offline generated plans to online execution. Sferrazza et al. [262] illustrated the recent trend to use hierarchical RL to solve whole-body loco-manipulation problems for humanoid robots with dexterous hands. Still, TAMP for unified locomotion and manipulation remains a challenge, due to complicating factors, such as the complexity of dexterous grasping and locomotion planning in uneven terrains.

TAMP for HRC: TAMP for HRC faces challenges due to the uncertainties of human intentions and behaviors. For effective collaboration, it is essential for robots to predict human's symbolic intentions and continuous motions and integrate this understanding into the planning process [263]. Recent development in this area include human-aware task planning [264], hierarchical planning approaches [265], [266], and the incorporation of human motion prediction into LGP [267]. An emerging area of research in HRC is the exploration of novel communication modalities between humans and robots, for example, the integration of conversation interactions [268] in the robot planning framework. Moreover, human intent can be expressed via physical interactions in physical HRC scenarios, such as hand-over [269] and collaborative transport of objects, where the object and human dynamics are considered [270]. How physical HRC can be achieved efficiently via TAMP remains an active area of research. These recent trends signals a move toward more intuitive and natural human–robot interactions through both physical and language interfaces.

TAMP in Real-World Applications: Optimization-based TAMP has diverse real-world applications across various industries. In industrial settings, TAMP is used for construction planning [174], and rebar grid traversal [21]. In addition, TAMP enables autonomous aerial vehicles (AAVs) to navigate complex environments for delivery services and environmental monitoring [271], as well as agricultural tasks [272]. In domestic applications, TAMP facilitates household tasks such as cooking [273], and manipulation of doors and dishwashers [19]. In lab environment, TAMP is deployed for medical test tube rearrangements [274]. Expanding TAMP to broader real-world applications requires overcoming current challenges, such as developing more robust methods to accommodate varying environmental factors, problem settings, and human interactions. In addition, proper scene understanding and representation that captures spatial and semantic relationships is essential and remains an open research area for deploying TAMP in open-world environments. Furthermore, implementing low-level control for robots in real-world scenarios involves handling environmental variability, sensor noise, and calibration issues, which affect feedback reliability. Addressing these challenges is crucial for effective and safe robot deployment.

Ethical and Societal Implications: The deployment of TAMP and robotics raise several ethical and societal considerations [275], [276], including the following:

1) ensuring the safety and reliability of robot planning, particularly in human-populated environments;
2) developing strategies to mitigate socio-economic impact of job displacement caused by increased automation;
3) responsible handling of data collected during training of machine learning models to protect privacy and intellectual property;
4) minimizing environmental impact via more energy efficient training process for machine learning models and sustainable practices in the the production, operation, and disposal of robots.

Glossaries of Terms

Alternating Direction Method of Multipliers (ADMM): An algorithm that solves convex optimization problems by breaking it into smaller pieces, each of which is easier to handle [100].

AI Planning: An area of artificial intelligence that studies the process of automated generation of sequence of actions to achieve specific goals [22].

Bilevel Optimization: A mathematical program, where an optimization problem contains another optimization problem as a constraint [277].

Foundation Model: A general purpose model, typically trained on a large amount of data, that can be adapted for various downstream applications [278].

Heuristics: Methods to organize the search space and guide the search algorithm [22].

Large Language Model: A large pretrained language model designed to understand and generate human-like text [92].

Mixed Integer Programming: An optimization problem with a combination of continuous and discrete decision variables [180].

Mixed Logical Dynamical System: A formulation of system dynamics with mixed integer constraints [37].

Model Predictive Control: A class of control methods where a model is used to predict the future of the controlled system over a receding planning horizon [173].

Motion Planning: The problem of moving a mechanical system from a start state to a goal region [11].

Nonlinear Programming: An optimization problem where the objective function or constraints are nonlinear [99].

Reinforcement Learning: A problem where an agent learns to make decisions through a goal-directed interaction with the uncertain environment [279].

Task and Motion Planning: A hybrid planning problem that integrates high-level task planning and low-level motion planning, which enables reasoning over long-horizon, dynamic tasks [2].

Task Planning: The problem of generating an action sequence that accomplishes the goal of the task and satisfies the task specifications [23].

Temporal Logic: The formal methods for describing time dependent rules and symbols, often used to specify the correctness of a finite-state transition system [37].

Trajectory Optimization: The problem of generating a continuous robot motion path and a control sequence that optimizes an objective function subject to a set of kinematics and/or dynamics constraints [99].

References

[1] A. A. Bredu, N. Devraj, and O. C. Jenkins, "Optimal constrained task planning as mixed integer programming," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2022, pp. 12029–12036.

[2] C. R. Garrett et al., "Integrated task and motion planning," *Annu. Rev. Control Robot. Auton. Syst.*, vol. 4, pp. 265–293, 2021.

[3] J. Envall, R. Poranne, and S. Coros, "Differentiable task assignment and motion planning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst*, 2023, pp. 2049–2056.

[4] N. T. Dantam, Z. K. Kingston, S. Chaudhuri, and L. E. Kavraki, "An incremental constraint-based framework for task and motion planning," *Int. J. Robot. Res.*, vol. 37, no. 10, pp. 1134–1151, 2018.

[5] T. Lozano-Pérez and L. P. Kaelbling, "A constraint-based method for solving sequential manipulation planning problems," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2014, pp. 3684–3691.

[6] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling, "Pddlstream: Integrating symbolic planners and blackbox samplers via optimistic adaptive planning," in *Proc. Int. Conf. Autom. Plan. Scheduling*, 2020, pp. 440–448.

[7] A. Krontiris and K. E. Bekris, "Efficiently solving general rearrangement tasks: A fast extension primitive for an incremental sampling-based planner," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2016, pp. 3924–3931.

[8] M. Toussaint, "Logic-geometric programming: An optimization-based approach to combined task and motion planning," in *Proc. Int. Joint Conf. Artif. Intell.*, 2015, pp. 1930–1936.

[9] R. Takano, H. Oyama, and M. Yamakita, "Continuous optimization-based task and motion planning with signal temporal logic specifications for sequential manipulation," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2021, pp. 8409–8415.

[10] H. Guo, F. Wu, Y. Qin, R. Li, K. Li, and K. Li, "Recent trends in task and motion planning for robotics: A survey," *ACM Comput. Surv.*, 2023.

[11] A. Orthey, C. Chamzas, and L. E. Kavraki, "Sampling-based motion planning: A comparative review," *Annu. Rev. Control Robot. Auton. Syst.*, vol. 7, pp. 285–310, 2023.

[12] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Robot. Res.*, vol. 30, no. 7, pp. 846–894, 2011.

[13] P. S. Schmitt, W. Neubauer, W. Feiten, K. M. Wurm, G. V. Wichert, and W. Burgard, "Optimal, sampling-based manipulation planning," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2017, pp. 3426–3432.

[14] J. D. Gammell and M. P. Strub, "Asymptotically optimal sampling-based motion planning methods," *Annu. Rev. Control Robot. Auton. Syst.*, vol. 4, pp. 295–318, 2021.

[15] M. A. Toussaint, K. R. Allen, K. A. Smith, and J. B. Tenenbaum, "Differentiable physics and stable modes for tool-use and manipulation planning," in *Proc. Robot. Sci. Syst.*, 2018.

[16] T. Migimatsu and J. Bohg, "Object-centric task and motion planning in dynamic environments," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 844–851, Feb. 2020.

[17] T. Stouraitis, I. Chatzinikolaidis, M. Gienger, and S. Vijayakumar, "Online hybrid motion planning for dyadic collaborative manipulation via bilevel optimization," *IEEE Trans. Robot.*, vol. 36, no. 5, pp. 1452–1471, May 2020.

[18] B. Aceituno-Cabezas et al., "Simultaneous contact, gait, and motion planning for robust multilegged locomotion via mixed-integer convex optimization," *IEEE Robot. Autom. Lett.*, vol. 3, no. 3, pp. 2531–2538, Mar. 2017.

[19] J.-P. Sleiman, F. Farshidian, and M. Hutter, "Versatile multicontact planning and control for legged loco-manipulation," *Sci. Robot.*, vol. 8, no. 81, 2023, Art. no. eadg 5014.

[20] Y. Zhao, Y. Li, L. Sentis, U. Topcu, and J. Liu, "Reactive task and motion planning for robust whole-body dynamic locomotion in constrained environments," *Int. J. Robot. Res.*, vol. 41, no. 8, pp. 812–847, 2022.

[21] M. Asselmeier, J. Ivanova, Z. Zhou, P. A. Vela, and Y. Zhao, "Hierarchical experience-informed navigation for multi-modal quadrupedal rebar grid traversal," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2024, pp. 8065–8072.

[22] M. Ghallab, D. Nau, and P. Traverso, *Automated Planning: Theory and Practice*. Amsterdam, The Netherlands: Elsevier, 2004.

[23] D. Meli, H. Nakawala, and P. Fiorini, "Logic programming for deliberative robotic task planning," *Artif. Intell. Rev.*, vol. 56, pp. 1–39, 2023.

[24] P. M. Wensing, M. Posa, Y. Hu, A. Escande, N. Mansard, and A. Del Prete, "Optimization-based control for dynamic legged robots," *IEEE Trans. Robot*, vol. 40, pp. 43–63, Oct. 2023.

[25] M. Posa, C. Cantu, and R. Tedrake, "A direct method for trajectory optimization of rigid bodies through contact," *Int. J. Robot. Res.*, vol. 33, no. 1, pp. 69–81, 2014.

[26] Y. Tassa, N. Mansard, and E. Todorov, "Control-limited differential dynamic programming," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2014, pp. 1168–1175.

[27] L. Xu, T. Ren, G. Chalvatzaki, and J. Peters, "Accelerating integrated task and motion planning with neural feasibility checking," 2022, *arXiv:2203.10568*.

[28] Z. Yang, C. Garrett, T. Lozano-Perez, L. Kaelbling, and D. Fox, "Sequence-based plan feasibility prediction for efficient task and motion planning," in *Proc. Robot. Sci. Syst.*, 2023.

[29] D. Driess, J.-S. Ha, and M. Toussaint, "Deep visual reasoning: Learning to predict action sequences for task and motion planning from an initial scene image," in *Proc. Robot. Sci. Syst.*, 2020.

[30] K. Lin, C. Agia, T. Migimatsu, M. Pavone, and J. Bohg, "Text2motion: From natural language instructions to feasible plans," *Auton. Robots*, vol. 47, no. 8, pp. 1345–1365, 2023.

[31] T. Silver, S. Dan, K. Srinivas, J. B. Tenenbaum, L. Kaelbling, and M. Katz, "Generalized planning in PDDL domains with pretrained large language models," in *Proc. AAAI Conf. Artif. Intell.*, 2024, pp. 20256–20264.

[32] W. Huang, P. Abbeel, D. Pathak, and I. Mordatch, "Language models as zero-shot planners: Extracting actionable knowledge for embodied agents," in *Proc. Int. Conf. Mach. Learn. PMLR*, 2022, pp. 9118–9147.

[33] S. Cheng and D. Xu, "League: Guided skill learning and abstraction for long-horizon manipulation," *IEEE Robot. Autom. Lett.*, vol. 8, no. 10, pp. 6451–6458, Oct. 2023.

[34] M. J. McDonald and D. Hadfield-Menell, "Guided imitation of task and motion planning," in *Proc. Conf. Robot. Learn. PMLR*, 2022, pp. 630–640.

[35] M. Mansouri, F. Pecora, and P. Schüller, "Combining task and motion planning: Challenges and guidelines," *Front. Robot. AI*, vol. 8, 2021, Art. no. 637888.

[36] L. Antonyshyn, J. Silveira, S. Givigi, and J. Marshall, "Multiple mobile robot task and motion planning: A survey," *ACM Comput. Surv.*, vol. 55, no. 10, pp. 1–35, 2023.

[37] 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.

[38] O. Shorinwa, T. Halsted, J. Yu, and M. Schwager, "Distributed optimization methods for multi-robot systems: Part 1—A tutorial," *IEEE Robot. Autom. Mag.*, vol. 31, no. 3, pp. 121–138, 2024.

[39] T. Silver, R. Chitnis, J. Tenenbaum, L. P. Kaelbling, and T. Lozano-Pérez, "Learning symbolic operators for task and motion planning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 3182–3189.

[40] A. Pnueli, "The temporal logic of programs," in *Annu. Symp. Found. Comput. Sci.*, 1977, pp. 46–57.

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

[42] O. Maler and D. Nickovic, "Monitoring temporal properties of continuous signals," in *Int. Symp. Formal Tech. Real-Time Fault-Tolerant Syst.*, 2004, pp. 152–166.

[43] R. Koymans, "Specifying real-time properties with metric temporal logic," *Real-Time Syst.*, vol. 2, no. 4, pp. 255–299, 1990.

[44] Constructions Aeronautiques . et al., "PDDL|the planning domain definition language," Yale Center for Computational Vision and Control, New Haven, CT, USA, Tech. Rep. CVC TR-98-003, 1998.

[45] P. Haslum, N. Lipovetzky, D. Magazzeni, and C. Muise, "An introduction to the planning domain definition language," *Synth. Lect. Artif. Intell. Mach. Learn.*, vol. 13, no. 2, pp. 1–187, 2019.

[46] M. Fox and D. Long, , "Pddl2,1: An extension to PDDL for expressing temporal planning domains," *J. Artif. Intell. Res.*, vol. 20, pp. 61–124, 2003.

[47] D. L. Kovacs, "BNF definition of PDDL 3.1," Unpublished manuscript from the IPC-2011 website, vol. 15, 2011.

[48] M. Fox and D. Long, "Modelling mixed discrete-continuous domains for planning," *J. Artif. Intell. Res.*, vol. 27, pp. 235–297, 2006.

[49] H. L. Younes and M. L. Littman, "PPDDL1. 0: An extension to PDDL for expressing planning domains with probabilistic effects," Techn. Rep. CMU-CS-04-162, vol. 2, 2004, Art. no. 99.

[50] D. L. Kovács, "A multi-agent extension of PDDL3.1," in *Proc. 3rd Workshop Int. Plan. Compet.*, 2012, pp. 19–37.

[51] E. A. Emerson and E. M. Clarke, "Using branching time temporal logic to synthesize synchronization skeletons," *Sci. Comput. Prog.*, vol. 2, no. 3, pp. 241–266, 1982.

[52] A. Donzé and O. Maler, "Robust satisfaction of temporal logic over real-valued signals," in *Proc. Int. Conf. Formal Model. Anal. Timed Syst.*, 2010, pp. 92–106.

[53] 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.

[54] V. Kurtz and H. Lin, "Mixed-integer programming for signal temporal logic with fewer binary variables," *IEEE Control Syst. Lett.*, vol. 6, pp. 2635–2640, 2022.

[55] T. Silver et al., "Predicate invention for bilevel planning," in *Proc. AAAI Conf. Artif. Intell.*, 2023, pp. 12120–12129.

[56] R. Chitnis, T. Silver, J. B. Tenenbaum, T. Lozano-Perez, and L. P. Kaelbling, "Learning neuro-symbolic relational transition models for bilevel planning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2022, pp. 4166–4173.

[57] R. Chitnis, T. Silver, B. Kim, L. Kaelbling, and T. Lozano-Perez, "Camps: Learning context-specific abstractions for efficient planning in factored mdps," in *Proc. Conf. Robot. Learn. PMLR*, 2021, pp. 64–79.

[58] T. Silver, R. Chitnis, A. Curtis, J. B. Tenenbaum, T. Lozano-Pérez, and L. P. Kaelbling, "Planning with learned object importance in large problem instances using graph neural networks," in *Proc. AAAI Conf. Artif. Intell.*, 2021, pp. 11962–11971.

[59] Y. Zhu, J. Tremblay, S. Birchfield, and Y. Zhu, "Hierarchical planning for long-horizon manipulation with geometric and symbolic scene graphs," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2021, pp. 6541–6548.

[60] C. Wang, D. Xu, and L. Fei-Fei, "Generalizable task planning through representation pretraining," *IEEE Robot. Autom. Lett.*, vol. 7, no. 3, pp. 8299–8306, Mar. 2022.

[61] Y. Ding et al., "Integrating action knowledge and LLMS for task planning and situation handling in open worlds," *Auton. Robots*, vol. 47, no. 8, pp. 981–997, 2023.

[62] B. Liu et al., "Llm p: Empowering large language models with optimal planning proficiency," 2023, *arXiv:2304.11477*.

[63] I. Singh et al., "Progprompt: Generating situated robot task plans using large language models," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2023, pp. 11523–11530.

[64] Z. Zhao, W. S. Lee, and D. Hsu, "Large language models as commonsense knowledge for large-scale task planning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2024, vol. 36, pp. 31967–31987.

[65] A. Z. Ren et al., "Robots that ask for help: Uncertainty alignment for large language model planners," in *Proc. Conf. Robot. Learn. PMLR*, 2023, pp. 661–682.

[66] Y. Xie, C. Yu, T. Zhu, J. Bai, Z. Gong, and H. Soh, "Translating natural language to planning goals with large-language models," 2023, *arXiv:2302.05128*.

[67] J. Pan, G. Chou, and D. Berenson, "Data-efficient learning of natural language to linear temporal logic translators for robot task specification," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2023, pp. 11554–11561.

[68] Y. Chen, J. Arkin, C. Dawson, Y. Zhang, N. Roy, and C. Fan, "Autotamp: Autoregressive task and motion planning with LLMS as translators and checkers," in *IEEE Int. Conf. Robot. Autom. IEEE*, 2024, pp. 6695–6702.

[69] J. Hoffmann, "FF: The fast-forward planning system," *AI Mag.*, vol. 22, no. 3, pp. 57–57, 2001.

[70] J. A. Baier, F. Bacchus, and S. A. McIlraith, "A heuristic search approach to planning with temporally extended preferences," *Artif. Intell.*, vol. 173, no. 5-6, pp. 593–618, 2009.

[71] L. Zhu and R. Givan, "Simultaneous heuristic search for conjunctive subgoals," in *Proc. Nat. Conf. Artif. Intell.*, vol. 3, 2005, pp. 1235–1240.

[72] M. Helmert, "The fast downward planning system," *J. Artif. Intell. Res.*, vol. 26, pp. 191–246, 2006.

[73] S. Richter, M. Westphal, and M. Helmert, "LAMA 2008 and 2011," in *Proc. Int. Plan. Compet.*, 2011, pp. 117–124.

[74] I. Georgievski and M. Aiello, "An overview of hierarchical task network planning," 2014, *arXiv:1403.7426*.

[75] J. E. Hopcroft, R. Motwani, and J. D. Ullman, "Introduction to automata theory, languages, and computation," *ACM SIGACT News*, vol. 32, no. 1, pp. 60–65, 2001.

[76] A. Pnueli and R. Rosner, "On the synthesis of a reactive module," in *Proc. ACM SIGPLAN-SIGACT Symp. Princ. Program. Lang.*, 1989, pp. 179–190.

[77] S. Maoz and J. O. Ringert, "Gr (1) synthesis for ltl specification patterns," in *Proc. Joint Meet. Found. Softw. Eng.*, 2015, pp. 96–106.

[78] R. Ehlers and V. Raman, "Slugs: Extensible GR (1) synthesis," in *Proc. Computer Aided Verif.: 28th Int. Conf.*, 2016, pp. 333–339.

[79] H. M. Pasula, L. S. Zettlemoyer, and L. P. Kaelbling, "Learning symbolic models of stochastic domains," *J. Artif. Intell. Res.*, vol. 29, pp. 309–352, 2007.

[80] E. Amir and A. Chang, "Learning partially observable deterministic action models," *J. Artif. Intell. Res.*, vol. 33, pp. 349–402, 2008.

[81] G. Konidaris, L. P. Kaelbling, and T. Lozano-Perez, "Symbol acquisition for probabilistic high-level planning," in *Proc. Int. Joint Conf. Artif. Intell.*, 2015, pp. 3619–3627.

[82] G. Konidaris, L. P. Kaelbling, and T. Lozano-Perez, "From skills to symbols: Learning symbolic representations for abstract high-level planning," *J. Artif. Intell. Res.*, vol. 61, pp. 215–289, 2018.

[83] B. Ames, A. Thackston, and G. Konidaris, "Learning symbolic representations for planning with parameterized skills," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 526–533.

[84] D. Xu et al., "Neural task programming: Learning to generalize across hierarchical tasks," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2018, pp. 3795–3802.

[85] D.-A. Huang et al., "Neural task graphs: Generalizing to unseen tasks from a single video demonstration," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 8565–8574.

[86] D. Xu, R. Martín-Martín, D.-A. Huang, Y. Zhu, S. Savarese, and L. F. Fei-Fei, "Regression planning networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, vol. 32, pp. 1319–1329.

[87] F. Ceola, E. Tosello, L. Tagliapietra, G. Nicola, and S. Ghidoni, "Robot task planning via deep reinforcement learning: A tabletop object sorting application," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, 2019, pp. 486–492.

[88] D. Xu, A. Mandlekar, R. Martín-Martín, Y. Zhu, S. Savarese, and L. Fei-Fei, "Deep affordance foresight: Planning through what can be done in the future," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2021, pp. 6206–6213.

[89] J. Liang, M. Sharma, A. LaGrassa, S. Vats, S. Saxena, and O. Kroemer, "Search-based task planning with learned skill effect models for lifelong robotic manipulation," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2022, pp. 6351–6357.

[90] OpenAI, ChatGPT, Accessed: Aug. 2, 2023. [Online]. Available: https://openai.com/blog/chatgpt/

[91] H. Touvron et al., "Llama: Open and efficient foundation language models," 2023, *arXiv:2302.13971*.

[92] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig, "Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing," *ACM Comput. Surv.*, vol. 55, no. 9, pp. 1–35, 2023.

[93] S. Imani, L. Du, and H. Shrivastava, "Mathprompter: Mathematical reasoning using large language models," in *Proc. Annu. Meet. Assoc. Comput. Linguist.*, 2023, vol. 5, pp. 37–42.

[94] V. Gaur and N. Saunshi, "Reasoning in large language models through symbolic math word problems," in *Proc. Finding Assoc. Comput. Linguist.*, 2023, pp. 5889–5903.

[95] Y. Ding, X. Zhang, C. Paxton, and S. Zhang, "Task and motion planning with large language models for object rearrangement," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2023, pp. 2086–2092.

[96] A. Brohan et al., "Do as i can, not as i say: Grounding language in robotic affordances," in *Proc. Conf. Robot. Learn. PMLR*, 2023, pp. 287–318.

[97] W. Huang et al., "Inner monologue: Embodied reasoning through planning with language models," in *Proc. Conf. Robot. Learn. PMLR*, 2023, pp. 1769–1782.

[98] Y. Hu et al., "Toward general-purpose robots via foundation models: A survey and meta-analysis," 2023, *arXiv:2312.08782*.

[99] J. T. Betts, "Survey of numerical methods for trajectory optimization," *J. Guid. Control Dyn.*, vol. 21, no. 2, pp. 193–207, 1998.

[100] S. Boyd et al., "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, 2011.

[101] I. Mordatch and E. Todorov, "Combining the benefits of function approximation and trajectory optimization," in *Proc. Robot. Sci. Syst.*, 2014, Art. no. 23.

[102] M. Janner, Q. Li, and S. Levine, "Offline reinforcement learning as one big sequence modeling problem," *Adv. Neural Inf. Process. Syst.*, vol. 34, pp. 1273–1286, 2021.

[103] M. Kelly, "An introduction to trajectory optimization: How to do your own direct collocation," *SIAM Rev.*, vol. 59, no. 4, pp. 849–904, 2017.

[104] D. Pardo, L. Möller, M. Neunert, A. W. Winkler, and J. Buchli, "Evaluating direct transcription and nonlinear optimization methods for robot motion planning," *IEEE Robot. Autom. Lett.*, vol. 1, no. 2, pp. 946–953, Feb. 2016.

[105] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Math. Prog.*, vol. 106, pp. 25–57, 2006.

[106] P. E. Gill, W. Murray, and M. A. Saunders, "SNOPT: An SQP algorithm for large-scale constrained optimization," *SIAM Rev.*, vol. 47, no. 1, pp. 99–131, 2005.

[107] D. Mayne, "A second-order gradient method for determining optimal trajectories of non-linear discrete-time systems," *Int. J. Control*, vol. 3, no. 1, pp. 85–95, 1966.

[108] Z. Xie, C. K. Liu, and K. Hauser, "Differential dynamic programming with nonlinear constraints," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2017, pp. 695–702.

[109] B. Plancher, Z. Manchester, and S. Kuindersma, "Constrained unscented dynamic programming," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 5674–5680.

[110] T. A. Howell, B. E. Jackson, and Z. Manchester, "Altro: A fast solver for constrained trajectory optimization," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019, pp. 7674–7679.

[111] J.-P. Sleiman, F. Farshidian, and M. Hutter, "Constraint handling in continuous-time DDP-based model predictive control," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2021, pp. 8209–8215.

[112] W. Jallet, A. Bambade, N. Mansard, and J. Carpentier, "Constrained differential dynamic programming: A primal-dual augmented lagrangian approach," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2022, pp. 13371–13378.

[113] Y. Wang, H. Li, Y. Zhao, X. Chen, X. Huang, and Z. Jiang, "A fast coordinated motion planning method for dual-arm robot based on parallel constrained DDP," *IEEE/ASME Trans. Mechatron.*, vol. 29, no. 3, pp. 2350–2361, Jun. 2024.

[114] J. Nocedal and S. J. Wright, *Numerical Optimization*. Berlin, Germany: Springer, 1999.

[115] R. Featherstone, *Rigid Body Dynamics Algorithms*. Berlin, Germany: Springer, 2014.

[116] J. Eckstein and D. P. Bertsekas, "On the Douglas—Rachford splitting method and the proximal point algorithm for maximal monotone operators," *Math. Prog.*, vol. 55, pp. 293–318, 1992.

[117] B. Wohlberg, "Admm penalty parameter selection by residual balancing," 2017, *arXiv:1704.06209*.

[118] T. Goldstein, B. O'Donoghue, S. Setzer, and R. Baraniuk, "Fast alternating direction optimization methods," *SIAM J. Imag. Sci.*, vol. 7, no. 3, pp. 1588–1623, 2014.

[119] L. Ferranti, L. Lyons, R. R. Negenborn, T. Keviczky, and J. Alonso-Mora, "Distributed nonlinear trajectory optimization for multi-robot motion planning," *IEEE Trans. Control Syst. Technol*, vol. 31, no. 2, pp. 809–824, Mar. 2023.

[120] O. Shorinwa, T. Halsted, J. Yu, and M. Schwager, "Distributed optimization methods for multi-robot systems: Part 2—a survey," *IEEE Robot. Autom. Mag.*, vol. 31, no. 3, pp. 154–169, 2024.

[121] R. Ni, Z. Pan, and X. Gao, "Robust multi-robot trajectory optimization using alternating direction method of multiplier," *IEEE Robot. Autom. Lett.*, vol. 7, no. 3, pp. 5950–5957, Mar. 2022.

[122] L. Amatucci, G. Turrisi, A. Bratta, V. Barasuol, and C. Semini, "Accelerating model predictive control for legged robots through distributed optimization," 2024, *arXiv:2403.11742*.

[123] A. Aydinoglu and M. Posa, "Real-time multi-contact model predictive control via admm," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2022, pp. 3414–3421.

[124] S. L. Cleac'h and Z. Manchester, "Fast solution of optimal control problems with l1 cost," in *Proc. AAS/AIAA Astrodyn. Spec. Conf.*, 2019, vol. 904, pp. 1–11.

[125] Z. Zhao, Z. Zhou, M. Park, and Y. Zhao, "Sydebo:Symbolic-decision-embedded bilevel optimization for long-horizon manipulation in dynamic environments," *IEEE Access*, vol. 9, pp. 128817–128826, 2021.

[126] L. Wijayarathne, Z. Zhou, Y. Zhao, and F. L. Hammond, "Real-time deformable-contact-aware model predictive control for force-modulated manipulation," *IEEE Trans. Robot.*, vol. 39, no. 5, pp. 3549–3566, Oct. 2023.

[127] H. Li, R. J. Frei, and P. M. Wensing, "Model hierarchy predictive control of robotic systems," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 3373–3380, Feb. 2021.

[128] C. Khazoom, S. Heim, D. Gonzalez-Diaz, and S. Kim, "Optimal scheduling of models and horizons for model hierarchy predictive control," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2023, pp. 9952–9958.

[129] A. Herzog, S. Schaal, and L. Righetti, "Structured contact force optimization for kino-dynamic motion generation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2016, pp. 2703–2710.

[130] Z. Zhou, B. Wingo, N. Boyd, S. Hutchinson, and Y. Zhao, "Momentum-aware trajectory optimization and control for agile quadrupedal locomotion," *IEEE Robot. Autom. Lett.*, vol. 7, no. 3, pp. 7755–7762, Mar. 2022.

[131] R. Budhiraja, J. Carpentier, and N. Mansard, "Dynamics consensus between centroidal and whole-body models for locomotion of legged robots," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2019, pp. 6727–6733.

[132] A. Meduri, P. Shah, J. Viereck, M. Khadiv, I. Havoutis, and L. Righetti, "Biconmp: A nonlinear model predictive control framework for whole body motion planning," *IEEE Trans. Robot*, vol. 39, no. 2, pp. 905–922, Apr. 2023.

[133] Z. Zhou and Y. Zhao, "Accelerated ADMM based trajectory optimization for legged locomotion with coupled rigid body dynamics," in *Proc. Amer. Control Conf. IEEE*, 2020, pp. 5082–5089.

[134] C. Finn, S. Levine, and P. Abbeel, "Guided cost learning: Deep inverse optimal control via policy optimization," in *Proc. Int. Conf. Mach. Learn. PMLR*, 2016, pp. 49–58.

[135] P. Englert, N. A. Vien, and M. Toussaint, "Inverse KKT: Learning cost functions of manipulation tasks from demonstrations," *Int. J. Robot. Res.*, vol. 36, no. 13-14, pp. 1474–1488, 2017.

[136] P. Sharma et al., "Correcting robot plans with natural language feedback," in *Proc. Robot. Sci. Syst.*, 2022.

[137] W. Yu et al., "Language to rewards for robotic skill synthesis," in *Proc. Conf. Robot. Learn. PMLR*, 2023, pp. 374–404.

[138] W. Huang, C. Wang, R. Zhang, Y. Li, J. Wu, and L. Fei-Fei, "Voxposer: Composable 3D value maps for robotic manipulation with language models," in *Proc. Conf. Robot. Learn. PMLR*, 2023, pp. 540–562.

[139] M. Parmar, M. Halm, and M. Posa, "Fundamental challenges in deep learning for stiff contact dynamics," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 5181–5188.

[140] S. Pfrommer, M. Halm, and M. Posa, "Contactnets: Learning discontinuous contact dynamics with smooth, implicit representations," in *Proc. Conf. Robot. Learn. PMLR*, 2021, pp. 2279–2291.

[141] B. Bianchini, M. Halm, and M. Posa, "Simultaneous learning of contact and continuous dynamics," in *Proc. Conf. Robot. Learn. PMLR*, 2023, pp. 3966–3978.

[142] S. L. Cleac'h, et al., "Differentiable physics simulation of dynamics-augmented neural objects," *IEEE Robot. Autom. Lett.*, vol. 8, no. 5, pp. 2780–2787, May 2023.

[143] D. Driess, J.-S. Ha, M. Toussaint, and R. Tedrake, "Learning models as functionals of signed-distance fields for manipulation planning," in *Proc. Conf. Robot. Learn. PMLR*, 2022, pp. 245–255.

[144] S. Levine and V. Koltun, "Guided policy search," in *Proc. Int. Conf. Mach. Learn. PMLR*, 2013, pp. 1–9.

[145] S. Levine and V. Koltun, "Learning complex neural network policies with trajectory optimization," in *Proc. Int. Conf. Mach. Learn. PMLR*, 2014, pp. 829–837.

[146] A. Duburcq, Y. Chevaleyre, N. Bredeche, and G. Boéris, "Online trajectory planning through combined trajectory optimization and function approximation: Application to the exoskeleton atalante," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2020, pp. 3756–3762.

[147] Z. Zhao, S. Zuo, T. Zhao, and Y. Zhao, "Adversarially regularized policy learning guided by trajectory optimization," in *Proc. Learn. Dyn. Control Conf. PMLR*, 2022, pp. 844–857.

[148] M. J. Bency, A. H. Qureshi, and M. C. Yip, "Neural path planning: Fixed time, near-optimal path generation via oracle imitation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019, pp. 3965–3972.

[149] A. H. Qureshi, J. Dong, A. Choe, and M. C. Yip, "Neural manipulation planning on constraint manifolds," *IEEE Robot. Autom. Lett.*, vol. 5, no. 4, pp. 6089–6096, Apr. 2020.

[150] I. Radosavovic, T. Xiao, B. Zhang, T. Darrell, J. Malik, and K. Sreenath, "Real-world humanoid locomotion with reinforcement learning," *Sci. Robot.*, vol. 9, no. 89, 2024, Art. no. eadi9579.

[151] C. Chi et al., "Diffusion policy: Visuomotor policy learning via action diffusion," in *Proc. Robot. Sci. Syst.*, 2023.

[152] J. Viereck and L. Righetti, "Learning a centroidal motion planner for legged locomotion," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2021, pp. 4905–4911.

[153] X. Lin, G. I. Fernandez, Y. Liu, T. Zhu, Y. Shirai, and D. Hong, "Multimodal multi-agent optimization for limms, a modular robotics approach to delivery automation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2022, pp. 12674–12681.

[154] Z. Gu, R. Guo, W. Yates, Y. Chen, Y. Zhao, and Y. Zhao, "Walking-by-logic: Signal temporal logic-guided model predictive control for bipedal locomotion resilient to external perturbations," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2024, pp. 1121–1127.

[155] M. Toussaint and M. Lopes, "Multi-bound tree search for logic-geometric programming in cooperative manipulation domains," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2017, pp. 4044–4051.

[156] S.-Y. Lo, S. Zhang, and P. Stone, "The petlon algorithm to plan efficiently for task-level-optimal navigation," *J. Artif. Intell. Res.*, vol. 69, pp. 471–500, 2020.

[157] E. M. Wolff, U. Topcu, and R. M. Murray, "Optimization-based trajectory generation with linear temporal logic specifications," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2014, pp. 5319–5325.

[158] J. Chen, B. C. Williams, and C. Fan, "Optimal mixed discrete-continuous planning for linear hybrid systems," in *Proc. Int. Conf. Hybrid Syst.: Comput. Control*, 2021, pp. 1–12.

[159] T. Kogo, K. Takaya, and H. Oyama, "Fast MILP-based task and motion planning for pick-and-place with hard/soft constraints of collision-free route," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, 2021, pp. 1020–1027.

[160] M. Katayama, S. Tokuda, M. Yamakita, and H. Oyama, "Fast LTL-based flexible planning for dual-arm manipulation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 6605–6612.

[161] S. Saha and A. A. Julius, "Task and motion planning for manipulator arms with metric temporal logic specifications," *IEEE Robot. Autom. Lett.*, vol. 3, no. 1, pp. 379–386, Jan. 2017.

[162] Y. V. Pant, H. Abbas, R. A. Quaye, and R. Mangharam, "Fly-by-logic: Control of multi-drone fleets with temporal logic objectives," in *Proc. ACM/IEEE Int. Conf. Cyber-Phys. Syst.*, 2018, pp. 186–197.

[163] F. Bacchus and Q. Yang, "Downward refinement and the efficiency of hierarchical problem solving," *Artif. Intell.*, vol. 71, no. 1, pp. 43–100, 1994.

[164] A. Akbari, F. Lagriffoul, and J. Rosell, "Combined heuristic task and motion planning for bi-manual robots," *Auton. Robots*, vol. 43, no. 6, pp. 1575–1590, 2019.

[165] A. Agostini and J. Piater, "Unified task and motion planning using object-centric abstractions of motion constraints," 2023, *arXiv:2312.17605*.

[166] S. Srivastava, E. Fang, L. Riano, R. Chitnis, S. Russell, and P. Abbeel, "Combined task and motion planning through an extensible planner-independent interface layer," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2014, pp. 639–646.

[167] K. Hauser and J.-C. Latombe, "Multi-modal motion planning in non-expansive spaces," *Int. J. Robot. Res.*, vol. 29, no. 7, pp. 897–915, 2010.

[168] Z. Kingston and L. E. Kavraki, "Scaling multimodal planning: Using experience and informing discrete search," *IEEE Trans. Robot.*, vol. 39, no. 1, pp. 128–146, Jan. 2022.

[169] M. Toussaint, "A tutorial on Newton methods for constrained trajectory optimization and relations to SLAM, Gaussian process smoothing, optimal control, and probabilistic inference," *Geom. Numer. Found. Mov.*, pp. 361–392, 2017.

[170] M. Toussaint, J.-S. Ha, and D. Driess, "Describing physics for physical reasoning: Force-based sequential manipulation planning," *IEEE Robot. Autom. Lett.*, vol. 5, no. 4, pp. 6209–6216, Apr. 2020.

[171] S. Zimmermann, G. Hakimifard, M. Zamora, R. Poranne, and S. Coros, "A multi-level optimization framework for simultaneous grasping and motion planning," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 2966–2972, Feb. 2020.

[172] M. S. Phoon, P. S. Schmitt, and G. v. Wichert, "Constraint-based task specification and trajectory optimization for sequential manipulation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2022, pp. 197–202.

[173] M. Schwenzer, M. Ay, T. Bergs, and D. Abel, "Review on model predictive control: An engineering perspective," *Int. J. Adv. Manuf. Technol.*, vol. 117, no. 5, pp. 1327–1349, 2021.

[174] V. N. Hartmann, O. S. Oguz, D. Driess, M. Toussaint, and A. Menges, "Robust task and motion planning for long-horizon architectural construction planning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 6886–6893.

[175] N. Castaman, E. Pagello, E. Menegatti, and A. Pretto, "Receding horizon task and motion planning in changing environments," *Robot. Auton. Syst.*, vol. 145, 2021, Art. no. 103863.

[176] C. V. Braun, J. Ortiz-Haro, M. Toussaint, and O. S. Oguz, "RHH-LGP: Receding horizon and heuristics-based logic-geometric programming for task and motion planning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2022, pp. 13761–13768.

[177] Y. Chen, U. Rosolia, W. Ubellacker, N. Csomay-Shanklin, and A. D. Ames, "Interactive multi-modal motion planning with branch model predictive control," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 5365–5372, Feb. 2022.

[178] E. L. Lawler and D. E. Wood, "Branch-and-bound methods: A survey," *Oper. Res.*, vol. 14, no. 4, pp. 699–719, 1966.

[179] L. Huang et al., "Branch and bound in mixed integer linear programming problems: A survey of techniques and trends," 2021, *arXiv:2111.06257*.

[180] A. Schrijver et al. *Combinatorial Optimization: Polyhedra and Efficiency*. Berlin, Germany: Springer, 2003.

[181] T. Berthold, "Primal heuristics for mixed integer programs," Ph.D. dissertation, Zuse Institute Berlin (ZIB), Berlin, Germany, 2006.

[182] S. S. Dey, Y. Dubey, M. Molinaro, and P. Shah, "A theoretical and computational analysis of full strong-branching," *Math. Prog.*, vol. 205, pp. 1–34, 2023.

[183] M. Fischetti and A. Lodi, "Local branching," *Math. Prog.*, vol. 98, pp. 23–47, 2003.

[184] Gurobi Optimization, LLC, "Gurobi optimizer reference manual," 2023. [Online]. Available: https://www.gurobi.com

[185] M. ApS, "The MOSEK optimization toolbox for MATLAB manual. version 9.0.," 2019. [Online]. Available: http://docs.mosek.com/9.0/toolbox/index.html

[186] T. M. Inc., "Matlab version: 9. 13.0 (r2022b)," Natick, Massachusetts, United States, 2022. [Online]. Available: https://www.mathworks.com

[187] N. Funk, S. Menzenbach, G. Chalvatzaki, and J. Peters, "Graph-based reinforcement learning meets mixed integer programs: An application to 3 d robot assembly discovery," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2022, pp. 10215–10222.

[188] A. Shamsah, Z. Gu, J. Warnke, S. Hutchinson, and Y. Zhao, "Integrated task and motion planning for safe legged navigation in partially observable environments," *IEEE Trans. Robot*, vol. 39, no. 6, pp. 4913–4934, Dec. 2023.

[189] J. Warnke, A. Shamsah, Y. Li, and Y. Zhao, "Towards safe locomotion navigation in partially observable environments with uneven terrain," in *Proc. IEEE Conf. Decis. Control*, 2020, pp. 958–965.

[190] Y. Shirai et al., "Simultaneous contact-rich grasping and locomotion via distributed optimization enabling free-climbing for multi-limbed robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2022, pp. 13563–13570.

[191] Y. V. Pant, H. Abbas, and R. Mangharam, "Smooth operator: Control using the smooth robustness of temporal logic," in *Proc. IEEE Conf. Control Technol. Appl.*, 2017, pp. 1235–1240.

[192] N. Mehdipour, C.-I. Vasile, and C. Belta, "Arithmetic-geometric mean robustness for control from signal temporal logic specifications," in *Proc. Amer. Control Conf.*, 2019, pp. 1690–1695.

[193] 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, Jan. 2020.

[194] Z. Gu et al., "Robust-locomotion-by-logic: Perturbation-resilient bipedal locomotion via signal temporal logic guided model predictive control," 2024, *arXiv:2403.15993*.

[195] D. Sun, J. Chen, S. Mitra, and C. Fan, "Multi-agent motion planning from signal temporal logic specifications," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 3451–3458, Feb. 2022.

[196] S. S. Farahani, V. Raman, and R. M. Murray, "Robust model predictive control for signal temporal logic synthesis," *Int. Fed. Autom. Control*, vol. 48, no. 27, pp. 323–328, 2015.

[197] S. Sadraddini and C. Belta, "Robust temporal logic model predictive control," in *Proc. Annu. Allerton Conf. Commun., Control, Comput.*, 2015, pp. 772–779.

[198] D. Sadigh and A. Kapoor, "Safe control under uncertainty with probabilistic signal temporal logic," in *Proc. Robot. Sci. Syst.*, 2016.

[199] L. P. Kaelbling and T. Lozano-Pérez, "Hierarchical task and motion planning in the now," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2011, pp. 1470–1477.

[200] A. M. Wells, N. T. Dantam, A. Shrivastava, and L. E. Kavraki, "Learning feasibility for task and motion planning in tabletop environments," *IEEE Robot. Autom. Lett.*, vol. 4, no. 2, pp. 1255–1262, Feb. 2019.

[201] M. Noseworthy et al., "Active learning of abstract plan feasibility," in *Proc. Robot. Sci. Syst.*, 2021.

[202] A. Mandlekar, C. Garrett, D. Xu, and D. Fox, "Human-in-the-loop task and motion planning for imitation learning," in *Proc. Conf. Robot. Learn. PMLR*, 2023, 3030–300.

[203] T. Silver, A. Athalye, J. B. Tenenbaum, T. Lozano-Perez, and L. P. Kaelbling, "Learning neuro-symbolic skills for bilevel planning," in *Proc. Conf. Robot. Learn. PMLR*, 2022, pp. 701–714.

[204] A. Cauligi, P. Culbertson, B. Stellato, D. Bertsimas, M. Schwager, and M. Pavone, "Learning mixed-integer convex optimization strategies for robot planning and control," in *Proc. IEEE Conf. Decis. Control.*, 2020, pp. 1698–1705.

[205] Y. Sung, Z. Wang, and P. Stone, "Learning to correct mistakes: Backjumping in long-horizon task and motion planning," in *Proc. Conf. Robot. Learn. PMLR*, 2023, pp. 2115–2124.

[206] A. Curtis, X. Fang, L. P. Kaelbling, T. Lozano-Pérez, and C. R. Garrett, "Long-horizon manipulation of unknown objects via task and motion planning with estimated affordances," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2022, pp. 1940–1946.

[207] Z. Wang, C. R. Garrett, L. P. Kaelbling, and T. Lozano-Pérez, "Active model learning and diverse action sampling for task and motion planning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 4107–4114.

[208] Z. Wang, C. R. Garrett, L. P. Kaelbling, and T. Lozano-Pérez, "Learning compositional models of robot skills for task and motion planning," *Int. J. Robot. Res.*, vol. 40, no. 6–7, pp. 866–894, 2021.

[209] B. Kim, L. Kaelbling, and T. Lozano-Pérez, "Guiding search in continuous state-action spaces by learning an action sampler from off-target search experience," in *Proc. AAAI Conf. Artif. Intell.*, 2018, pp. 6509–6516.

[210] B. Kim, Z. Wang, L. P. Kaelbling, and T. Lozano-Pérez, "Learning to guide task and motion planning using score-space representation," *Int. J. Robot. Res.*, vol. 38, no. 7, pp. 793–812, 2019.

[211] R. Chitnis et al., "Guided search for task and motion plans using learned heuristics," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2016, pp. 447–454.

[212] B. Kim and L. Shimanuki, "Learning value functions with relational state representations for guiding task-and-motion planning," in *Proc. Conf. Robot. Learn. PMLR*, 2020, pp. 955–968.

[213] J. Ortiz-Haro, V. N. Hartmann, O. S. Oguz, and M. Toussaint, "Learning efficient constraint graph sampling for robotic sequential manipulation," in, *Proc. IEEE Int. Conf. Robot. Autom.*, 2021, pp. 4606–4612.

[214] J. Ortiz-Haro, J.-S. Ha, D. Driess, and M. Toussaint, "Structured deep generative models for sampling on constraint manifolds in sequential manipulation," in *Proc. Conf. Robot. Learn. PMLR*, 2022, pp. 213–223.

[215] D. B. Noureddine, A. Gharbi, and S. B. Ahmed, "Multi-agent deep reinforcement learning for task allocation in dynamic environment," in *Proc. Int. Conf. Softw. Technol.*, 2017, pp. 17–26.

[216] S. Ding, D. Lin, and X. Zhou, "Graph convolutional reinforcement learning for dependent task allocation in edge computing," in *Proc. IEEE Int. Conf. Agents.*, 2021, pp. 25–30.

[217] C. Shyalika, T. Silva, and A. Karunananda, "Reinforcement learning in dynamic task scheduling: A review," *SN Comput. Sci.*, vol. 1, no. 6, 2020, Art. no. 306.

[218] Z. Li, K. Yu, S. Cheng, and D. Xu, "League: Empowering continual robot learning through guided skill acquisition with large language models," in *Proc. Int. Conf. Learn. Represent.,* 2024.

[219] Y. Meng and C. Fan, "Signal temporal logic neural predictive control," *IEEE Robot. Autom. Lett*, vol. 8, no. 11, pp. 7719–7726, Nov. 2023.

[220] T. Pan, A. M. Wells, R. Shome, and L. E. Kavraki, "Failure is an option: Task and motion planning with failing executions," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2022, pp. 1947–1953.

[221] A. S. Wang and O. Kroemer, "Learning robust manipulation strategies with multimodal state transition models and recovery heuristics," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2019, pp. 1309–1315.

[222] S. Luo, H. Wu, S. Duan, Y. Lin, and J. Rojas, "Endowing robots with longer-term autonomy by recovering from external disturbances in manipulation through grounded anomaly classification and recovery policies," *J. Intell. Robot. Syst.*, vol. 101, pp. 1–40, 2021.

[223] X. Zhang et al., "Grounding classical task planners via vision-language models," in *Proc. ICRA Workshop Robot Exec. Failures Fail. Manag. Strateg.*, 2023.

[224] Z. Liu, A. Bahety, and S. Song, "Reflect: Summarizing robot experiences for failure explanation and correction," in *Proc. Conf. Robot. Learn. PMLR*, 2023, pp. 3468–3484.

[225] K. Rana, J. Haviland, S. Garg, J. Abou-Chakra, I. Reid, and N. Suenderhauf, "Sayplan: Grounding large language models using 3 d scene graphs for scalable robot task planning," in *Proc. Conf. Robot. Learn. PMLR*, 2023, pp. 23–72.

[226] J. Zhang et al., "A survey for solving mixed integer programming via machine learning," *Neurocomputing*, vol. 519, pp. 205–217, 2023.

[227] Y. Bengio, A. Lodi, and A. Prouvost, "Machine learning for combinatorial optimization: A methodological tour d'horizon," *Eur. J. Oper. Res.*, vol. 290, no. 2, pp. 405–421, 2021.

[228] A M Alvarez, Q. Louveaux, and L. Wehenkel, "A supervised machine learning approach to variable branching in branch-and-bound," Dept. EE&CS, Universite de Liege, Liege, Belgium, Tech. Rep., 2014.

[229] E. Khalil, P. Le Bodic, L. Song, G. Nemhauser, and B. Dilkina, "Learning to branch in mixed integer programming," in *Proc. AAAI Conf. Artif. Intell.*, 2016, pp. 724–731.

[230] M. Gasse, D. Chételat, N. Ferroni, L. Charlin, and A. Lodi, "Exact combinatorial optimization with graph convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, vol. 32, pp. 15580–15592.

[231] J. Song et al., "A general large neighborhood search framework for solving integer linear programs," *Adv. Neural Inf. Process. Syst.*, vol. 33, pp. 20012–20023, 2020.

[232] N. Sonnerat, P. Wang, I. Ktena, S. Bartunov, and V. Nair, "Learning a large neighborhood search algorithm for mixed integer programs," 2021, *arXiv:2107.10201*.

[233] D. Liu, M. Fischetti, and A. Lodi, "Learning to search in local branching," in *Proc. AAAI Conf. Artif. Intell.*, 2022, pp. 3796–3803.

[234] M. Srinivasan, A. Chakrabarty, R. Quirynen, N. Yoshikawa, T. Mariyama, and S. Di Cairano, "Fast multi-robot motion planning via imitation learning of mixed-integer programs," *Int. Fed. Autom. Control*, vol. 54, no. 20, pp. 598–604, 2021.

[235] A. Cauligi, P. Culbertson, E. Schmerling, M. Schwager, B. Stellato, and M. Pavone, "CoCo: Online mixed-integer control via supervised learning," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 1447–1454, Feb. 2021.

[236] R. Deits, T. Koolen, and R. Tedrake, "LVIS: Learning from value function intervals for contact-aware robot controllers," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2019, pp. 7762–7768.

[237] A. Ajay et al., "Compositional foundation models for hierarchical planning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2024, vol. 36, pp. 22304–22325.

[238] Y. Qiu, Z. Zhao, Y. Ziser, A. Korhonen, E. Ponti, and S. B. Cohen, "Are large language model temporally grounded?," in *Proc. Conf. North Amer. Chap. Assoc. Comput. Linguist.: Hum. Lang. Technol.*, 2024, pp. 7057–7076.

[239] X. L. Li, A. Kuncoro, J. Hoffmann, C. de Masson d'Autume, P. Blunsom, and A. Nematzadeh, "A systematic investigation of commonsense knowledge in large language models," in *Proc. Conf. Empir. Methods Nat. Lang. Process.*, 2022, pp. 11838–11855.

[240] J. Chen, W. Shi, Z. Fu, S. Cheng, L. Li, and Y. Xiao, "Say what you mean! large language models speak too positively about negative commonsense knowledge," in *Proc. Annu. Meet. Assoc. Comput. Linguist.*, 2023, pp. 9890–9908.

[241] B. Chen et al., "SpatialVLM: Endowing vision-language models with spatial reasoning capabilities," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2024, pp. 14455–14465.

[242] J. Rocamonde, V. Montesinos, E. Nava, E. Perez, and D. Lindner, "Vision-language models are zero-shot reward models for reinforcement learning," in *Proc. Int. Conf. Learn. Represent.*, 2023.

[243] L. Chen et al., "Large language models are visual reasoning coordinators," in *Proc. Adv. Neural Inf. Process. Syst.*, 2024, vol. 36, pp. 70115–70140.

[244] L. Liu et al., "Think-in-memory: Recalling and post-thinking enable llms with long-term memory," 2023, *arXiv:2311.08719*.

[245] W. Wang et al., "Augmenting language models with long-term memory," in *Proc. Adv. Neural Inf. Process. Syst.*, 2024, vol. 36, pp. 74530–74543.

[246] S. S. Kannan, V. L. Venkatesh, and B.-C. Min, "Smart-LLM: Smart multi-agent robot task planning using large language models," 2023, *arXiv:2309.10062*.

[247] J. Yang, H. Zhang, F. Li, X. Zou, C. Li, and J. Gao, "Set-of-mark prompting unleashes extraordinary visual grounding in gpt-4v," 2023, *arXiv:2310.11441*.

[248] B. Chen et al., "Open-vocabulary queryable scene representations for real world planning," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2023, pp. 11509–11522.

[249] L. Wang et al., "GenSim: Generating robotic simulation tasks via large language models," in *Proc. Int. Conf. Learn. Represent.*, 2023.

[250] M. Janner, Y. Du, J. Tenenbaum, and S. Levine, "Planning with diffusion for flexible behavior synthesis," in *Proc. Int. Conf. Mach. Learn. PMLR*, 2022, pp. 9902–9915.

[251] A. Ajay, Y. Du, A. Gupta, J. Tenenbaum, T. Jaakkola, and P. Agrawal, "Is conditional generative modeling all you need for decision-making?," 2022, *arXiv:2211.15657*.

[252] Y. Ze, G. Zhang, K. Zhang, C. Hu, M. Wang, and H. Xu, "3 d diffusion policy: Generalizable visuomotor policy learning via simple 3 d representations," in *Proc. Robot. Sci. Syst.*, 2024.

[253] C. Pan, Z. Yi, G. Shi, and G. Qu, "Model-based diffusion for trajectory optimization," 2024, Accessed:Jun. 5, 2024. [Online]. Available: https://lecar-lab.github.io/mbd/

[254] U. A. Mishra, S. Xue, Y. Chen, and D. Xu, "Generative skill chaining: Long-horizon skill planning with diffusion models," in *Proc. Conf. Robot. Learn. PMLR*, 2023, pp. 2905–2925.

[255] X. Fang, C. Garrett, C. Eppner, T. Lozano-Pérez, L. Kaelbling, and D. Fox, "Dimsam: Diffusion models as samplers for task and motion planning under partial observability," in *Proc. CoRL 2023 Workshop Learn. Effective Abstr. Plan.*, 2023.

[256] M. A. Lee et al., "Making sense of vision and touch: Learning multimodal representations for contact-rich tasks," *IEEE Trans. Robot.*, vol. 36, no. 3, pp. 582–596, Mar. 2020.

[257] K. Yu, Y. Han, M. Zhu, and Y. Zhao, "Mimictouch:Learning human's control strategy with multi-modal tactile feedback," in *Proc. NeurIPS Workshop Touch Process.: New Sens. Modality AI*, 2023.

[258] H. Li et al., "See, hear, and feel: Smart sensory fusion for robotic manipulation," in *Proc. Conf. Robot. Learn. PMLR*, 2023, pp. 1368–1378.

[259] M. T. Mason, "Toward robotic manipulation," *Annu. Rev. Control Robot. Auton. Syst.*, vol. 1, pp. 1–28, 2018.

[260] Z. Zhang, J. Yan, X. Kong, G. Zhai, and Y. Liu, "Efficient motion planning based on kinodynamic model for quadruped robots following persons in confined spaces," *IEEE/ASME Trans. Mechatron.*, vol. 26, no. 4, pp. 1997–2006, Apr. 2021.

[261] C. McGreavy and Z. Li, "Reachability map for diverse and energy efficient stepping of humanoids," *IEEE/ASME Trans. Mechatron.*, vol. 27, no. 6, pp. 5307–5317, Jun. 2022.

[262] C. Sferrazza, D.-M. Huang, X. Lin, Y. Lee, and P. Abbeel, "Humanoidbench: Simulated humanoid benchmark for whole-body locomotion and manipulation," in *Proc. Robot. Sci. Syst.*, 2024.

[263] W. Liu, X. Liang, and M. Zheng, "Task-constrained motion planning considering uncertainty-informed human motion prediction for human–robot collaborative disassembly," *IEEE/ASME Trans. Mechatron.*, vol. 28, no. 4, pp. 2056–2063, Apr. 2023.

[264] Y. Cheng, L. Sun, and M. Tomizuka, "Human-aware robot task planning based on a hierarchical task model," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 1136–1143, Feb. 2021.

[265] K. Darvish, E. Simetti, F. Mastrogiovanni, and G. Casalino, "A hierarchical architecture for human–robot cooperation processes," *IEEE Trans. Robot.*, vol. 37, no. 2, pp. 567–586, Feb. 2020.

[266] M. Faroni, A. Umbrico, M. Beschi, A. Orlandini, A. Cesta, and N. Pedrocchi, "Optimal task and motion planning and execution for multiagent systems in dynamic environments," *IEEE Trans. Cybern*, vol. 54, no. 6, pp. 3366–3377, Jun. 2024.

[267] A. T. Le, P. Kratzer, S. Hagenmayer, M. Toussaint, and J. Mainprice, "Hierarchical human-motion prediction and logic-geometric programming for minimal interference human-robot tasks," in *Proc. IEEE Int. Conf. Robot. Hum. Interact. Commun.*, 2021, pp. 7–14.

[268] C. Zhang, J. Chen, J. Li, Y. Peng, and Z. Mao, "Large language models for human-robot interaction: A review," *Biomimetic Intell. Robot.*, vol. 3, no. 4, 2023, Art. no. 100131.

[269] A. Kshirsagar, H. Kress-Gazit, and G. Hoffman, "Specifying and synthesizing human-robot handovers," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019, pp. 5930–5936.

[270] A. Mörtl, M. Lawitzky, A. Kucukyilmaz, M. Sezgin, C. Basdogan, and S. Hirche, "The role of roles: Physical cooperation between humans and robots," *Int. J. Robot. Res.*, vol. 31, no. 13, pp. 1656–1674, 2012.

[271] A. Otto, N. Agatz, J. Campbell, B. Golden, and E. Pesch, "Optimization approaches for civil applications of unmanned aerial vehicles (UAVs) or aerial drones: A survey," *Networks*, vol. 72, no. 4, pp. 411–458, 2018.

[272] J. Conesa-Muñoz, J. M. Bengochea-Guevara, D. Andujar, and A. Ribeiro, "Route planning for agricultural tasks: A general approach for fleets of autonomous vehicles in site-specific herbicide applications," *Comput. Electron. Agric.*, vol. 127, pp. 204–220, 2016.

[273] J. Siburian, C. C. Beltran-Hernandez, and M. Hamaya, "Integrated task and motion planning for real-world cooking tasks," in *Proc. IEEE Int. Conf. Robot. Autom. 2024 Workshop Cook. Robot.: Percept. Motion Plann*, 2024.

[274] W. Wan, T. Kotaka, and K. Harada, "Arranging test tubes in racks using combined task and motion planning," *Robot. Auton. Syst.*, vol. 147, 2022, Art. no. 103918.

[275] P. Lin, K. Abney, and G. A. Bekey, *Robot Ethics: The Ethical and Social Implications of Robotics*. Cambridge, MA, USA: MIT press, 2014.

[276] C.-J. Wu et al., "Sustainable ai: Environmental implications, challenges and opportunities," *Proc. Mach. Learn. Syst.*, vol. 4, pp. 795–813, 2022.

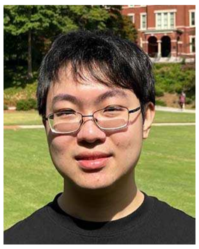
[277] A. Sinha, P. Malo, and K. Deb, "A review on bilevel optimization: From classical to evolutionary approaches and applications," *IEEE Trans. Evol. Comput.*, vol. 22, no. 2, pp. 276–295, Feb. 2017.

[278] R. Bommasani et al., "On the opportunities and risks of foundation models," 2021, *arXiv:2108.07258*.

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

Zhigen Zhao received the B.S. degree in mechanical engineering in 2018 from the Georgia Institute of Technology, Atlanta, GA, USA, where he is currently working toward the Ph.D. degree in robotics.

His research interests include task and motion planning, trajectory optimization, combining model-based and learning approaches for robot planning, and applications involving robot manipulation and loco-manipulation tasks, where he aims to advance the capabilities of robotic systems in complex, dynamic environments.



Shuo Cheng is currently working toward the Ph.D. degree in computer science with the Georgia Institute of Technology, Atlanta, GA, USA.

To approach this research objective, he studies novel representations for efficient, generalizable, and scalable robot skill acquisition and develop systems that compose and reuse the skills robustly, by exploring techniques such as task and motion planning, imitation learning, reinforcement learning, and visual perception learning. His research focuses on enabling robots with the ability to reason and perform in complex and highly variable environments for achieving long-horizon tasks.

Dr. Cheng has been nominated for Best Paper at IEEE RA-L.

Yan Ding (Student Member, IEEE) received the bachelor's degrees in mechanical engineering and master's degree in computer science from Chongqing University, Chongqing, China, in 2016 and 2019, respectively, and the Ph.D. degree in computer science from the State University of New York at Binghamton, Binghamton, NY, USA, in 2024.

He is currently a Researcher with Shanghai AI Laboratory, Shanghai, China. His research focuses on spatial intelligence for robotics, where he aims to empower robots with the capability to understand and interact with the real world, and skill learning for robotics, focusing on enabling robots to effectively transform the real world.

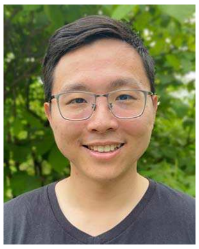
Ziyi Zhou received the B.S. degree in automation from Northeastern University, Shenyang, China, in 2018, and the M.S. degree in electrical and computer engineering in 2021 from the Georgia Institute of Technology, Atlanta, GA, USA, where he is currently working toward the Ph.D. degree in electrical and computer engineering, with a focus on robotics.

His research interests include contact-rich trajectory optimization and task planning applied to legged robots.

Shiqi Zhang (Member, IEEE) received the B.S. and M.S. degrees from the Harbin Institute of Technology, Harbin, China, in 2006 and 2008, respectively, and the Ph.D. degree in computer science from Texas Tech University, Lubbock, TX, USA, in 2013.

He is currently an Associate Professor with the School of Computing, State University of New York at Binghamton, Binghamton, NY, USA. He was an Assistant Professor with Cleveland State University, Cleveland, OH, USA, and before that was a Postdoctoral Fellow with the University of Texas at Austin, Austin, TX.

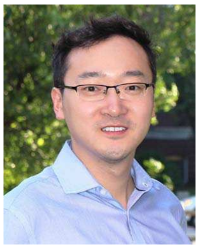
Dr. Zhang was the recipient of the AAMAS-2018 Best Robotics Paper Award, Ford URP Award in 2019, OPPO Faculty Research Award in 2020, Top Cited Article recognition from AI Magazine in 2023 and Outstanding Associate Editor recognition from IEEE Robotics and Automation Letters in 2024. He served on the organizing committees of the AAMAS-2022 and KR-2023 conferences.



Danfei Xu received the Ph.D. degree in computer science from Stanford University, Stanford, CA, USA, in 2021.

He is currently an Assistant Professor with the School of Interactive Computing, Georgia Institute of Technology, Atlanta, GA, USA. His research interests include machine learning methods for robotics, with a focus on manipulation planning and imitation learning, and to enable physical autonomy in everyday human environments with minimum expert intervention.

Dr. Xu is an Associate Editor for *International Journal on Robotics Research*. He was named as a 2022 DARPA Riser. His work has been nominated for Best Paper at CoRL and IEEE RA-L.



Ye Zhao (Senior Member, IEEE) received the Ph.D. degree in mechanical engineering from The University of Texas at Austin, Austin, TX, USA, in 2016.

He is currently an Assistant Professor with the George W. Woodruff School of Mechanical Engineering, Georgia Institute of Technology, Atlanta, GA, USA. He was a Postdoctoral Fellow with the John A. Paulson School of Engineering and Applied Sciences, Harvard University, Cambridge, MA, USA. His research interests include robust task and motion planning, contact-rich trajectory optimization, formal methods for legged locomotion and navigation.

Dr. Zhao is an Associate Editor of IEEE TRANSACTIONS ON ROBOTICS, IEEE/ASME TRANSACTIONS ON MECHATRONICS, IEEE ROBOTICS AND AUTOMATION LETTERS, and IEEE CONTROL SYSTEMS LETTERS. He was the recipient of the George W. Woodruff School Faculty Research Award at Georgia Tech in 2023, NSF CAREER Award in 2022, and ONR YIP Award in 2023.