Recent Trends in Task and Motion Planning for Robotics: A Survey

HUIHUI GUO, FAN WU, YUNCHUAN QIN, and RUIHUI LI, Hunan University, China
KEQIN LI, State University of New York, USA
KENLI LI, Hunan University, China

Autonomous robots are increasingly served in real-world unstructured human environments with complex long-horizon tasks, such as restaurant serving and office delivery. Task and motion planning (TAMP) is a recent research method in Artificial Intelligence Planning for these applications. TAMP integrates high-level abstract reasoning with the low-level geometric feasibility check and thus is more comprehensive than traditional task planning methods. While regular TAMP approaches are challenged by different types of uncertainties and the generalization of various applications when implemented in real-world scenarios. This article systematically reviews the most relevant approaches to TAMP and classifies them according to their features and emphasis; it categorizes the challenges and presents online TAMP and machine learning-based TAMP approaches for addressing them.

CCS Concepts: • **General and reference** → **Surveys and overviews;** • **Computing methodologies** → **Robotic planning**; *Knowledge representation and reasoning*; *Planning under uncertainty*; *Machine learning*;

Additional Key Words and Phrases: Task and motion planning, online planning, learning for planning

ACM Reference format:
Huihui Guo, Fan Wu, Yunchuan Qin, Ruihui Li, Keqin Li, and Kenli Li. 2023. Recent Trends in Task and Motion Planning for Robotics: A Survey. *ACM Comput. Surv.* 55, 13s, Article 289 (July 2023), 36 pages.
https://doi.org/10.1145/3583136

1 INTRODUCTION

Robots equipped with rich perception and mobile manipulation systems have increasingly served in real-world unstructured human environments [29, 104], such as homes, hospitals, and hotels. Given a task in these scenarios, the robot needs to actively sense its surroundings, make decisions, and execute them. For example, if a robot needs to deliver an object to the goal region of another room, then it has to approach the target object, sense its precise location, grasp it, search paths to transport it to the target room, and finally place it in the goal region. Traditional AI planning approaches can reason at a high level for the sequence of actions to achieve the task. However, the

The research was partially funded by the Key R&D program of Hunan Province, China (Grant No. 2022GK2004) and the National Natural Science Foundation of China (Grant No. 62172151).
Authors' addresses: H. Guo, F. Wu (corresponding author), Y. Qin, R. Li, and K. Li (corresponding author), Hunan University, 2 Lushan South Road, Yuelu Qu, Changsha Shi, Hunan Province, China, 410082; emails: {ghh1991, wufan, qinyunchuan, liruihui, lkl}@hnu.edu.cn; K. Li, State University of New York, New Paltz 12561, New York, USA; email: lik@newpaltz.edu.

0360-0300/2023/07-ART289 $15.00
https://doi.org/10.1145/3583136

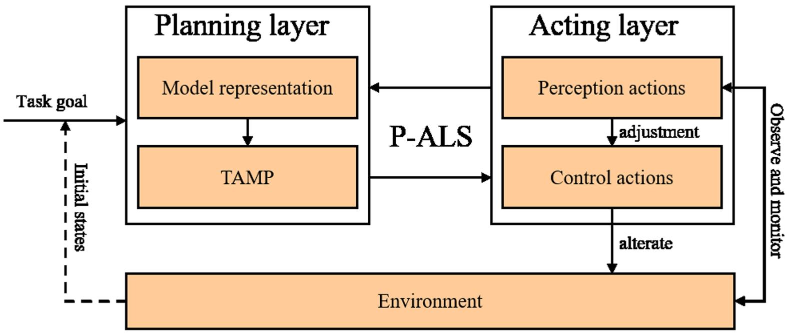

Fig. 1. The architecture of P-ALS.

planned solutions are not comprehensive enough and may fail due to conflicts with the environment, such as collisions.

Task and Motion Planning (TAMP) or Hybrid Planning [14, 36, 125] has received a significant amount of attention from researchers in the AI Planning Community over the past few years. TAMP takes account of geometric constraints while logically reasoning at the task level, it integrates high-level task planning with low-level motion planning to obtain feasible solutions for completing long-horizon tasks. TAMP avoids repetitively planning from scratch by these low-level considerations compared with the traditional planning approaches; however, it also provides concrete input for the motion planner and instructs the implementation of the schedule comprehensively. Although, even if TAMP can provide a near-perfect action plan for the robot, dynamic environments, imperfect models, imperfect perceptions, and imperfect executions still pose great challenges to TAMP in real-world applications [127, 145]. As Wolfgang and Goethe wrote, “To think is easy, to act is hard, The hardest is to act following your thinking” [67]. Besides, TAMP is a highly integrated planning system internally where action sequencing coupled with parameter searching, the computational efforts grow exponentially in the number of related objects; externally, planning is limited by domain knowledge and action library, which are manually defined and coded. Consequently, it is intractable for generalizing TAMP to different tasks due to these internal and external constraints. This survey concludes two main types of challenges when applying TAMP approaches to real-world unstructured human environments, determining (1) how to improve the robustness of TAMP structure and (2) how to afford generalization from TAMP and leverage it.

The AI Planning community has focused on extending the basic TAMP approaches to handle the first type of challenges in belief space [64, 90, 165]. Along with the development of TAMP, the behavior planning approaches with action control structures such as **Hierarchical Task Network (HTN)** [49] and **Behavior Trees (BTs)** [33, 68], they have been applied to robotics from computer game applications. These approaches present control structures for action executions and are capable of handling disturbances reactively to some extent. The next big step in TAMP is combining TAMP with acting as an online closed-loop planning system. There are many factors to be accounted for by the robot during the planning and acting, for example, determining what to do when a grasped target drops from the robot’s hand. Such unexpected events are likely to cause a failure in performing tasks. A comprehensive online planning system should be able to monitor and understand these unexpectations sensitively, and then decide what adjustments need to be made, or replan the task from scratch. The online planning system formulates as a **Planning-Acting Loop System (P-ALS)** interacts with the environment frequently, Figure 1 gives the architecture of P-ALS. Each control action is followed by perceptual action to monitor and estimate current states, which are then compared with the predicted states. The improvements make the internal structure of TAMP more reliable and robust.

In recent years, machine learning has been a useful tool in TAMP to solve the second type of challenges [100, 156, 161]. Geometric constraints are coupled with action sequencing, they affect the search space of TAMP jointly. For example, the poses where the robot manipulates and the physical properties of objects to be manipulated are important information for TAMP. Learning algorithms can extract characteristics of different transitions and relevant constraints from past experience or demonstrations, and then guide TAMP in addressing similar problems as well as expediting the planning process. Besides, learning symbolic primitive operators has been studied for a very long history [6, 7] and implemented for TAMP in recent years [179, 180]. The capability of performing various complex long-horizon tasks optimally with a robot depends on the richness of the operator library to some extent. The learned operators with symbolic representations describe the transition relationships between states and guide the overall planning procedure. Moreover, generalizing TAMP to real-world applications also needs to account for uncertainties during the planning and acting. Reinforcement learning has been used in many decision-making systems and has been presented to solve TAMP problems under uncertainties in recent years [86, 135]. Pioneering work of learning algorithms in TAMP can refine the prior model of task planner with visual data instead of learning from scratch [113, 192]. The majority of relative learning algorithms can be embedded in P-ALS, they can cover each part of P-ALS, aiming to reveal the general features of TAMP and improve the scalability of TAMP in various real-world applications.

We organize this survey according to solutions for the following three Research Questions:

- *RQ1*: What are the strategy inclinations of integrating task planning and motion planning?
- *RQ2*: What are the current tools, strategies and mechanisms used to improve the robustness of TAMP structure internally under uncertainties?
- *RQ3*: How to leverage learning algorithms for generalizing TAMP approaches?

The answers to the three Research Questions summarize the development of TAMP research field. Section 2 offers a historical background of the relevant literature in TAMP. Then, we show the recent trends in regular TAMP approaches and answer *RQ1* according to their concentrations in Section 3. The studies about *RQ1* figure out the basic structure and strategy development process of early TAMP. They provide the technical basis for the follow-up researches about *RQ2* and *RQ3*. We further detail the trends in online TAMP and answer *RQ2* in Section 4. The researches of *RQ2* improve and expand the basic structure of TAMP to deal with uncertainties. Compared with the studies about *RQ1*, they either consider the failure response strategy or added the execution into the planning system to form an effective closed-loop, making TAMP more robust. Section 5 presents different learning algorithms in TAMP and gives solutions for *RQ3*. The approaches in *RQ3* cover the entire structure of TAMP, including domain knowledge representations, primitive action generations, transitions between different states and the planning policies. Learning algorithms are more like external tools for TAMP to help it extract features for generalization. Finally, we conclude this survey and present some promising future research fields of TAMP approaches in Section 6.

2 BACKGROUND

The AI Planning Community has focused on automated solutions for planning and scheduling problems in the last decades. These problems are specified and solved by using a descriptive model called *planning domains*, which is also an approximate representation of environment and actions. The STRIPS [54] is developed with the first systematic classical representations to decide what sequence of commands for controlling the autonomous robot Shakey. The well-known **Planning Domain Definition Language (PDDL)** [2] is the extended representation language for

```
(:action pick
   :parameters     ( ?o - block ?a - arm ?p - pose ?g - grasp ?q - config ?t - traj )
   :precondition   (and (SafeTraj ?a ?o ?p ?g ?q ?t)
                   (AtPose ?o ?p) (HandEmpty ?a) (AtConf ?q)
   :effect         (and (AtGrasp ?a ?o ?g ) (CanMove)
                   (not (AtPose ?o ?p)) (not (HandEmpty ?a))
                   (increase (total-cost) (PickCost)))
```

Fig. 2. PDDL representation for action *pick*.

classical planning, there are many further evolutions of it such as PDDL2.1 [56], PDDL3.0 [66], and PDDL+ [55]. Alternatively, **Simple Action Specification (SAS)** [8] is a simplified action structure for describing the planning problems with finite domain representations [130, 199], it is expressively equivalent to STRIPS without action parameters, while the main difference is that it supports variables with both propositional domains and discrete domains. The **Hierarchical Task Network (HTN)** is a refined structure comprised of partially or totally ordered actions constrained by state variables, it has been proved that HTN planning makes better expressive performance than classical planning [48].

TAMP approaches are formally different according to the implemented symbolic task planners. In total, we have detailed 87 papers of which 61 are explicitly implementing a specification language for task planners, these are presented in Table 1. TAMP combines discrete classical planning with continuous geometric satisfying to find a sequence of actions to reach the final states. Given an initial set of states $\mathcal{S}_0$ and a goal set of states $\mathcal{S}_*$, classical task planning is to find a discrete sequence of actions π in all model-defined transitions $\mathcal{T} \subseteq \mathcal{S} \times \mathcal{S}$ from $\mathcal{S}_0$ to $\mathcal{S}_*$. $\mathcal{S}$ represents all possible states in *planning domains* and each action corresponds to a transition defined in the domain. However, given an initial configuration q_0 and a target configuration q_*, the motion planning problem for a robot with d degrees of freedom is to find a continuous feasible path τ in the robot configuration space $Q \subset \mathbb{R}^d$, such as a collision-free path from start to end [83]. A PDDL action a of π for $\mathcal{S}$ is a tuple $a =< head(a), pre(a), eff(a), cost(a) >$, each element of which is:

- `head(a)`: Containing the action name and a list of related parameters $a(z_1, z_2, z_3, \ldots, z_k)$, the parameters must cooperate all of the variables in $pre(a)$ and $eff(a)$.
- `pre(a)`: A set of *literas* $\{p_1, p_2, \ldots, p_m\}$, which represent the *facts* must hold before performing action a, including *positive literal* and *negative literal.*
- `eff(a)`: A set of *literas* $\{e_1, e_2, \ldots, e_m\}$, which represent the *facts* must hold after performing action a, including *positive literal* and *negative literal.*
- `cost(a)`: A positive number denoting the cost of action a, sometimes is omitted with a default value of 1.

Figure 2 presents the PDDL representation for action `pick`, the robot in configuration q will pick up the object at pose p once there are a safe trajectory plan and available arm, the related world states will change according to the effect.

TAMP is essentially a strategy for determining how to handle hybrid constraint satisfaction problems, it couples action sequencing with continuous parameter searching. Shakey the Robot [136] performs the first TAMP algorithm by first finding high-level abstract action sequences and then searching for continuous values for motion planning. The strategy of semantic attachments [44, 45] searches for continuous motion parameters during task planning. Alternatively, some methods [39, 162] perform continuous parameter searching after and in alternation with

Table 1. TAMP Approaches in Sections 3 and 4

Reference	Task formulation/Tools Employed	Applications/Experiments
[92, 93]	HPN with HTN formulations	restaurant manipulations
[164]	HTN formulations	Barrett WAM robots, dual-arm manipulations
[163]	CHIMP with HTN formulations	PR2 restaurant manipulations
[97]	Hierarchical planning system with HTN formulations	online re-planning, dual-arm manipulations
[3]	AC decomposed by symbolic tasks	LFD, manipulations under external perturbation
[109]	CTAMP with CSP solver	Robot manipulations
[38, 39]	IDTMP with STRIPS formulations and SMT solver	Rethink Robotics Baxter manipulations
[129]	Constrained Graph representations and CSP solver	Rethink Robotics Baxter manipulations
[107]	ASP task planner	Robot manipulations and culprit detection problems
[162]	PDDL formulations	PR2 manipulations
[60]	EAS extended by SAS+	PR2 manipulations
[61, 62, 63]	PDDL formulations and CSP solver	PR2 restaurant manipulations
[150]	PDDL formulations and MCTS search algorithm	PR2 restaurant manipulations
[169]	PDDL formulations	PR2 manipulations in Gazebo
[191]	heuristic TSP solver as a weighted graph	pick-and-place domain experiments
[155]	MTL formulations and MILP solver	Baxter manipulations in MATLAB
[26]	LTL formulations and MILP solver	Robot manipulations
[103]	HIOA [120] formulations and MILP solver	Mars Transportation, Air Refueling and Truck-and-Drone Delivery domains
[74]	LTL formulations and MDP planner	iRobot navigation
[146]	POLGP with Decision tree formulations	Robot manipulations under uncertainties
[110]	M-VFH formulations and A^* search algorithm	Robot manipulations in clutter with Moveit!
[4]	PDDL formulations and Fast Forward planner	Robot manipulations
[132]	MPLP with graph-based formulations and parallel algorithm	PR2 manipulations in Gazebo
[87]	VKC with PDDL formulations	Robot manipulations
[195, 196]	Ontology-based formulations of Action-specific Knowledge	Manipulation trajectory control
[46]	PDDL2.1 formulations	Robot navigation
[149]	RobMAP with PDDL2.1 formulations	Robot navigation
[116]	PELON with ASP and PDDL formulations, FD solver	Robot navigation
[168]	MPTP with PDDL2.1 formulations	Robot navigation under uncertainties
[42]	TMPUD with ASP task planner	Urban self-Driving under uncertainties
[77]	LGP formulations	Architectural robotic manipulations
[184]	HTN formulations	Robot manipulations with incomplete knowledge
[47]	HTN formulations and FSM executor	Dynamic robot manipulations
[152]	HTN and PDDL formulations, BTs executor	Dynamic robot manipulations
[126]	PDDL2.1 formulations and BTs executor, PlanSys2	Dynamic robot manipulations
[165]	TMM with MDP planner	Robot navigation under uncertainties
[157]	STAMP with MDP planner	Robot manipulations under uncertainties
[194]	Hierarchical POMDP planner	Robot manipulations under uncertainties
[90]	HTN formulations	Robot manipulations under uncertainties
[76]	IBSP with STRIPS formulations	PR2 manipulations under uncertainties
[80]	HTN formulations and A^* search algorithm	AUV navigation under uncertainties
[64]	PDDL formulations	PR2 manipulations under uncertainties
[122]	Directed acyclic contingent planning graphs	Wumpus, localize and rock sample domains
[1]	LESAMPLE with PDDL formulations	Robot manipulations in clutter under uncertainties

(Continued)

Table 1. Continued

Reference	Task formulation/Tools Employed	Applications/Experiments
[153]	BBTs executor	R1 robot manipulations under uncertainties
[187, 188, 189]	MDP planner, ASP formulations	Robot navigation and detection under uncertainties in Gazebo
[37]	TMKit with PDDL formulations	Robot manipulations under uncertainties
[70]	Hierarchical tree structures and MoveIt!	Robot manipulations under uncertainties
[177]	Heuristic search task planner	Test tubes manipulation and rearrangement
[128]	extended LGP with STRIPS formulations	Robot manipulations with Tower of Hanoi problem
[142]	PDDL2.1 formulations	KUKA LBR iiwa robots, Dual-arm manipulations under uncertainties
[176]	Extended SAS+ formulations and vector-field reactive layer	Robot navigation and manipulations under uncertainties
[53]	PLATINUm task planner with EPSL	Human-robot coordination in manipulations
[24]	RH-TAMP with PDDL formulations	Panda and UR5 manipulations under uncertainties

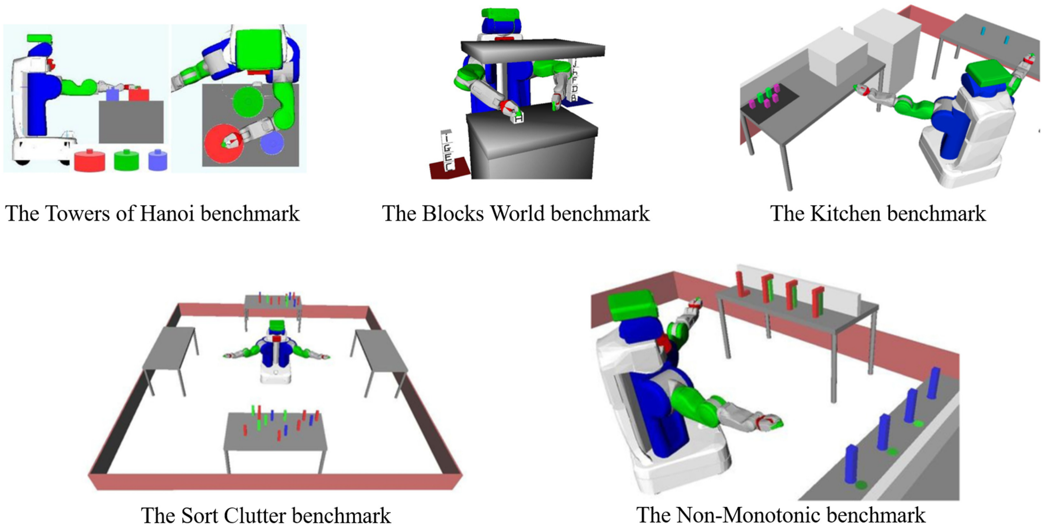
The Towers of Hanoi benchmark
The Blocks World benchmark
The Kitchen benchmark
The Sort Clutter benchmark
The Non-Monotonic benchmark

Fig. 3. PR2 benchmark problems for TAMP approaches in Reference [108].

task planning. A recent review from Garrett et al. [59] provides a comprehensive description of TAMP. It is based on foundations in classical task planning, motion planning and multi-modal motion planning [121]. They classify general TAMP methods according to the sequential strategies of solving hybrid constraint satisfaction problems. While either sequencing before satisfying or satisfying before sequencing or alternating between them is to obtain a detailed plan considering the low-level constraints. Additionally, Lagriffoul et al. [108] present the first platform-independent evaluation benchmark scenarios for TAMP, the properties of these benchmark scenarios provide a reference for many subsequent research experiments [24, 62, 63], Figure 3 shows some PR2 application scenarios. This article surveys the strategies of these regular TAMP approaches that generally assume the environment is static or perceptions are perfect. The strategy inclinations of these approaches reflect the origin of TAMP structure and how to solve the hybrid planning problem. In contrast to the previous work, we further survey online TAMP methods that can deal with uncertainties, these methods improve the internal structure of TAMP, making the planning system more robust. Additionally, we also present how learning algorithms contribute to TAMP in recent years.

A survey by Ingrand et al. [84] presents a global overview of deliberation functions for autonomous robots including planning, acting, monitoring, observing and learning. They show the perspective that deliberation in robotics does not limit to task planning but also acting and monitoring. Similarly, a book from Ghallab et al. [67] shows the significant need for deliberatively acting to decide how to perform each primitive action, they focus on the reasoning functions. Moreover, a book from Colledanchise et al. [34] introduces BTs for online behavior planning and acting. They show the historical development of BTs from the computer game industry to robotic applications. BTs synthesize the advantages of different decision-making structures, including Finite-state Machines, Subsumption Architecture, Teleo-Reactive programs, and Decision Trees. Jiménez et al. [88] review the techniques in machine learning for automated planning and organize them into two categories, including learning algorithms for action model planning and search controlling.

While our article is different from their studies, we verify their arguments from the perspective of TAMP. In other words, we limit the scope of this article to TAMP, a comprehensive planning approach considering both high-level task planning and low-level motion planning. We do not limit the approaches used in task planning or motion planning, because that TAMP is composed of them, there are large amounts of state-of-art for them separately. We do care about the studies combining them to solve real-world long-horizon tasks. Besides, whether the task planner is based on PDDL or STRIPS and other related planners is not important.

3 RECENT TRENDS IN REGULAR TASK AND MOTION PLANNING

TAMP is a more comprehensive planning method that treats low-level motion planning constraints as factors in high-level task planning than traditional task planners. Benefiting from these considerations, long-horizon tasks are more tractable without complex programming in motion planning. In this section, we survey the trends of relevant approaches reported in the literature for determining how to integrate task planning with motion planning (Section 3.1); how to make TAMP computational efficient (Section 3.2). Most of these TAMP methods are applied in robot manipulation, and a small number of them are used in robot navigation or other field scenarios (Section 3.3).

3.1 General Task and Motion Planning

Our work does not classify TAMP methods according to the performing strategy of task planning and motion planning [59], we classify them according to their preferred concentrations instead. The earlier TAMP methods focus on the hierarchical architecture of TAMP (Section 3.1.1), while some of these methods concentrate on the constraint relationship in TAMP (Section 3.1.2). Recent TAMP studies concentrate on getting a probability complete feasible solution in the constrained search space, such as the sampling-based approaches (Section 3.1.3), while other studies seek optimal plan trajectories according to objective functions (Section 3.1.4).

3.1.1 Hierarchical-based Task and Motion Planning. Hierarchical-based TAMP approaches decompose high-level abstract tasks into many sub-tasks, which can be easily solved, such as HTN. Given a global task, the hierarchical structure can alleviate the computational effort of combining task sequencing and feasibility check by solving smaller decomposed TAMP problems.

Kaelbling and Lozano-Pérez [92] propose a **hierarchical planning and execution strategy for task and motion planning in the now (HPN)**. They integrate the continuous geometric information with task planning to construct suitable choices for the operator's parameters by using geometric "suggesters." This method does not need a pre-discretization of the state space or action space. It continuously decomposes high-level abstract tasks into concrete primitive actions by nested procedures. The same authors extend their work to apply it to belief space [93].

Similarly, Suárez-Hernández et al. [164] propose a framework that integrates HTN task planner with geometric feasibility check. The HTN planner constructs hierarchical control structures with complex geometrical constraints of robotic arms. There are two types of task classification, *primitive* or *compound* task. The HTN planner iteratively decomposes a compound task into one or more sub-tasks until finding a feasible plan, these sub-tasks are stored in the planning stack.

Stock et al. [163] propose a **Conflict-driven Hierarchical Meta-CSP Planner (CHIMP)** with an expressive domain language to account for subtask dependencies, geometric or temporal constraints and resource usages. The hybrid search space is rigorously restricted by pre-defined decomposition rules. CHIMP uses a meta-CSP [124] reasoning method to search for a plan online. It spends a low planning time despite the huge hybrid search space. CHIMP does not need to plan from scratch when receiving a new goal, it merges tasks by unifying the former task steps, which can significantly reduce the planning and reaction time when facing some special situations.

Kast et al. [97] propose a hierarchical planning system to solve the task and motion problem with difficult geometric constraints and combinatorial complexity. The system combines expert knowledge with introspection capabilities to reach a given goal while solving the huge computational time problem due to high-dimensional state space. The hierarchical planner orchestrates the action sequences even for long-horizon tasks based on the previous work [96], it constructs a planning domain with the description of concepts and operators. In addition, the hierarchical planner is critical for efficient recovery and backtracking because of unforeseen circumstances during execution. Moreover, the approach can generate sequences of asynchronously parallel actions on real robots, by doing this, it can avoid unnecessary sequential execution and waiting times due to resource locks.

A strategy that hierarchically decomposes high-level abstract tasks into symbolic actions is proposed by Agostini et al. [3]. This method uses a novel representation of object-centered geometric constraints in the object configuration space to generate consistent solutions, it avoids unnecessary computation by the motion planner. Symbolic tasks are decomposed into rooted tree nodes that correspond to certain robot actions. The process from task planner to certain actions is articulated through a new structure called ***action context*** **(AC)**. AC consists of information about what action to do next and what action was executed before.

3.1.2 Constraint-based Task and Motion Planning. Constraint-based TAMP approaches regard the combination of task sequencing and low-level geometric constraints as **Constraint Satisfaction Problems (CSPs)**, they carefully process the constrained relationships by special solvers. Task planning and geometric constraints evaluation are interleaved in the TAMP framework of Reference [109] to generate symbolic action sequences. The planning space is explored during geometric backtracking. The method constructs a constraint network to address the significant computation of geometric backtracking in kinematically constrained problems. The set of constraints in the network is formulated as a set of linear inequalities and equalities to remove inconsistent actions. Dantam et al. [38] propose the IDTMP, which uses **Satisfiability Modulo Theories (SMT)** solvers incrementally and dynamically to make motion feasibility checks in task planning. The "constraint" corresponds to the logical assertion of the task planner based on satisfiability checking in this study. The method generates task plans efficiently by incorporating geometric constraint information from failed motion feasibility checks incrementally. The planning domains are defined with a customized task language and *scene graph*. With a time-limited bound increase, the constraint-based task planner is coupled with a sampling-based motion planner that ensures the probabilistically complete. IDTMP is validated on a physical Baxter manipulator and has better performance for task scalability and plan length than the previous similar framework by He et al. [78], The extended version [39] modifies their work with a formal analysis of the algorithm and enhances communication between the task and motion layer.

The constrained relationships can be constructed as a *Constraint Graph* by unifying task planning and motion planning under the rearrangement rules [129]. Rearrangement Planning [139] is a subclass of TAMP [59], including the topic of **Navigation Among Movable Obstacles (NAMO)** or **Manipulation Among Movable Obstacles (MAMO)**. A concrete algorithm called the *Manipulation RRT* is proposed based on *Constraint Graph* to solve N/MAMO problems. *Constraint Graph* gives a representation of facts to help *Manipulation RRT* understand the geometric relationships between objects. At the same time, *Manipulation RRT* provides more feedback to the task planner than a normal motion planner, which makes the method suitable for integrating with a task planner. *Constraint Graph* addresses task planning and motion planning problems simultaneously, it avoids complex feedback loops resulting from hierarchical planning.

Two culprit detection mechanisms are built to transmit geometric constraints to the task level in Reference [107]. The first mechanism generates relaxed geometric information by a set of approximated bounding boxes, which are represented by the network of linear constraints. The second mechanism takes input from the first mechanism to construct a *spatial constraints graph* of geometric dependencies between operators. The planning problem becomes a hybrid search problem when the failures are detected and fed back to the task layer. The shorter subsequence of actions can be generated and evaluated independently by the **Answer Set Programming (ASP)** [19] task planner. The task planner prunes the entire family of similar failure branches repeatedly to obtain a feasible solution.

3.1.3 Sampling-based Task and Motion Planning. Sampling-based TAMP approaches focus on finding feasible solutions in the constrained search space with probabilistic completeness. They are often sampling at the intersectional region of different constraints, which significantly reduces the hybrid search space. A novel representation of the geometric constraints is presented with the form of logical predicates in Reference [162], it makes the task-level actions to be identifiable and expressive. The author leverages off-the-shelf task planners and motion planners to bridge the gap between task planning and motion planning. The geometric representations of logical predicates such as "grasping pose for b_1" and "trajectory for reaching grasping pose for b_1," are translated from the geometric reasoning layer. The task planner searches for the feasible solution efficiently by parameterizing the discretization values sampled from the relevant constraints intersections.

Similarly, Garrett et al. [60] propose FFRob as an efficient and probabilistically complete algorithm for TAMP. The method introduces **Extended Action Specification (EAS)** as a new representation for symbolic planning. The conditions of actions are predicates incorporating geometric and kinematic constraints. The method pre-processes the discretization by sampling a finite number of actions and then represents the geometric constraints in a finite domain with EAS. The task planner applies efficient heuristic search algorithms from AI Planning Community to find feasible solutions. Their next work [61] gives a more comprehensive and theoretical explanation of sampling-based methods for TAMP. They model the TAMP problems as factored transition systems that convert system states to the intersection submanifold of several constraint manifolds where solutions lie. The method characterizes conditions on the submanifold and then uses a special solver to find feasible solutions under these conditions. Each condition corresponds to a conditional sampler that can be composed to generate suitable values on the submanifold. Additionally, the author presents two algorithms (*Incremental* and *Focus*) to search for feasible solutions. Then Garrett et al. [62] modify and upgrade their previous work, present STRIPStream and the final version PDDLStream [63] with an open-source TAMP framework. PDDLStream is a general-purpose framework for incorporating sampling generators in PDDL language, the conditional samplers are substituted with streams that are presented in a stream.pddl file. Two new algorithms (*Adaptive* and *Bindings*), which are kind of *Focus* algorithms that satisfy constraints lazily are proposed to solve a series of TAMP problems. The *Bindings* algorithm reconsiders each previously evaluated stream plan during the search, it is relatively computationally expensive. While the *Adaptive* algorithm balances the computational effort of searching and sampling processes to aggressively explore more possible bindings.

Ren et al. [150] propose eTAMP that integrates a top-k skeleton planner to generate different symbolic plan skeletons for long-horizon manipulation tasks. The method chooses the lowest cost skeleton candidate according to the current domain description. At the same time, the actions in the symbolic plan are refined to concrete values from optimistic objects according to the geometric constraints. Moreover, these concrete values are generated by streams [63], the pre-defined black-box conditional generators in the task planning domain. The method integrates the skeleton choosing process as an "additional stream" for sampling alternate skeletons in the discrete task space. Then a tree-structured search algorithm **Monte Carlo Tree Search (MCTS)** is used to solve the hybrid planning problem.

Thomason and Knepper [169] propose a novel TAMP method with a sampling-based motion planner to search for a symbolically and geometrically solution simultaneously in a single call. The key idea is to consider symbolic state into continuous parameter space, this is done by automatically deriving a function and leading a planner to regions of continuous space where symbolic action conditions are met. A factorization of the sampling problem is needed to decrease the dimensionality of the composite space. Besides, a continuous "semantics" is proposed for explicitly representing the regions hold by geometric predicates, such as a description of how far a given state is from the nearest state. Finally, a planner-agnostic sampling algorithm is presented to search for feasible plans.

3.1.4 Optimization-based Task and Motion Planning. Many TAMP methods only concentrate on satisfaction problems, while optimization-based TAMP methods search for feasible solutions minimizing the planning cost or time. They concentrate on finding optimal solutions while keeping manageable computational efficiency in TAMP. The specifications of operators are related to temporal logic and cost features [123].

Zhang and Shah [191] propose a hierarchical optimization-based TAMP method. The large-scale robotic manipulation applications are formulated as multi-level optimization problems, including task, action and motion planning levels. Given an optimization objective, the method finds an optimal performance metric sequence in the top task planning level by utilizing a heuristic TSP solver. The approach explicitly models the robot migration cost from one task to another. For example, if one object blocks another object, the transition cost will be infinite and the task-level algorithm will consider lower-level problems to optimize the cost.

Some methods optimize TAMP problems by using **Mixed Integer-Linear Programming (MILP)**. Saha and Julius [155] propose a TAMP framework that combines high-level MILP with low-level ***Metric Temporal Logic*** **(MTL)** formula for generating optimal solutions autonomously. MILP searches for the MTL task specifications about the sequence of actions by using a *knowledge map* of the workspace. The low-level MTL specifications extend ***Linear Temporal Logic*** **(LTL)** by appending time intervals to the temporal operators, the intervals are considered as constraints to optimize TAMP problem. The process iteratively calls an optimization algorithm in each step to modify the motion plan of the manipulator's arm until it chooses an optimal task sequence to meet the task-specified performance objective. However, if the chosen candidate sequence is not feasible, then the knowledge map will update the specifications to propose a new candidate. The optimization algorithm is based on gradient descent, it is proven to obtain a probabilistically complete solution even though it only guarantees locally optimal solutions. An optimization-based hybrid planning method uses MILP to inference on temporal concurrent goals [26]. MILP fixes the action number of automaton runs and plans over the mixed discrete-continuous spaces within a reasonable time to generate provably optimal plans. Kogo et al. [103] propose an improved model of optimization-based TAMP that reduces computational costs in robotic manipulation problems. The model integrated with collision avoidance is formulated as a MILP problem. The method solves the problem efficiently by two approaches leveraging the properties of collision check. The first presented approach is to reduce binary variables, which are related to collision check and substituted as continuous variables with additional hard constraints. The second approach proposes soft constraints as a new term added to the objective function, which restricts the motion paths softly in a delivery mission. Different from the hard constraints, soft constraints help the MILP solver more likely to find fractional solutions with shallower branching.

Soft constraints satisfaction can not only guarantee probabilistic satisfiability it can also optimize the mean cost in planning. Guo et al. [74] propose a hybrid plan algorithm for probabilistic motion planning, it combines high-level LTL task specifications with risk constraints. Given a control objective, the method uses the **Markov Decision Process (MDP)** to solve optimization planning problems. The goal is to synthesize a finite-memory control policy to generate robot motion plan trajectories. The policy is refined by high-level task sequence specifications with desired high probability and objective cost. The prefix and suffix parts of the system trajectories with two coupled linear programs are solved to obtain the optimal policies. In addition, this study considers cases where the probability of feasible solutions is zero, this makes the MDP without an **Accepting End Component (AEC)** [41, 58]. To avoid returning no solutions in this situation as most existing methods do, the method treats task level satisfactions as soft constraints and presents a relieved suffix plan to minimize the frequency of reaching a bad state.

The optimal plans are treated as trajectory trees in the study of Phiquepal and Toussaint [146]. They extend the previous work **Logic-geometric Programming (LGP)** [172] and propose a novel optimization-based TAMP solver. The solver works in two stages: first, it obtains a symbolic policy by approximate path costs from a piece-wise trajectory optimization, and then optimizes the joint trajectory tree corresponding to the policy. The whole trajectory in the symbolic decision tree is optimized all at once instead of optimizing the sequences separately. Some methods optimize TAMP manipulation problems in Clutter situations. Lee et al. [110] propose a tree search-based TAMP to grasp targets optimally in clutter considering both prehensile and non-prehensile manipulations. The method points out three key issues in this problem: (1) figuring out the minimum number of obstacles to be relocated, (2) manipulation planning of these clustered obstacles, (3) feasible motion planning to handle obstacles in issues (1) and (2). The first issue is optimally solved by the tree-search-based planning algorithm with **Modified Vector Field Histogram (M-VFH)**. M-VFH is a local path generation algorithm that figures how many obstacles block an object. The second issue is solved by taking extra consideration of non-prehensile manipulation during tree-search-based planning such as pushing obstacles by a collision. The third issue is a general problem of feasibility checks in the planning process.

3.2 Computational Optimization in Task and Motion Planning

TAMP is a hybrid planning approach, the computation of high-level task planning is less expensive than low-level motion planning because of the discrete search space. While motion planning needs to search frequently in the continuous parameter space such as searching for a collision-free path that will take a lot of computational resources. As a result, it is significant to improve the computational efficiency [111] of the deliberative planning system.

Sometimes a lazy search algorithm instead of an incremental one [61, 63] can make the hybrid planning process more efficient. Akbari et al. [4] propose a TAMP method that delays the geometric feasibility check until the corresponding action is selected by the heuristic. This reduces the computation cost in the motion planner, at the same time, the backtracking is reduced, because the task planner captures most of the geometric constraints, including grasp and place poses, inverse kinematics solutions and collision checks. Parallelization of computing [182, 183, 186, 190] is always a suitable choice to improve the computational bottleneck of search algorithms in TAMP. A **Massively Parallelized Lazy Search Algorithm (MPLP)** [132] leverages the modern multicore CPU to solve planning problems efficiently. The computational effort is mainly dependent on the evaluations of graph edges, which stand for the constraint satisfaction problems. MPLP implements massive parallelization by a **Multiple-instruction Multiple-data (MIMD)** execution model, which evaluates potential edges parallelly. Then the problem becomes determining what edges to evaluate, in what order and how to incorporate all these evaluations in the planning domains, the problem is solved by graph search algorithms.

Optimal compound actions may simplify the planning process instead of primitive actions. Jiao et al. [87] propose a **Virtual Kinematic Chain (VKC)**, which is a composition of primitive actions. They treat VKC as a new primitive action to improve the task planning efficiency of mobile manipulation problems. The method deliberatively consolidates the constraints of the mobile base, manipulation arm and the object to be manipulated. Then the VKC perspective can define *abstract actions* and avoid unnecessary predicates in the intermediate of symbolic operators.

Some methods use an ontology-based approach to reduce computational costs in motion planning. Zhao et al. [196] propose an ontology-based method that models a well-structured 3D environment as the task-oriented domain knowledge to improve path planning efficiency. The method generates a special path planning query for the primitive operator. The ontology method describes objects or regions involved in geometric constraints for the task, it presents a geometric model of the environment as well. As a result, the specific path planning will obtain better performance on computational time and path relevance. However, this work is still preliminary because of limited conceptions and rules, the validation and evaluation processes are also lacking as well. In the next step, they present their new work [195], focusing on simulating and validating processes with strong geometric constraints in robotic manipulation. The task-related information is a 3D **Environment Ontology (ENVOn)**, which introduces any geometric information of a given rigid body or a given place. The new method is validated through two different cases with incremental geometric constraints added to the environment. They also point out that few works have considered task-related information at the motion planning level, while it is the contribution they have done [10, 59], this is also similar to the state-of-the-art autonomous exploration [21, 198] in robotics, they are using global task information to guide local exploration.

3.3 Task and Motion Planning in Navigation and Other Applications

Over the past few years, TAMP has been mainly applied in long-horizon pick and place tasks among the research community. While TAMP approaches for navigation or other field applications have not yet been paid much attention to and therefore lack sufficient literature [116, 168], they present different challenges compared with TAMP approaches for manipulation.

In recent years, researchers have drawn attention to applying TAMP in navigation and autonomous urban driving. Edelkamp et al. [46] integrate temporal task planning with sampling-based motion planning to solve multi-goal motion planning problems under dynamic environments and time limitations. The roadmap is obtained as the symbolic task-level specification that informs the connectivity of the navigation space and helps the temporal task planner find a solution at each iteration. Then the sampling-based motion planner expands the roadmap based on the discrete solution. Similarly, a hybrid motion planning approach **Robotic Motion and Action Planning (RobMAP)** [149] is proposed to generate the optimal path with the prior task level sequence information. RobMAP leads to efficient navigation in a large-scale indoor environment. The method is based on ROSPlan [23], using PDDL planner and sampling-based RRT to find collision-free solutions.

Lo et al. [116] propose **Planning Efficiently for Task-level-optimal Navigation (PETLON)**, which is a TAMP approach for robot navigation within optimal cost. They combine ASP and PDDL formulations as the task planner to solve planning problems with FastDownward solver [79]. Based on their approach, Thomas et al. [168] propose a TAMP framework for navigation in large-scale environments. They show a need for motion-planning-aware task planners. They present a probabilistically complete method for navigation in unstructured human environments leveraging interactions between task and motion planning. The task planner generates an optimal path plan with a given roadmap conditioned on action cost. A reliable planning system is critical for the navigation of self-driving cars [9], Ding et al. [42] propose a **TAMP algorithm for Urban Driving (TMPUD)**, the first progress that uses TAMP to guide the safety level of urban driving actions. TMPUD presents a new safety estimator to judge actions' safety levels, it is implemented and evaluated with an autonomous driving simulation platform.

Building and construction applications are an extension of the basic manipulation problems, but with more rigorous requirements. Hartmann et al. [77] apply TAMP algorithm in architectural applications. In the building industry, integrating automated planning with suitable design analysis tools can make the design iteration cycles faster for designers and engineers. The method presents a multi-agent TAMP framework to solve hard long-horizon construction problems, such as constructing a full-scale building. The long-horizon planning process is decomposed as receding horizon planning problems in TAMP. This study uses LGP as the planner and combines it with a sampling-based method as well as an optimization-based method to form a robust TAMP-solver. The geometric, kinematic and static feasibility is iteratively evaluated during the whole construction process. Similarly, Huang et al. [81] integrate TAMP with architectural design. They bridge the gap between robotic construction and TAMP by presenting the plan skeleton formulations for realistic architectural case studies with TAMP solver.

3.4 Summary

TAMP is an essential research topic and progressing significantly in AI Planning Community for the last decade, it is more general and comprehensive than other symbolic task planners. The hierarchical framework combining task planner and motion planner is studied in the earlier research. Then TAMP develops into a hybrid planning system for long-horizon tasks and brings more constraints information from the motion layer to the task layer, CSP solvers are leveraged to find feasible solutions in the hybrid search space. But the search space is computationally intractable for planning with the incremental growth of constrained factors. Sampling-based TAMP methods transparently reduce the dimensionality of the searching parameter space and generate feasible solutions with probabilistic completeness, while optimization-based approaches look for solutions with minimal costs efficiently.

TAMP works as a high-level centralized information system, therefore implementing an efficient search algorithm is critical. It is an especially important research field in real-time planning and execution systems. Significant research efforts have been devoted to this field. In Table 1, we can figure out that most TAMP approaches are applied in robot manipulations. Alternatively, there is also a high potential in other planning domains where TAMP can be widely deployed such as robot navigation, exploration and architectural construction applications.

4 RECENT TRENDS IN ONLINE TASK AND MOTION PLANNING

The regular TAMP methods we have talked about are often associated with one or more assumptions, such as the full observability of the world or perfect perception, planning with static environments, a complete knowledge base and deterministic actions. However, it is hard to complete a long-horizon task in a real-world environment. There are a few of the introduced TAMP approaches have considered uncertainties [42, 46, 74, 93, 97, 146, 168]. The dynamic factor from other agent or the object, the unexpected failure with action, the inaccurate sensory information, the neglected or unobserved relevant part and the incomplete knowledge base, all of these will develop into uncertainties [18, 94, 138, 185] that fail in the execution of the plan with a high possibility. In this section, we review the research trends of online TAMP. These studies further optimize regular TAMP into an online planning system to deal with most uncertainties in real applications. Online TAMP considers re-planning according to unpredictable state changes. The internal robustness of TAMP has been improved by different planning tools, strategies and mechanisms in this section. We introduce the robust and reactive behavior planning approaches that are related to the execution of actions in Section 4.1; Section 4.2 discusses the trends of probability distributed TAMP approaches; Section 4.3 discusses trends of control-based TAMP approaches, which formulate the planning and acting processes as a control loop.

4.1 Behavior Planning

If the task planner has found a sequence of actions, then the next step is to implement these actions into relevant executors. Thanks to the developments in motion planning, we do not have to deal with the specific parameters in motion planner, such as velocities and accelerations [137]. Since that, the simplest way to implement actions is to directly send commands (goal configurations in path planning) to the motion planner. However, this manner is sensitive to disturbances in the environment even tiny tolerances or changes of the constrained parameters will cause a failure of the action. Therefore a special action executor and replanning mechanism of actions are needed for the planning system considering this situation.

4.1.1 Hierarchical Task Network. HTN is designed for solving complex tasks. It can not only work as a task planner to search for solutions with domain knowledge; but can also manage the execution of each operator [98]. HTN is constructed by two types of tasks, called compound task and primitive task. Given a task description, HTN iteratively decomposes the task into primitive tasks to find solutions until there is no compound task. Each primitive task is encoded into the standard operator including precondition and effect specified by a set of literals. If the precondition is not satisfiable during execution, then HTN can replan at that point instead of replanning from the root task with a changed domain. Furthermore, a forward-search procedure is used for most existing HTN planners [133, 134]. As a result, HTN can speed up planning with partial plans while does not have to complete the entire search before acting.

Weser et al. [184] uses HTN to handle complex robotic scenarios without the ***Closed-world Assumption (CWA)*** [167]. They integrate HTN with a robot control module to act and monitor primitive actions. The method generates a plan skeleton even with an incomplete domain representation (incorrect assumptions and unknown objects) while maintaining the properties of the HTN planner. It has a deliberative replanning procedure that corporates perceptual actions during execution to deal with unknown objects. Einig et al. [47] propose a State Machine-based parallel execution layer to perform actions generated from a high-level HTN planner. The method parallelizes the original sequence of actions by a *defensive or offensive* approach. Then it manages these optimal actions by a **State Machine Architecture (SMACH)** interpreter, which transposes the atomic actions to sequence and parallel containers. The containers consist of states corresponding to the atomic actions to be executed. Furthermore, the framework formulates as a closed-loop structure through a cost-function-based replanning algorithm to increase the robustness of the system.

4.1.2 Behavior Trees. BTs are also a kind of behavior planning structure for reactive and fault-tolerant task executions. BTs are either manually made by human experts or automatically generated by symbolic planners to solve complex tasks. BTs generalize advantages of many reactive behavior planning structures such as the **Finite-state Machines (FSM)**, Teleo-reactive Paradigm and Decision Trees [34]. There are four control flow nodes (Fallback, Sequence, Parallel, and Decorator) and two execution nodes (Action and Condition) in BTs. Control flow nodes manage how the execution nodes operate, while the Action nodes are directly related to concrete operators constrained by the Condition nodes. BTs are well known for their modularity, reusability and reactivity in challenging behavior planning applications. BTs are available for many symbolic task planners that find a feasible plan and implement it into BTs [33].

Automatic generation of BTs is more practical than manual creation. Rovida et al. [152] combine execution behaviors with the planning domain to generate an **extended Behavior Tree (eBT)**. They use HTN planner and PDDL-based domain specifications to model planning problems. Given a goal state, the planning algorithm FastDownward [79] is used to search for the sequence of actions that are then initialized to eBT. The modified HTN planner expands eBT by substituting the abstract nodes with primitive actions. Additionally, the eBT specifies a total organization of nodes according to the context of the planner to optimize the execution time as well as the use of resources. Similarly, an autonomous and optimal BTs generating method with a PDDL task planner is presented in Reference [126]. They use a temporal task planner based on PDDL2.1 to model the planning problems and generate the sequence of actions to complete long-horizon tasks. These actions are converted to a **Directed Acyclic Graph (DAG)**, of which the edges represent the matched precondition and effect of actions according to the PDDL specifications. Finally, the execution of the planned solution is managed by control flow nodes of BTs, it is a compact, robust, and reactive way to execute the plan.

4.2 Task and Motion Planning in Belief Space

General TAMP methods usually solve deterministic planning problems in a fully observable environment with zero entropy. However, there have been large amounts of studies about planning in belief space within the AI Planning Community [12, 20]. Planning problems with high entropy are often solved by **Partially Observable Markov Decision Process (POMDP)** methods [75, 106], which deal with policy evaluation and state estimation in belief space. While it is in trouble with the curse of dimensionality and long duration. Figure 4 presents the relationships between computational effort and entropy of TAMP approaches. General TAMP approaches are more tractable due to the neglection of uncertainties, while directly planning in the belief space with high entropy is computationally expensive. Benefiting from recent developments in robot sensor systems, including efficient algorithms and economical hardware, there is a research trend of deliberatively planning with low entropy, which balances the advantage of computational efficiency in zero-entropy planning with intractable policy optimization in high-entropy planning [1]. We mainly survey the relevant low-entropy TAMP approaches in this section.

4.2.1 MDP-based Methods. MDP is a mathematical model of sequential decision-making, which generates stochastic policy and achievable reward to an agent. MDPs have been extended to robotic planning under uncertainties for a long history [147, 170] by searching for optimal policies with objective functions. However, we focus on the approaches keeping the basic structure of TAMP to complete the goal with a nondeterministic model instead of the approaches optimizing the sum of rewards over a fixed long-horizon or the infinite horizon.

Uncertainty information can be transferred from motion planning to task planning by the **Task Motion Multigraph (TMM)** proposed in a **Simultaneous Task and Motion Planning**

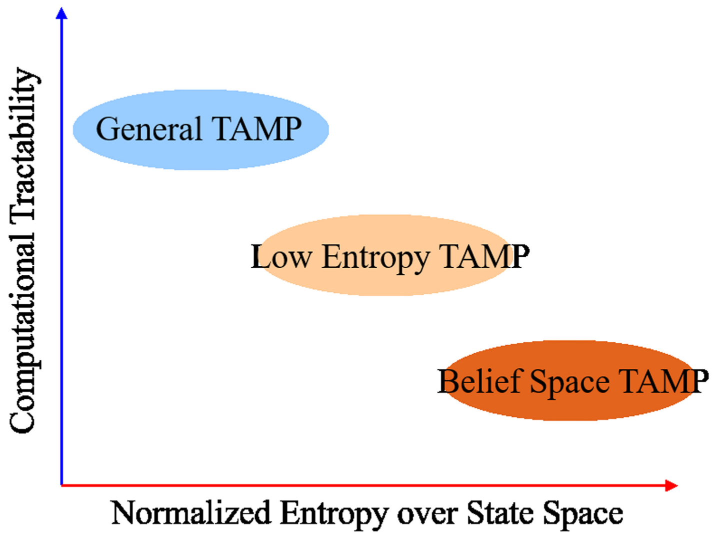
Computational Tractability
General TAMP
Low Entropy TAMP
Belief Space TAMP
Normalized Entropy over State Space

Fig. 4. An illustration comparing the relative computational tractability of TAMP approaches and the level of entropy in belief space [1].

(STAMP) framework [165]. The MDP planner is embedded with TMM, it solves directed acyclic graphs problem at the symbolic task level and chooses the higher probabilistic feasible solution. Given an MDP model, the value function is iteratively called by the planning system to find the optimal policy and generate a sequence of states. Each state extracted from the MDP corresponds to the node in TMM. Moreover, specifications for uncertainties can be reflected on the edges of the TMM. Similarly, Shah et al. [157] propose a probabilistically complete algorithm for *anytime* computing task and motion policies. The task-level specifications are formulated as **Stochastic Shortest Path (SSP)** planner problem, which is a subclass of MDPs [13]. The motion planner refines each action of the abstract action policy by using the ***Plan Refinement Graph*** **(PRG)**. The edges in PRG stand for policy-based transitions between states. The method iteratively extracts a path and refines it from PRG until the desired solution for most-likely outcomes is found during execution.

Zhao and Chen [194] propose a new hierarchical POMDP approach that models the manipulation planning problem in clutter. Both the high-level task planning and low-level motion planning are formulated by POMDP and utilized to search for feasible solutions. An original belief tree is generated from the POMDP formulations, the motion planner checks the feasibilities of actions and expands the original tree. Furthermore, the expanded belief tree is generated by abstract POMDP procedures, which incorporate state and action definitions as well as the control policy. Finally, the robot observes and updates its belief represented by a set of particles after each action.

4.2.2 *Approximated Model-based Methods.*

Some methods approximate the POMDP planning model as a deterministic model, then classical task planners can be used to solve the planning problems efficiently with a determinization process and even search in the large-scale space.

The uncertainties of both the outcomes of actions and the current state are considered in a hierarchical TAMP framework [90]. The method searches with an approximate determinization of the task domain according to the uncertainties of operators. It replans when the outcomes monitored during execution are distinct from the predictions. The uncertainties of the current state are characterized as symbolic fluents. Symbolic fluents are also used to describe the world states, goals and regression conditions. Finally, a regression-based planning algorithm is used to generate the plan, which is then executed with continuously monitoring. Hadfield-Menell et al. [76] propose an **Interfaced Belief Space Planning (IBSP)** TAMP framework to solve long-horizon tasks under uncertainty. They extend the domain specifications from Srivastava et al. [162] to use the ***Maximum Likelihood Observation*** **(MLO)** determinization as an approximate method for POMDPs. The abstract domain specifications model system dynamics for MLO determinization. Then IBSP searches for a plan skeleton, refines it and determines its success or failure. Moreover, the operators are divided into *actions* and *observations*, they are defined in the belief space with the probability distribution.

POMDP is a general model with a state estimator for planning under uncertainties. A high-level planning method incorporating the symbolic representation of the belief state is proposed to solve POMDP problems by Hou et al. [80]. K-means algorithm is used in their work to divide partitions of the belief space with a fixed number of symbolic representations. The method avoids exponential computation growth as the increase of dimensionality. It uses a classical task planner (HTN and A^*) to solve the planning problems with the discretization strategy. Then it constructs a deterministic graph in the POMDP domain, which is a symbolic modeled transition system that consists of abstracted probabilistic distribution nodes corresponding to a set of belief states.

4.2.3 Belief Representation-based Methods. Symbolic task planners can still leverage geometric representations from motion planners even in a partially observable environment. Some researchers directly search plans in the hybrid belief space with the representations of probabilistic distributed actions.

Online planners combine actuation action with sensing action to estimate the state after execution. An online TAMP method [64] performs in the hybrid belief space by modeling deterministic observation, visibility checking and Bayesian belief filtering with weighted particles. The stochastic planning problem is formulated as a hybrid, belief space ***Stochastic Shortest-path Problem*** **(SSPP)** [13]. Given a start prior belief state, the objective is to find a goal set of belief states for the planning system. The method extends the previous PDDLStream method [63] and constructs perceptual action with the outcomes determined by probability distributions. The relative stream of the perception-action is operating on distribution with the encoded Bayesian filtering process. In addition, this study presents a replanning algorithm that plans reference to the old plan skeleton, as a result, it can make the online planning progress move towards the target avoiding duplicate searching. Similarly, Maliah et al. [122] propose an online contingent planner CPOR to avoid repeated computation of sub-tasks caused by uncertainties. CPOR is planning throughout the execution process instead of searching for a complete plan offline. It iteratively calls to the online planner for the current belief state and then represents the planning tree as a modified and pruned directed acyclic graph, where edges represent sensing actions and nodes stand for actuation actions.

Adu-Bredu et al. [1] propose a goal-directed planning method **Low-entropy Sampling planner (LESAMPLE)** leveraging the advantages in classical planning and belief space representation to solve low-entropy problems. LESAMPLE uses a set of weighted hypotheses to model each state, which is then sent to a weighted sampler to get a reliable estimation. LESAMPLE simultaneously updates the observed belief states after each planned action has been executed. If there are differences between the updated belief states and the predicted outcomes, then LESAMPLE will reweight, resample, and replan for new feasible solutions.

4.2.4 BTs-based Methods. BTs can also work in belief space with their control flow nodes and probabilistic actions. ***Belief Behavior Trees*** **(BBTs)** add an unknown status of the condition node that extends general BTs to handle partially-observable applications [153]. The task planner plans

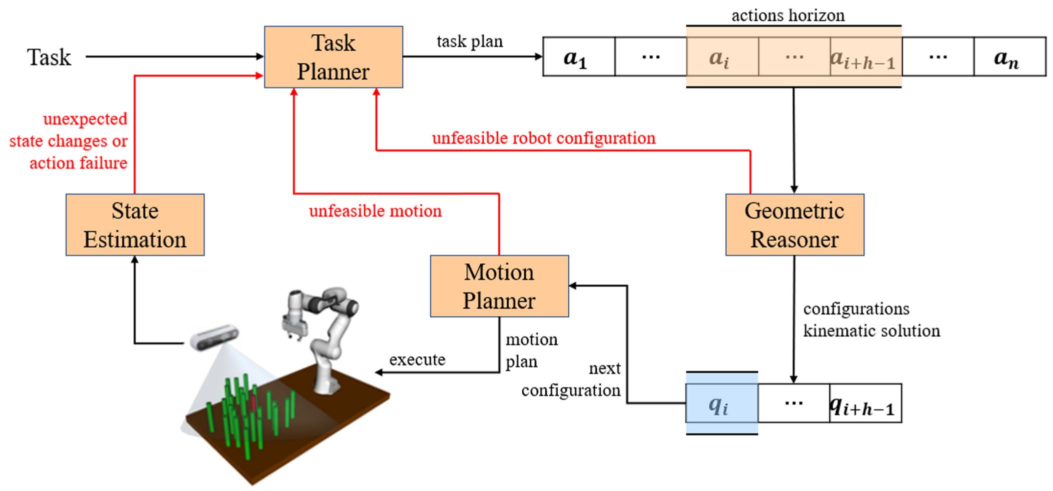

Fig. 5. A classic control-based TAMP framework from Reference [24].

directly in the belief space for constructing BBTs. The belief state is defined as the probabilistic distribution of physical states. Action nodes are divided into actuation actions and perception actions, both of which have probabilistic outcomes. The study uses Skipper node [154] as a new control BTs node to handle current state uncertainties, additionally, it specifies actions with precondition and postcondition to handle future state uncertainties.

Yang et al. [187] propose an **Adjoint Action Model (AAM)** that describes the continuous relationship between actuation actions and observation actions. AAM integrates with the POMDP planner to construct a plan-executing model with intermediate invariant conditions, which enables run-time observing during the execution of actuation actions. The same author [188] proposes a novel **Adjoint Sensing and Acting (ASA)** framework that generates a high-level plan with two scheme modes and constructs an extended BT for executing the plan. The scheme mode is either a sequential adjoint interaction with the robot observing and actuating or a parallel one. The planner is integrated with the extended BT to deliberatively plan and reactively execute tasks in partially observable domains. Furthermore, their recent study [189] integrates the POPF [32] planner and the Answer Set Planner Clingo [65] with motion controllers to implement their adjoint observation model during the planning and acting.

4.3 Control-based Task and Motion Planning

Planning in belief space can solve many non-deterministic problems. It is important to monitor the execution of plan steps in a deliberative planning system. The methods that combine planning with acting and formulate an online planning control loop are practical for challenging real-world applications. Control-based TAMP approaches typically separate the planning system into two layers, the TAMP and the acting layer of P-ALS as mentioned in Section 1. The planning layer generates a feasible sequence of actions with detailed guidance to the acting layer, then the acting layer performs these ordered actions by the implemented executors. Efficient and real-time interactions between the planning and acting layers have a major influence on the practical utilization values of the TAMP framework. Figure 5 presents a classic structure of the control-based TAMP approach [24] that combines the planning layer with the acting layer. State estimation is carried out after the performing of each ordered action; if an unexpected event is monitored during execution, then TAMP will react quickly and generate a modified plan based on the former one.

An open-source **TAMP framework TM Kit (TMKit)** from Dantam et al. [37] is an end-to-end system for probabilistically complete planning and instantaneous acting. TMKit can integrate different planning methods with multiple domains. The refined sequence of actions to achieve task goals are the immediate output of TMKit, it follows the execution process by interpolating values to the motion planner and performing the corresponding actions. A feedback control law is used to correct the positioning error during executions of the motion plan.

Sometimes a recovery strategy to deal with failures is important during acting. Görner et al. [70] propose a modular and flexible TAMP system for robotic manipulation. The high-level task is represented as a hierarchical tree structure that generates sequential as well as parallel primitive actions. This study points out that a key bottleneck of existing general TAMP approaches is that these methods may suffer from the modification of the world state even a minor parameter change may cause a failure during the execution. The method does not need to forward the planned solution trajectory to the low-level controller, it can replan during execution especially when failures are happening. An efficient scheduler is presented to first find out economical solutions as early as possible. Then continuing planning is implemented during the execution of the early partial plan. Failures can be readily evaluated by perceptions and separated from the planning stage. Wan et al. [177] propose a hybrid control-based planning system that is implemented in the Medical Laboratory application scenario where large amounts of test tubes are needed to be pre-processed to prevent COVID-19. The system autonomously recognizes, deliberatively plans and manipulates to separate and arrange these tubes. Moreover, this study proposes an error identification and recovery strategy by visual detection. The hybrid planning system works recursively until reaching the goal, and it ensures the completeness of the planner.

We have shown the capability of a reactive action controller to perform in a dynamic environment such as BTs. A reactive operational space controller [99] is used to consummate the TAMP framework presented in Reference [128]. The method is robust with the dynamic environment where even if the objects are moving, the planner can be processed by a reactive controller in real-time. The high-level symbolic specifications are described by STRIPS. If a candidate solution is presented, then the planner calculates the optimal trajectory for the solution according to an objective function. Furthermore, this work extends LGP to handle disturbances in the execution environment with the new Cartesian frame formulation. It simplifies the manipulation task as controlling a point relative to some reference frame. Pane et al. [142] combines the conventional TAMP approach with acting as a comprehensive planning system. The main contribution of the study is the composition of reactive actions in the acting control layer, such as adding the collision avoidance ability to a primitive action without interrupting the executing task. The PDDL2.1-based task planner generates a nominal feasible plan, which is then refined into constraint-based actions. Then a FSM scheduler embeds each action into an **eTaSL Controller (eTC)**. The eTC can not only execute each action in the plan reactively but can also receive feedback simultaneously by monitoring through the sensory modules of the platform layer. The planner will replan for a new reactive composing solution when a disturbance is detected, which leaves the old plan infeasible.

Vasilopoulos et al. [176] integrate a deliberative planning layer with a low-level vector-field-based reactive motion layer to plan in highly dynamic environments. The deliberative layer uses the ***Continuous Constraint Contract*** **(C3)**, which is an extended version of the SAS+ formalism to describe the states. Each state is a set of variable-value pairs that represent the constrained configurations. The planner generates a sequence of actions and their configurations. Then the corresponding C3 contracts are checked by the vector-field-based motion layer. Instead of refining a single motion plan, the motion layer constructs a control policy that either completes the objective from the deliberative layer or returns a failure specifying the incorrectness. Given a target location, the vector field of the motion planner eventually leads the robot to it while avoiding the known or unforeseen obstacles in the environment.

Real-time monitoring of the execution is important for most control-based TAMP approaches, especially in the study [53] for **Human-Robot Collaboration (HRC)** tasks. The plan that consists of human-robot coordination, symbolic action sequence and assignment is obtained from the task planner. Online planning with feedback from continuously monitoring is needed, since the duration of human tasks is not sure. The framework is capable of handling the dynamic problem by pursuing a Sense-Plan-Act loop. The task planner is implemented by PLATINUm [174] (a temporal planning framework), it plans or replans for a feasible solution.

Castaman et al. [24] propose ***Receding Horizon Task and Motion Planning*** **(RH-TAMP)**, which is an online planning algorithm over a receding horizon. RH-TAMP can handle disturbances in the environment, unlike most existing general TAMP algorithms, which have to assume the static environment and ideal perception. RH-TAMP does not limit to the once plan algorithms, it defines an *actions horizon* as a partial sequence of the full plan computed once by the task planner. Each action in the *actions horizon* is refined iteratively by the geometric reasoner. The planning process will continue when the first action of the horizon is feasible to execute, it moves forward the actions horizon window during execution. At the same time, a state estimation module implemented with a sensory device will evaluate the actual states with the predicted output; if a configuration difference happened, which means a failure or environment modification, then the task planner will replan for a new solution.

4.4 Summary

The implementation of regular TAMP in real applications is not always going well because of uncertainties caused by the dynamic environment, partial observability or incomplete knowledge. Studies in this section improve and expand the basic TAMP structure, formulating a closed-loop online planning system to deal with uncertainties. We first introduce the behavior planning methods in the related research field, which present control structures for action execution instead of sending commands to the motion planner directly. Tolerable dynamics or disturbances can be self-handled by the control flows of these methods without replanning in the task layer. Then, we show the low-entropy TAMP approaches in the belief state space. MDP-based approaches offer a classical framework for planning with probabilistic effects and compute an optimal action policy that maps states into applicable actions. However, approximated model-based approaches simplify the planning processes in MDPs by symbolic task planning with deterministic transitions, they are more tractable in long-horizon tasks than MDP-based methods. In addition, some approaches make representations for probabilistic distributed operators (sensory actions and actuation actions), they search for solutions directly in the belief space. Finally, we present the recent trends of control-based TAMP approaches, these approaches formulate the online planning-acting framework to address the planning problem with environmental variability. It is a comprehensive and robust framework for robot planning systems, as we presented in Section 1. More investigations in this research field are needed for improving the capabilities of the promising framework.

5 RECENT TRENDS IN LEARNING ALGORITHMS FOR TASK AND MOTION PLANNING

Given a goal, TAMP is aim to search for a sequence of actions to finish the task by specifying a model with comprehensible representations. Then the question is whether we can leverage the rapid development of machine learning to build the model in TAMP. TAMP is a deliberative decision-making process that can also be solved by a famous pure-learning approach, reinforcement learning [91, 166]. It learns a policy in MDPs to decide what step to execute according to current and future states with a discount factor. However, it is limited in the capability of generalization, since it is complicated to generalize in different long-horizon manipulation problems. Recent trends of learning algorithms in TAMP either learn guidance for planning (Section 5.1) or learn for an intelligent low-level motion planner related to the task sequences (Section 5.2); furthermore, generalizing TAMP to different applications will also face uncertainties, some of these learning algorithms can handle uncertainties in real-world applications (Section 5.3).

5.1 Learning Guidance for Symbolic Planning

Most of the general TAMP methods we presented before do not have the capability of learning from similar planning procedures or past experiences, while the capability can reduce the computational effort significantly. However, pure-learning methods are often data-inefficient in long-horizon problems such as AlphaGo Zero [160] with a value function and learned policy from past experiences. A promising approach to balance general TAMP methods with pure-learning methods is to find suitable representations for plan trajectories to guide deliberative planning.

Kim et al. [101] propose a learning method that uses score-space to evaluate the quality of a set of *solution constraints* corresponding to a problem instance. The method learns to predict solution constraints by finding a subset of decision parameters on the search space. The score-space representation directly describes the similarity between problem instance and *solution constraints*. Furthermore, the method transfers knowledge from past experiences to current instances by referring to the prediction of score-space and correlation of former solution constraints. Their recent work [100] proposes a hierarchical learning algorithm called ***Sampling-based Abstract-edge Heuristic Search* (SAHS)** to guide TAMP in the NAMO problems. They present a learned rank function to guide the symbolic search in high-level task planning; moreover, a learned sampler is also presented for finding feasible motion-level solutions efficiently. An objective function is designed to follow these actions that have not been chosen in the states; as a result, the method considers most of the state-action pairs and has a high probability to achieve a goal successfully. Besides, *geometric predicates* are used as representations for a graph, which is an input for **Graph Neural Networks (GNNs)**. Then the abstract action ranking is computed by GNNs. At the same time, the continuous parameter sampler is separately learned by a general adversarial network WGAN-GP [72] with the input of the key-configuration-based representations.

The transition relationships between states can formulate as a task graph and learn from past experiences or demonstrations. A hierarchical end-to-end learning algorithm is designed for 6-DoF grasping in clutter [178], it is based on partially-observed point cloud data. In this study, a variational autoencoder is learned from expert demonstrations to generate latent plans. The plans are then encoded in an embedded space scored by a critic network to sample various motion trajectories during execution. The critic network is used for plan selection at the high level, which is trained by Q-learning [31, 197]; it integrates with a control policy classifier, which is trained for switching reactive policies to handle different instances. Similarly, a **Geometric Task Network (GTN)** embedded with the hierarchical and compositional planning framework [73] is presented for offline learning and online execution in various scenarios. GTN is a task graph learned from demonstrations without manual parameterized action representations. It details the transitions among operators and the associated geometric constraints. Zhu et al. [200] propose a vision-based and goal-conditioned hierarchical planning framework that integrates neuro-symbolic task planning with motion planning for long-horizon manipulation applications. They use task-level *symbolic scene graph* and motion-level *geometric scene graph* to represent the manipulation environment with object-centered information. The *symbolic scene graph* models the abstract semantic relationships between the robot and objects, while the *geometric scene graph* represents 6-DoF poses and geometric constraints of the objects. The hierarchical graphs use GNNs to learn features between nodes and edges. The task-level model is trained with demonstrations including state transitions, while the motion model is trained with task-agnostic primitive actions that are successfully executed.

Imitation learning is gaining popularity in AI Planning, since demonstrating the desired action is easier than manually programming it [52, 82]. Primitive operators learned from demonstrations and specified with domain representations can generalize to similar scenarios without heavy re-programming efforts. The **Learning Operators for TAMP (LOFT)** algorithm [161] treats the operators as non-deterministic transitions, since they can learn from the unavoidable lossiness. Given a dataset of low-level transitions, then these symbolic operators with effects and matching preconditions are learned from a bottom-up relational algorithm. The algorithm decouples the effect clustering, precondition learning and parameter estimation procedures. Diehl et al. [40] propose a method to generate planning domain representations from human demonstrations automatically. They specify segmentation and recognition of different classified action transitions from human demonstration. The relevant precondition and effect are extracted from a generated novel operator. Finally, given a user-defined goal, the learned operators with an optimization criterion are used by a symbolic planner to search for feasible solutions.

Li et al. [112] propose a hierarchical learned framework including a low-level controller and a discrete high-level latent action space planner. They use imitation learning to generate a set of expert primitive actions, which are then transformed from the discrete space into a continuous space while learning a set of latent actions. A model-based high-level planner searches for a solution in the learned latent action space to reach the desired goal. The low-level controller iteratively learns a dynamic model given a latent action input. Among each cycle of learning, the dynamic model is used for model-predictive controlling, it can plan in a dynamic environment reactively. Sometimes self-imitation learning algorithm is data-efficient in applications where massive and broadly distributed data are needed. The ***Self-imitation Learning by Planning*** **(SILP)** algorithm from Luo et al. [118] relabels the robot's successful actions as demonstrations for policy learning. A probabilistic roadmap is used as a planner to build the directed collision-free graph on the visited states, which are treated as nodes in the PRM. Then the planned path is specified as MDP formulations for reinforcement learning. SILP can generate massive training data for reinforcement learning with motion planning tasks while not causing extra computational effort during training.

5.2 Learning Guidance for Task-constrained Motion Planning

Another worth researching type of learning algorithms in TAMP is to guide task-related motion planning efficiently [100]. We survey the learning algorithms in task information constrained motion planning based on the significant progress in pure-motion planning and machine learning [193]. Since the low-level feasibility check of TAMP is expensive, learning algorithms can reduce computational efforts of sampling or trajectory optimization procedures effectively, they either learn from past experiences or expert demonstrations.

5.2.1 Learning Guidance for Sampling. Bowen et al. [16] employ a closed-loop, task model guided motion planner that samples collision-free paths by sensing obstacles efficiently. The task model is learned from a set of expert demonstrations and can be generalized to new environments for motion planning even with movable obstacles during execution. A TAMP algorithm with a learned classifier [181] balances the heavy cost of motion feasibility check with the impracticality of learning a classifier from scratch. The method uses the classifier on approximations and does not aim to generalize the classifier, but it can expand to new domains. Given a set of minimal standard scenes, the classifier is characterized by SVM. The information about world states, actions and obstacles is presented in the scene. The dataset is a set of labeled outputs from the sampling-based motion planner, which runs for a long time ensuring probabilistic completeness. Qureshi et al. [148] propose a neural planning framework called **Constrained Motion Planning Networks X (CoMPNetX)**. It consists of a conditional deep neural generator, discriminator, neural projection operator, and neural samplers. The generator and discriminator are conditioned on the representations of task and scene observations. CoMPNetX generates informed implicit manifold configurations and works with a motion planning algorithm to find feasible plans quickly. COMPNetX is suitable for speeding up many underlying sampling-based planning algorithms due to the informed but random sampling procedures as well as the parallelization capacity. **Hierarchical Abstraction guided Robot Planner (HARP)** in Reference [156] learns for a predictive model of critical regions and guides the high-level navigation and low-level sampling of the planning process. HARP computes hierarchical state and action transitions with the model. The training data formulate a structure called the **Region-based Voronoi Diagram (RBVD)** from different domain environments.

The low relevant information gathered now may influence future planning. The **Learning a Motion Policy (LAMP)** framework [173] gathers past paths in a navigation stack. LAMP learns hidden properties from the experiences, such as the obstacles observed before, this helps with reactive online planning in navigation applications. Like expert demonstrations, successful plan trajectories can be formulated as training data for the planner. Pairet et al. [140] propose two planners, **Experience-driven Random Trees (ERT)** and bi-directional version ERTConnect. They leverage prior experiences, which are decomposable and malleable to search in the task space. The planners can generate a single prior action sequence among obviously varied task instances. The planners iteratively construct a tree of segmented experience from a total path experience, the suitable branches come from semi-randomly morphing segments of the experience. In addition, the planners enable the ability to choose the best candidate given a library of path experiences. The library can be incrementally updated by adding newly generated action plans as well as learning experiences from human demonstrations. Similarly, an experience-based framework ALEF [102] is presented to solve multi-modal planning problems. ALEF constructs a sparse roadmap in ***Augmented Foliated Space*** **(AFS)**, which is a configuration state space including *manifold constraints* from different modes. AFS unifies and relates experience collected from each mode problem that belongs to the same family, it learns from the paths in the roadmap. The related transitions are used to bias sampling in a sampling-based planner when there is a new search, which expedites the search in similar problems.

5.2.2 Learning Guidance for Optimization. Loula et al. [117] propose a framework to learn constraint-based TAMP model. Given a demonstration of the task, the model decomposes it into different sets of constraints for a low-level trajectory optimization problem. Each set of constraints stands for a special mode, such as a contact between gripper and object. The observed demonstrations during training are assumed with the same sequence of modes until a switch is called. The model is trained by gradient descent to minimize the loss function, which corresponds to the parametrized function of mode constraints. Angelov et al. [5] propose a framework that automatically integrates different policies dealing with motion planning trajectories, dynamic action primitives and neural network controllers. They define a hierarchical controller according to the set of existing transitions from the current state to the future state. A Goal Score estimator is learned from expert demonstrations, it sequences the policies and leads the system to the task goal according to the selected controller and evaluations between future and current states.

5.3 Learning-based Task and Motion Planning under Uncertainties

We have reviewed the method of online TAMP to handle uncertainties in Section 4. Learning-based TAMP approaches can extract features from training data set to deal with uncertainties rather than optimize the internal structure of TAMP. The combinatorial explosion problem becomes more intractable when integrating learning algorithms in TAMP to handle uncertainties. Partial observability and incomplete knowledge of the constrained objects multiply the possible states that can be reached as well as the complexity of the search spaces. We take a review of recent learning approaches for TAMP under uncertainties and present how they leverage the learned knowledge to complete long-horizon tasks.

5.3.1 Learning Planning Policies in Partially Observable Environments. The learned high-level action policy and low-level action control policy in TAMP can be treated as part of a heuristic search algorithm to solve long-horizon tasks. However, there is a major challenge of TAMP in partially observable environments that no heuristic algorithm directly searches for the goal. Paxton et al. [144] integrate MCTS with hierarchical neural net policies trained by reinforcement learning to achieve dynamic self-driving tasks. The state-space of the planning system is augmented with deterministic Rabin automaton [41] formulated memory. The policies are trained separately with a set of constraints represented by LTL formulations, such as permitted and prohibited system actions. MCTS recursively descends through the tree from the current state and chooses a high-level option according to the Upper Confidence Bound metric. Then Progressive Widening [35] is used to add a new node and explore the branch until triggers a termination condition. Moreover, the LTL feasibility check is called during the exploration of the MCTS.

Policies in the hierarchical RL-based decision-making problems are limited by the modularity and generality of different agents or unseen tasks. Christen et al. [30] propose a novel hierarchical reinforcement learning architecture HiDe to solve long-horizon planning tasks and alleviates these limitations. The RL-based planning layer generates subgoals to achieve, it uses a learned dynamic attention window to transform the task-relevant prior into a value map. The task-relevant priors are proven as significant modules for HiDe in some challenging scenarios, such as unseen test environments and large-scale environments. HiDe is functionally decomposed as a high-level plan policy and low-level control policy, but the policies are trained jointly. The low-level control policy interacts with the environment and has access to the perceptual states to achieve subgoals. Similarly, policies that are directly learned from visual input can generalize to different environments and tasks without retraining. Bradley et al. [17] propose a method to achieve task specifications represented by temporal logic in a partially observable environment. Given a **Labeled Transition System (LTS)** specified environment and **Syntactically Co-safe LTL (scLTL)** [105] formulated task, a set of high-level actions is defined in the study based on transitions of system states. Then POMDP model computes the outcome of each action and divides the actions into two classes, actions successfully make their transitions or not. They estimate the cost and success rate of each action by a neural network and optimize the actions, the network is trained with images and encoding of the transitions.

The experiences obtained from interactions with uncertain environments can be utilized by Deep Reinforcement Learning algorithms in TAMP. To search for computationally efficient solutions for long-horizon tasks with uncertainties, Newaz et al. [135] employ MDPs to formulate a hierarchical TAMP framework in the stochastic environment. The motion planner considers geometric constraints and uncertainties to obtain an optimal control policy for single-robot motions in the continuous state space. While the task planner searches for an optimal subtask policy in the discrete state space considering both uncertainties in the task and motion layer. The task layer policy is learned through the Deep Q-Network [51] algorithm, while the motion layer policy is learned through the **Deep Deterministic Policy Gradient (DDPG)** curriculum learning algorithm [114]. Gieselmann and Pokorny [69] propose a **Planning-augmented Hierarchical Reinforcement Learning (PAHRL)** algorithm to solve long-horizon planning problems with unknown dynamics, such as deformable object manipulation. The method integrates hybrid planning with reinforcement learning and extends previous work [50] to make it suitable for implicitly specified goals. Hierarchical RL is used to estimate distances between different states. Then the method utilizes them to generate the corresponding state memory graphs iteratively. Finally, a classical graph-based search algorithm is implemented for computing the sequence of states pointing to the goal.

5.3.2 Learning with Incomplete Knowledge of Objects. For some real-world applications, the kinematic model or physical properties of related objects may be not informed to the planner, while the visual data from interactive perception are available for learning algorithms to complete the absent information. Give an unsegmented sequence of observed data from hybrid dynamic objects [119], A **Model Inference Conditioned on Actions for Hierarchical Planning (MICAH)** [85] is proposed to detect change points in the object kinematic model with a novel action-conditional reasoner called Act-CHAMP. MICAH then estimates the individual kinematic models with their parameters. These detections and estimations are converted into planning-compatible hybrid automaton [119] to help the hierarchical POMDP planner solve manipulation problems under uncertainty. Additionally, the *kinematic graphs* are used to represent the object kinematic model, there are six sets of demonstrations including kinesthetic data for the learning procedures. Ding et al. [43] propose the **Task Motion Object-centric planning (TMOC)** framework to handle manipulation scenarios where object properties are not known, such as block size and weight. A physics engine that describes the physical properties of grounding objects is embedded into the framework. Simulated interactive datasets of the grounding objects are used for learning procedures. With an unknown object, three features are proposed to characterize the physical model including basic physical property, state mapping function and transition function. Each of them is iteratively learned under the current estimation of the others by the TMOC algorithm. The uncertainties from incomplete knowledge are a type of unknown unknowns, which are different from uncertainties of partial observability problems. Sharma et al. [159] propose a method to handle manipulation tasks with an incomplete model in domain knowledge. Given an initial domain model, they assume that there are a set of expert demonstrations that consist of the missing domain knowledge before planning the sequence of actions. They learn from these demonstrations and search for a candidate model set that is updated from the initial domain model with a minimalistic and deterministic model assumption. Finally, a robust heuristic-based planning method searches for solutions with the highest probability of completing the task under the weighted set of candidate models.

5.4 Summary

Typical TAMP planners are not aware of leveraging the past planning data, while the pure-learning approaches in planning are data-inefficient. Although the state-of-art of learning algorithms in TAMP remain very preliminary, we have surveyed a few learning approaches for plan guidance, task-constrained motion planning and planning under uncertainties. We conclude these approaches in Table 2, most of which are learned from expert demonstrations and applied in robot manipulations and a few of them are applied in robot navigation. The main purpose of learning algorithms in TAMP remains in generalizing TAMP approaches, reducing computational efforts and learning domain models autonomously. We hope the current review of these methods can guide future research on intelligent learning algorithms in TAMP.

6 CONCLUSION AND FUTURE TRENDS

AI Planning is a central and promising research field in robotics. While TAMP is a deliberative planning method that integrates both high-level abstract reasoning and low-level feasibility

Table 2. Learning Approaches in TAMP

Reference	Learning objects	Training datas	Applications/Experiments
[101]	Score-space of solution constraints	1000 tuples of solution-constraint values	Robot manipulations
[100]	rank function and motion sampler	A fixed set of planning experience, GANS	PR2 restaurant manipulations
[178]	A variational autoencoder,control policy	Expert demonstrations	Robot manipulations in clutter
[73]	GTN	Expert demonstrations	Robot manipulations
[200]	Symbolic and geometric scene graphs	Expert demonstrations	Robot manipulations
[161]	Operators	Dataset of low-level transitions	Robot manipulations in clutter
[40]	Operators	Expert demonstrations	Robot manipulations
[112]	Task planner and motion controller	Expert demonstrations	Robot manipulations
[118]	PRM path policy	Self-generated demonstrations	Robot manipulations
[16]	Task model	Expert demonstrations	Baxter robot manipulations
[181]	Motion planning feasibility classifier	Set of labeled motion planner outputs	Robot manipulations
[148]	CoMPNetX	Expert demonstrations and observation data	Robot manipulations
[156]	HARP	20 environments formulated by RBVD	Robot navigation
[173]	Motion Policy	Hidden properties from the experiences	Robot navigation
[140]	ERT and ERTConnect	Prior experiences	Robot manipulations
[102]	ALEF	Path experiences	Robot manipulations
[117]	Constraint-based TAMP models	Expert demonstrations	Robot manipulations
[5]	Goal Score estimator	Expert demonstrations	PR2 manipulations
[144]	Hierarchical neural net policies	A set of constraints represented by LTL formulations	Self-driving applications
[30]	Hierarchical policies in TAMP	Hindsight action and hindsight goal transitions	Robot manipulations and navigation
[135]	Hierarchical policies in TAMP	All of the experiences accumulated	UAV manipulations and navigation
[17]	An action evaluation neural network	Visual inputs and encoding of the transitions	Robot navigation
[69]	learning for score-space of solution constraints	1000 tuples of solution-constraint values	Robot manipulations
[85]	The kinematic graph of MICAH	Six sets of demonstrations including kinesthetic data	Robot manipulations
[43]	Three features of the physical properties	Simulated interactive datasets of the grounding objects	Robot manipulations
[159]	Domain model	Expert demonstrations	Robot manipulations

evaluation, it generates detailed solutions that instruct the robot to complete long-horizon tasks in unstructured human environments. Intelligent robots will be more mature and comprehensively applied to human daily life when equipped with a deliberative planning system.

6.1 Conclusion

The deliberative planning system is not simply a matter of sending commands, it should capable of planning, acting and monitoring, it is more like a high-level controlling system with the structure of P-ALS. Since TAMP is an integrated planning system, the state-of-art is rich and broad but quite fragmented. The relationships between these fragments need to be studied. Numerous contributions to the deliberative planning system have been proposed in the last decade, and some of them have been presented in the preceding sections. The purpose of this article is to comprehensively survey the state-of-art in TAMP and present a detailed overview of this research field. Section 2 has discussed related surveys and distinctions between these surveys and our article. Our work presents three solutions corresponding to the Research Questions introduced in Section 1 for the key issue of implementing TAMP as a deliberative planning system in real-world applications.

First, we have shown the regular TAMP trends about *RQ1* that is how to generate detailed solutions combining task planning with motion planning. We categorize these approaches according to their concentrations. The former studies concentrate on structures of TAMP or planning solvers.

Then the researchers are interested in simplifying the hybrid planning problems to generate feasible solutions with probabilistic completeness or optimizing them. Recently, a few studies are beginning to focus on optimizing the computational efforts of TAMP and leveraging TAMP in navigation scenarios or other fields.

Second, we have sought the solutions for *RQ2*. The online methods optimize the internal structure of regular TAMP methods. We survey the reactive behavior planning, belief space TAMP and control-based TAMP approaches to deal with uncertainties. These approaches can make reactive responses to different levels of uncertainties, from tolerable disturbances to unexpected events or failures. They optimize TAMP into an online closed-loop planning system, making the system more reliable and robust.

Finally, we have presented the answers for *RQ3*. Learning algorithms are leveraged as external tools to help TAMP in symbolic planning, task-related motion planning, and handling uncertainties. TAMP approaches should formulate the model representations and search solutions in the hybrid space to generate a partial plan in a single loop of planning. Despite the features in TAMP procedures being hard to characterize, each part of the whole TAMP procedure has received positive effects from recent studies of learning algorithms. These algorithms either learn from past experiences, demonstrations, or real-time visual data, speeding up the searching processes or dealing with uncertainties.

6.2 Future Trends of Task and Motion Planning

This article has surveyed many TAMP approaches from the early 2010s and given a linear trend in the research field. However, most of the applications of TAMP approaches are implemented in robot manipulations. As a comprehensive planning system, there are many potential research fields of TAMP that can be developed.

6.2.1 Robotic Exploration. We have presented a few robotic navigation applications of TAMP approaches [46, 116, 149, 168], while robotic exploration is a branch of robotic navigation but is more complicated. The **Defense Advanced Research Projects Agency (DARPA)** has organized many competitions in the last decade to spur technology for the development of autonomous robots. The DARPA **Subterranean Challenge (SubT)** [151] is a multi-robotic Exploration competition of underground environments. The mobile robots have to detect and report the positions of specific objects, which is a complicated long-horizon task. Some excellent exploration algorithms present the idea of hierarchical exploration planning [198] and achieved good performance in the competition [21, 22]. Extending TAMP to the exploration of highly convoluted environments research field may help the robot finish the exploring mission and even more complicated tasks.

6.2.2 Multi-agent Task and Motion Planning. Cooperative **Multi-agent Planning (MAP)** is a relatively recent research field that combines the AI Planning community with the Multi-agent Systems community. MAP is more of an optimally distributed problem of solving scheme rather than the classical single-agent planning paradigm [15, 171]. With the significant progress of TAMP, TAMP has also extended to the MAP research field and presented in three ways:

- Single-robot dual-arm manipulation: Most of the surveyed works consider sequential actions and single-arm manipulation even implemented with a dual-arm robot such as a PR2 robot. However, the symbolic planner has to account for the feasible motion of both arms in bi-manual tasks [89, 164].
- Multi-robot cooperation: A complete TAMP method in multi-robot cooperation applications should reason the high-level abstract task decomposition as well as task allocation.

Multi-robot TAMP approaches pose challenges to scalability of regular TAMP approaches [11, 131, 141].

- Human-robot cooperation: TAMP approaches are increasingly integrated into an unstructured human workspace to complete collaborative tasks. The communication between human and robot is important [27, 71] and the robot should be capable of predicting human intentions during cooperation [28, 143].

6.2.3 Explainable Robot Planning. **Explainable AI Planning (XAIP)** focuses on explanations of the planning process and has received increasing interest in recent years [57, 95]. It bridges the gap between end-to-end planning algorithms and real-world applications. XAIP is composed of explanation generation and explanation communication [175]. Most of the TAMP approaches are dependent on domain representations and reasoning, while XAIP can leverage a popular approach to bring the human-defined domain model closer to the agent's model, which is called model reconciliation [25]. There are a few studies that integrate TAMP with XAIP, which have demonstrated significant communication benefits between users and planning system [115, 158].

On the whole, this survey has analyzed the recent trends in TAMP literature, giving state-of-the-art solutions for applying TAMP in real-world robots; it also points out the direction for future research. We hope that our research will enable more researchers to pay attention to TAMP and promote a rapid expansion of this field in a wide range of research directions.

ACKNOWLEDGMENTS

The authors thank the anonymous referees for their valuable and helpful feedback.

REFERENCES

[1] Alphonsus Adu-Bredu, Zhen Zeng, Neha Pusalkar, and Odest Chadwicke Jenkins. 2021. Elephants don't pack groceries: Robot task planning for low-entropy belief states. *IEEE Robot. Autom. Lett.* 7, 1 (2021), 25–32.

[2] Constructions Aeronautiques, Adele Howe, Craig Knoblock, Drew McDermott, Ashwin Ram, Manuela Veloso, Daniel Weld, David Wilkins, Anthony Barrett, Dave Christianson et al. 1998. PDDL: The planning domain definition language. *Technical Report.*

[3] Alejandro Agostini, Matteo Saveriano, Dongheui Lee, and Justus Piater. 2020. Manipulation planning using object-centered predicates and hierarchical decomposition of contextual actions. *IEEE Robot. Autom. Lett.* 5, 4 (2020), 5629–5636.

[4] Aliakbar Akbari, Fabien Lagriffoul, and Jan Rosell. 2019. Combined heuristic task and motion planning for bi-manual robots. *Auton. Robots* 43, 6 (2019), 1575–1590.

[5] Daniel Angelov, Yordan Hristov, Michael Burke, and Subramanian Ramamoorthy. 2020. Composing diverse policies for temporally extended tasks. *IEEE Robot. Autom. Lett.* 5, 2 (2020), 2658–2665.

[6] Brenna D. Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. 2009. A survey of robot learning from demonstration. *Robot. Auton. Syst.* 57, 5 (2009), 469–483.

[7] Ankuj Arora, Humbert Fiorino, Damien Pellier, Marc Métivier, and Sylvie Pesty. 2018. A review of learning planning action models. *Knowl. Eng. Rev.* 33 (2018).

[8] Christer Bäckström and Inger Klein. 1991. Planning in polynomial time: The SAS-PUBS class. *Comput. Intell.* 7, 3 (1991), 181–197.

[9] Claudine Badue, Rânik Guidolini, Raphael Vivacqua Carneiro, Pedro Azevedo, Vinicius B. Cardoso, Avelino Forechi, Luan Jesus, Rodrigo Berriel, Thiago M. Paixao, Filipe Mutz et al. 2021. Self-driving cars: A survey. *Expert Syst. Appl.* 165 (2021), 113816.

[10] F. Basile, F. Caccavale, P. Chiacchio, J. Coppola, and C. Curatella. 2012. Task-oriented motion planning for multi-arm robotic systems. *Robot. Comput.-Integr. Manufact.* 28, 5 (2012), 569–582.

[11] Patrick Bechon, Charles Lesire, and Magali Barbier. 2020. Hybrid planning and distributed iterative repair for multi-robot missions with communication losses. *Auton. Robots* 44, 3 (2020), 505–531.

[12] Piergiorgio Bertoli, Alessandro Cimatti, Marco Roveri, and Paolo Traverso. 2001. Planning in nondeterministic domains under partial observability via symbolic model checking. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'01)*, Vol. 2001. 473–478.

[13] Dimitri P. Bertsekas and John N. Tsitsiklis. 1991. An analysis of stochastic shortest path problems. *Math. Oper. Res.* 16, 3 (1991), 580–595.
[14] Devendra Bhave, Sagar Jha, Shankara Narayanan Krishna, Sven Schewe, and Ashutosh Trivedi. 2015. Bounded-rate multi-mode systems based motion planning. In *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control.* 41–50.
[15] Daniel Borrajo. 2013. Multi-agent planning by plan reuse. In *Proceedings of the International Conference on Autonomous Agents and Multi-agent Systems.* 1141–1142.
[16] Chris Bowen and Ron Alterovitz. 2018. Closed-loop global motion planning for reactive, collision-free execution of learned tasks. *ACM Trans. Hum.-Robot Interact.* 7, 1 (2018), 1–16.
[17] Christopher Bradley, Adam Pacheck, Gregory J. Stein, Sebastian Castro, Hadas Kress-Gazit, and Nicholas Roy. 2021. Learning and planning for temporally extended tasks in unknown environments. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'21).* IEEE, 4830–4836.
[18] Ronen I. Brafman, Jean-Claude Latombe, Yoram Moses, and Yoav Shoham. 1997. Applications of a logic of knowledge to motion planning under uncertainty. *J. ACM* 44, 5 (1997), 633–668.
[19] Gerhard Brewka, Thomas Eiter, and Mirosław Truszczyński. 2011. Answer set programming at a glance. *Commun. ACM* 54, 12 (2011), 92–103.
[20] Daniel Bryce, Subbarao Kambhampati, and David E. Smith. 2006. Planning graph heuristics for belief space search. *J. Artific. Intell. Res.* 26 (2006), 35–99.
[21] Chao Cao, Hongbiao Zhu, Howie Choset, and Ji Zhang. 2021. TARE: A hierarchical framework for efficiently exploring complex 3D environments. In *Robotics: Science and Systems.*
[22] Chao Cao, Hongbiao Zhu, Fan Yang, Yukun Xia, Howie Choset, Jean Oh, and Ji Zhang. 2021. Autonomous exploration development environment and the planning algorithms. Retrieved from https://arXiv:2110.14573 (2021).
[23] Michael Cashmore, Maria Fox, Derek Long, Daniele Magazzeni, Bram Ridder, Arnau Carrera, Narcis Palomeras, Natalia Hurtos, and Marc Carreras. 2015. Rosplan: Planning in the robot operating system. In *Proceedings of the International Conference on Automated Planning and Scheduling*, Vol. 25. 333–341.
[24] Nicola Castaman, Enrico Pagello, Emanuele Menegatti, and Alberto Pretto. 2021. Receding horizon task and motion planning in changing environments. *Robot. Auton. Syst.* 145 (2021), 103863.
[25] Tathagata Chakraborti, Sarath Sreedharan, Yu Zhang, and Subbarao Kambhampati. 2017. Plan explanations as model reconciliation: Moving beyond explanation as soliloquy. Retrieved from https://arXiv:1701.08317.
[26] Jingkai Chen, Brian C. Williams, and Chuchu Fan. 2021. Optimal mixed discrete-continuous planning for linear hybrid systems. In *Proceedings of the 24th International Conference on Hybrid Systems: Computation and Control.* 1–12.
[27] Xiaoping Chen, Jianmin Ji, Jiehui Jiang, Guoqiang Jin, Feng Wang, and Jiongkun Xie. 2010. Developing high-level cognitive functions for service robots. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'10)*, Vol. 10. 989–996.
[28] Yujiao Cheng, Liting Sun, and Masayoshi Tomizuka. 2021. Human-aware robot task planning based on a hierarchical task model. *IEEE Robot. Autom. Lett.* 6, 2 (2021), 1136–1143.
[29] Sachin Chitta, E. Gil Jones, Matei Ciocarlie, and Kaijen Hsiao. 2012. Perception, planning, and execution for mobile manipulation in unstructured environments. *IEEE Robot. Autom. Mag., Special Iss. Mobile Manip.* 19, 2 (2012), 58–71.
[30] Sammy Christen, Lukas Jendele, Emre Aksan, and Otmar Hilliges. 2021. Learning functionally decomposed hierarchies for continuous control tasks with path planning. *IEEE Robot. Autom. Lett.* 6, 2 (2021), 3623–3630.
[31] Jesse Clifton and Eric Laber. 2020. Q-learning: Theory and applications. *Annu. Rev. Stat. Appl.* 7 (2020), 279–301.
[32] Amanda Coles, Andrew Coles, Maria Fox, and Derek Long. 2010. Forward-chaining partial-order planning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, Vol. 20. 42–49.
[33] Michele Colledanchise and Lorenzo Natale. 2021. On the implementation of behavior trees in robotics. *IEEE Robot. Autom. Lett.* 6, 3 (2021), 5929–5936.
[34] Michele Colledanchise and Petter Ögren. 2018. *Behavior Trees in Robotics and AI: An Introduction.* CRC Press.
[35] Adrien Couëtoux, Mario Milone, Matyas Brendel, Hassan Doghmen, Michele Sebag, and Olivier Teytaud. 2011. Continuous rapid action value estimates. In *Proceedings of the Asian Conference on Machine Learning.* PMLR, 19–31.
[36] Neil T. Dantam. 2020. *Task and Motion Planning.* Springer, Berlin, 1–9. DOI: https://doi.org/10.1007/978-3-642-41610-1_176-1
[37] Neil T. Dantam, Swarat Chaudhuri, and Lydia E. Kavraki. 2018. The task-motion kit: An open source, general-purpose task and motion-planning framework. *IEEE Robot. Autom. Mag.* 25, 3 (2018), 61–70.
[38] Neil T. Dantam, Zachary K. Kingston, Swarat Chaudhuri, and Lydia E. Kavraki. 2016. Incremental task and motion planning: A constraint-based approach. In *Robotics: Science and Systems*, Vol. 12. Ann Arbor, MI, 00052.
[39] Neil T. Dantam, Zachary K. Kingston, Swarat Chaudhuri, and Lydia E. Kavraki. 2018. An incremental constraint-based framework for task and motion planning. *Int. J. Robot. Res.* 37, 10 (2018), 1134–1151.

[40] Maximilian Diehl, Chris Paxton, and Karinne Ramirez-Amaro. 2021. Automated generation of robotic planning domains from observations. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'21).* IEEE, 6732–6738.

[41] Xuchu Ding, Stephen L. Smith, Calin Belta, and Daniela Rus. 2014. Optimal control of Markov decision processes with linear temporal logic constraints. *IEEE Trans. Automat. Control* 59, 5 (2014), 1244–1257.

[42] Yan Ding, Xiaohan Zhang, Xingyue Zhan, and Shiqi Zhang. 2020. Task-motion planning for safe and efficient urban driving. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'20).* IEEE, 2119–2125.

[43] Yan Ding, Xiaohan Zhang, Xingyue Zhan, and Shiqi Zhang. 2022. Learning to ground objects for robot task and motion planning. *IEEE Robot. Autom. Lett.* 7, 2 (2022), 5536–5543.

[44] Christian Dornhege. 2015. *Task planning for high-level robot control.* Ph.D. Dissertation. Verlag nicht ermittelbar.

[45] Christian Dornhege, Patrick Eyerich, Thomas Keller, Sebastian Trüg, Michael Brenner, and Bernhard Nebel. 2009. Semantic attachments for domain-independent planning systems. In *Proceedings of the 19th International Conference on Automated Planning and Scheduling.*

[46] Stefan Edelkamp, Morteza Lahijanian, Daniele Magazzeni, and Erion Plaku. 2018. Integrating temporal reasoning and sampling-based motion planning for multigoal problems with dynamics and time windows. *IEEE Robot. Autom. Lett.* 3, 4 (2018), 3473–3480.

[47] Lasse Einig, Denis Klimentjew, Sebastian Rockel, Liwei Zhang, and Jianwei Zhang. 2013. Parallel plan execution and re-planning on a mobile robot using state machines with htn planning systems. In *Proceedings of the IEEE International Conference on Robotics and Biomimetics (ROBIO'13).* IEEE, 151–157.

[48] Kutluhan Erol, James Hendler, and Dana S. Nau. 1994. HTN planning: Complexity and expressivity. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI'94)*, Vol. 94. 1123–1128.

[49] Kutluhan Erol, James A. Hendler, and Dana S. Nau. 1994. UMCP: A sound and complete procedure for hierarchical task-network planning. In *Proceedings of the 2nd International Conference on Artificial Intelligence Planning Systems (AIPS'94) Aips*, Vol. 94. 249–254.

[50] Ben Eysenbach, Russ R. Salakhutdinov, and Sergey Levine. 2019. Search on the replay buffer: Bridging planning and reinforcement learning. *Advances in Neural Information Processing Systems* 32 (2019).

[51] Jianqing Fan, Zhaoran Wang, Yuchen Xie, and Zhuoran Yang. 2020. A theoretical analysis of deep Q-learning. In *Learning for Dynamics and Control.* PMLR, 486–489.

[52] Bin Fang, Shidong Jia, Di Guo, Muhua Xu, Shuhuan Wen, and Fuchun Sun. 2019. Survey of imitation learning for robotic manipulation. *Int. J. Intell. Robot. Appl.* 3, 4 (2019), 362–369.

[53] Marco Faroni, Manuel Beschi, Stefano Ghidini, Nicola Pedrocchi, Alessandro Umbrico, Andrea Orlandini, and Amedeo Cesta. 2020. A layered control approach to human-aware task and motion planning for human-robot collaboration. In *Proceedings of the 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN'20).* IEEE, 1204–1210.

[54] Richard E. Fikes and Nils J. Nilsson. 1971. STRIPS: A new approach to the application of theorem proving to problem solving. *Artific. Intell.* 2, 3-4 (1971), 189–208.

[55] Maria Fox and Derek Long. 2002. PDDL+: Modeling continuous time dependent effects. In *Proceedings of the 3rd International NASA Workshop on Planning and Scheduling for Space*, Vol. 4. 34.

[56] Maria Fox and Derek Long. 2003. PDDL2. 1: An extension to PDDL for expressing temporal planning domains. *J. Artific. Intell. Res.* 20 (2003), 61–124.

[57] Maria Fox, Derek Long, and Daniele Magazzeni. 2017. Explainable planning. Retrieved from https://arXiv:1709.10256.

[58] Jie Fu and Ufuk Topcu. 2015. Pareto efficiency in synthesizing shared autonomy policies with temporal logic constraints. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'15).* IEEE, 361–368.

[59] Caelan Reed Garrett, Rohan Chitnis, Rachel Holladay, Beomjoon Kim, Tom Silver, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. 2020. Integrated task and motion planning. Retrieved from https://arXiv:2010.01083.

[60] Caelan Reed Garrett, Tomas Lozano-Perez, and Leslie Pack Kaelbling. 2018. Ffrob: Leveraging symbolic planning for efficient task and motion planning. *Int. J. Robot. Res.* 37, 1 (2018), 104–136.

[61] Caelan Reed Garrett, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. 2018. Sampling-based methods for factored task and motion planning. *Int. J. Robot. Res.* 37, 13-14 (2018), 1796–1825.

[62] Caelan Reed Garrett, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. 2018. Stripstream: Integrating symbolic planners and blackbox samplers. Retrieved from https://arXiv:1802.08705.

[63] Caelan Reed Garrett, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. 2020. Pddlstream: Integrating symbolic planners and blackbox samplers via optimistic adaptive planning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, Vol. 30. 440–448.

[64] Caelan Reed Garrett, Chris Paxton, Tomás Lozano-Pérez, Leslie Pack Kaelbling, and Dieter Fox. 2020. Online replanning in belief space for partially observable task and motion problems. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'20).* IEEE, 5678–5684.

[65] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. 2019. Multi-shot ASP solving with clingo. *Theory Pract. Logic Program.* 19, 1 (2019), 27–82.

[66] Alfonso E. Gerevini, Patrik Haslum, Derek Long, Alessandro Saetti, and Yannis Dimopoulos. 2009. Deterministic planning in the fifth international planning competition: PDDL3 and experimental evaluation of the planners. *Artificial Intelligence* 173, 5–6 (2009), 619–668. https://www.sciencedirect.com/science/article/pii/S0004370208001847?via%3Dihub.

[67] Malik Ghallab, Dana Nau, and Paolo Traverso. 2016. *Automated Planning and Acting.* Cambridge University Press.

[68] Razan Ghzouli, Thorsten Berger, Einar Broch Johnsen, Swaib Dragule, and Andrzej Wąsowski. 2020. Behavior trees in action: A study of robotics applications. In *Proceedings of the 13th ACM SIGPLAN International Conference on Software Language Engineering.* 196–209.

[69] Robert Gieselmann and Florian T. Pokorny. 2021. Planning-augmented hierarchical reinforcement learning. *IEEE Robot. Autom. Lett.* 6, 3 (2021), 5097–5104.

[70] Michael Görner, Robert Haschke, Helge Ritter, and Jianwei Zhang. 2019. Moveit! Task constructor for task-level motion planning. In *Proceedings of the International Conference on Robotics and Automation (ICRA'19).* IEEE, 190–196.

[71] Elena Corina Grigore and Brian Scassellati. 2016. Constructing policies for supportive behaviors and communicative actions in human-robot teaming. In *Proceedings of the 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI'16).* IEEE, 615–616.

[72] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C. Courville. 2017. Improved training of wasserstein gans. *Advances in Neural Information Processing Systems* 30 (2017).

[73] Meng Guo and Mathias Bürger. 2022. Geometric task networks: Learning efficient and explainable skill coordination for object manipulation. *IEEE Trans. Robot.* 38, 3 (2022), 1723–1734. DOI: 10.1109/TRO.2021.3111481

[74] Meng Guo and Michael M. Zavlanos. 2018. Probabilistic motion planning under temporal tasks and soft constraints. *IEEE Trans. Automat. Control* 63, 12 (2018), 4051–4066.

[75] Himanshu Gupta, Bradley Hayes, and Zachary Sunberg. 2022. Intention-aware navigation in crowds with extended-space POMDP planning. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems.* 562–570.

[76] Dylan Hadfield-Menell, Edward Groshev, Rohan Chitnis, and Pieter Abbeel. 2015. Modular task and motion planning in belief space. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'15).* IEEE, 4991–4998.

[77] Valentin N. Hartmann, Ozgur S. Oguz, Danny Driess, Marc Toussaint, and Achim Menges. 2020. Robust task and motion planning for long-horizon architectural construction planning. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'20).* IEEE, 6886–6893.

[78] Keliang He, Morteza Lahijanian, Lydia E. Kavraki, and Moshe Y. Vardi. 2015. Towards manipulation planning with temporal logic specifications. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'15).* IEEE, 346–352.

[79] Malte Helmert. 2006. The fast downward planning system. *J. Artific. Intell. Res.* 26 (2006), 191–246.

[80] Mengxue Hou, Tony X. Lin, Haomin Zhou, Wei Zhang, Catherine R. Edwards, and Fumin Zhang. 2021. Belief space partitioning for symbolic motion planning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'21).* IEEE, 8245–8251.

[81] Yijiang Huang, Pok Yin Victor Leung, Caelan Garrett, Fabio Gramazio, Matthias Kohler, and Caitlin Mueller. 2021. The new analog: A protocol for linking design and construction intent with algorithmic planning for robotic assembly of complex structures. In *Proceedings of the Symposium on Computational Fabrication.* 1–17.

[82] Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. 2017. Imitation learning: A survey of learning methods. *ACM Comput. Surv.* 50, 2, Article 21 (Apr 2017), 35 pages. DOI: https://doi.org/10.1145/3054912

[83] Yong K. Hwang and Narendra Ahuja. 1992. Gross motion planning–a survey. *ACM Comput. Surv.* 24, 3 (1992), 219–291.

[84] Félix Ingrand and Malik Ghallab. 2017. Deliberation for autonomous robots: A survey. *Artific. Intell.* 247 (2017), 10–44.

[85] Ajinkya Jain and Scott Niekum. 2020. Learning hybrid object kinematics for efficient hierarchical planning under uncertainty. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'20).* IEEE, 5253–5260.

[86] Yuqian Jiang, Fangkai Yang, Shiqi Zhang, and Peter Stone. 2019. Task-motion planning with reinforcement learning for adaptable mobile service robots. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'19).* IEEE, 7529–7534.

[87] Ziyuan Jiao, Zeyu Zhang, Weiqi Wang, David Han, Song-Chun Zhu, Yixin Zhu, and Hangxin Liu. 2021. Efficient task planning for mobile manipulation: A virtual kinematic chain perspective. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'21).* IEEE, 8288–8294.

[88] Sergio Jiménez, Tomás De La Rosa, Susana Fernández, Fernando Fernández, and Daniel Borrajo. 2012. A review of machine learning for automated planning. *Knowl. Eng. Rev.* 27, 4 (2012), 433–467.

[89] Ariyan M. Kabir, Shantanu Thakar, Prahar M. Bhatt, Rishi K. Malhan, Pradeep Rajendran, Brual C. Shah, and Satyandra K. Gupta. 2020. Incorporating motion planning feasibility considerations during task-agent assignment to perform complex tasks using mobile manipulators. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'20).* IEEE, 5663–5670.

[90] Leslie Kaelbling and Tomas Lozano-Perez. 2011. Domain and plan representation for task and motion planning in uncertain domains. In *Proceedings of the IROS Workshop on Knowledge Representation for Autonomous Robots.*

[91] Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. 1996. Reinforcement learning: A survey. *J. Artific. Intell. Res.* 4 (1996), 237–285.

[92] Leslie Pack Kaelbling and Tomás Lozano-Pérez. 2010. Hierarchical planning in the now. In *Proceedings of the the 24th AAAI Conference on Artificial Intelligence.*

[93] Leslie Pack Kaelbling and Tomás Lozano-Pérez. 2013. Integrated task and motion planning in belief space. *Int. J. Robot. Res.* 32, 9-10 (2013), 1194–1227.

[94] Subbarao Kambhampati. 1995. AI planning: A prospectus on theory and applications. *ACM Comput. Surv.* 27, 3 (1995), 334–336.

[95] Subbarao Kambhampati. 2019. Synthesizing explainable behavior for human-AI collaboration. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems.* 1–2.

[96] Bernd Kast, Vincent Dietrich, Sebastian Albrecht, Wendelin Feiten, and Jianwei Zhang. 2019. A hierarchical planner based on set-theoretic models: Towards automating the automation for autonomous systems. In *Proceedings of the 16th International Conference on Informatics in Control, Automation and Robotics (ICINCO'19).* 249–260.

[97] Bernd Kast, Philipp S. Schmitt, Sebastian Albrecht, Wendelin Feiten, and Jianwei Zhang. 2020. Hierarchical planner with composable action models for asynchronous parallelization of tasks and motions. In *Proceedings of the 4th IEEE International Conference on Robotic Computing (IRC'20).* IEEE, 143–150.

[98] Christel Kemke and Erin Walker. 2006. Planning with action abstraction and plan decomposition hierarchies. In *Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology.* IEEE, 447–451.

[99] Oussama Khatib. 1987. A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE J. Robot. Autom.* 3, 1 (1987), 43–53.

[100] Beomjoon Kim, Luke Shimanuki, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. 2022. Representation, learning, and planning algorithms for geometric task and motion planning. *Int. J. Robot. Res.* 41, 2 (2022), 210–231.

[101] Beomjoon Kim, Zi Wang, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. 2019. Learning to guide task and motion planning using score-space representation. *Int. J. Robot. Res.* 38, 7 (2019), 793–812.

[102] Zachary Kingston, Constantinos Chamzas, and Lydia E. Kavraki. 2021. Using experience to improve constrained planning on foliations for multi-modal problems. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'21).* IEEE, 6922–6927.

[103] Takuma Kogo, Kei Takaya, and Hiroyuki Oyama. 2021. Fast MILP-based task and motion planning for pick-and-place with hard/soft constraints of collision-free route. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC'21).* IEEE, 1020–1027.

[104] Sascha Kolski, Dave Ferguson, Mario Bellino, and Roland Siegwart. 2006. Autonomous driving in structured and unstructured environments. In *Proceedings of the IEEE Intelligent Vehicles Symposium.* IEEE, 558–563.

[105] Orna Kupferman and Moshe Y. Vardi. 2001. Model checking of safety properties. *Formal Methods Syst. Design* 19, 3 (2001), 291–314.

[106] Hanna Kurniawati. 2022. Partially observable markov decision processes and robotics. *Annu. Rev. Control, Robot. Auton. Syst.* 5 (2022), 253–277.

[107] Fabien Lagriffoul and Benjamin Andres. 2016. Combining task and motion planning: A culprit detection problem. *Int. J. Robot. Res.* 35, 8 (2016), 890–927.

[108] Fabien Lagriffoul, Neil T. Dantam, Caelan Garrett, Aliakbar Akbari, Siddharth Srivastava, and Lydia E. Kavraki. 2018. Platform-independent benchmarks for task and motion planning. *IEEE Robot. Autom. Lett.* 3, 4 (2018), 3765–3772.

[109] Fabien Lagriffoul, Dimitar Dimitrov, Julien Bidot, Alessandro Saffiotti, and Lars Karlsson. 2014. Efficiently combining task and motion planning using geometric constraints. *Int. J. Robot. Res.* 33, 14 (2014), 1726–1747.

[110] Jinhwi Lee, Changjoo Nam, Jonghyeon Park, and Changhwan Kim. 2021. Tree search-based task and motion planning with prehensile and non-prehensile manipulation for obstacle rearrangement in clutter. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'21).* IEEE, 8516–8522.

[111] Kenli Li, Xiaoyong Tang, and Keqin Li. 2014. Energy-efficient stochastic task scheduling on heterogeneous computing systems. *IEEE Trans. Parallel Distrib. Syst.* 25, 11 (2014), 2867–2876. DOI: https://doi.org/10.1109/TPDS.2013.270

[112] Tianyu Li, Roberto Calandra, Deepak Pathak, Yuandong Tian, Franziska Meier, and Akshara Rai. 2021. Planning in learned latent action spaces for generalizable legged locomotion. *IEEE Robot. Autom. Lett.* 6, 2 (2021), 2682–2689.

[113] Xiaolong Li, He Wang, Li Yi, Leonidas J. Guibas, A. Lynn Abbott, and Shuran Song. 2020. Category-level articulated object pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 3706–3715.

[114] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. Retrieved from https://arXiv:1509.02971.

[115] Alan Lindsay. 2019. Towards exploiting generic problem structures in explanations for automated planning. In *Proceedings of the 10th International Conference on Knowledge Capture.* 235–238.

[116] Shih-Yun Lo, Shiqi Zhang, and Peter Stone. 2018. PETLON: Planning efficiently for task-level-optimal navigation. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems.* 220–228.

[117] Joao Loula, Kelsey Allen, Tom Silver, and Josh Tenenbaum. 2020. Learning constraint-based planning models from demonstrations. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'20).* IEEE, 5410–5416.

[118] Sha Luo, Hamidreza Kasaei, and Lambert Schomaker. 2021. Self-imitation learning by planning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'21).* IEEE, 4823–4829.

[119] J. Lygeros, K. H. Johansson, S. N. Simic, Jun Zhang, and S. S. Sastry. 2003. Dynamical properties of hybrid automata. *IEEE Transactions on Automatic Contro* 48, 1 (2003), 2–17. DOI: 10.1109/TAC.2002.806650

[120] Nancy Lynch, Roberto Segala, and Frits Vaandrager. 2003. Hybrid i/o automata. *Info. Comput.* 185, 1 (2003), 105–157.

[121] Mentar Mahmudi and Marcelo Kallmann. 2015. Multi-modal data-driven motion planning and synthesis. In *Proceedings of the 8th ACM SIGGRAPH Conference on Motion in Games.* 119–124.

[122] Shlomi Maliah, Radimir Komarnitski, and Guy Shani. 2022. Computing contingent plan graphs using online planning. *ACM Trans. Auton. Adapt. Syst.* 16, 1 (2022), 1–30.

[123] Matthew R. Maly, Morteza Lahijanian, Lydia E. Kavraki, Hadas Kress-Gazit, and Moshe Y. Vardi. 2013. Iterative temporal motion planning for hybrid systems in partially unknown environments. In *Proceedings of the 16th International Conference on Hybrid Systems: Computation and Control.* 353–362.

[124] Masoumeh Mansouri and Federico Pecora. 2014. More knowledge on the table: Planning with space, time and resources for robots. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'14).* IEEE, 647–654.

[125] Masoumeh Mansouri, Federico Pecora, and Peter Schüller. 2021. Combining task and motion planning: Challenges and guidelines. *Front. Robot. AI* 8 (2021), 133.

[126] Francisco Martín Rico, Matteo Morelli, Huascar Espinoza, Francisco J. Rodríguez-Lera, and Vicente Matellán Olivera. 2021. Optimized execution of PDDL plans using behavior trees. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems.* 1596–1598.

[127] Ruben Martinez-Cantin, Nando de Freitas, Arnaud Doucet, and José A. Castellanos. 2007. Active policy learning for robot planning and exploration under uncertainty. In *Robotics: Science and Systems*, Vol. 3. 321–328.

[128] Toki Migimatsu and Jeannette Bohg. 2020. Object-centric task and motion planning in dynamic environments. *IEEE Robot. Autom. Lett.* 5, 2 (2020), 844–851.

[129] Joseph Mirabel and Florent Lamiraux. 2016. Constraint graphs: Unifying task and motion planning for navigation and manipulation among movable obstacles. https://hal.science/hal-01281348v1.

[130] Reuth Mirsky, Kobi Gal, Roni Stern, and Meir Kalech. 2019. Goal and plan recognition design for plan libraries. *ACM Trans. Intell. Syst. Technol.* 10, 2 (2019), 1–23.

[131] James Motes, Read Sandström, Hannah Lee, Shawna Thomas, and Nancy M. Amato. 2020. Multi-robot task and motion planning with subtask dependencies. *IEEE Robot. Autom. Lett.* 5, 2 (2020), 3338–3345.

[132] Shohin Mukherjee, Sandip Aine, and Maxim Likhachev. 2022. MPLP: Massively parallelized lazy planning. *IEEE Robot. Autom. Lett.* 7, 3 (2022), 6067–6074.

[133] Dana Nau, Yue Cao, Amnon Lotem, and Hector Munoz-Avila. 1999. SHOP: Simple hierarchical ordered planner. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence.* 968–973.

[134] Dana S. Nau, Tsz-Chiu Au, Okhtay Ilghami, Ugur Kuter, J. William Murdock, Dan Wu, and Fusun Yaman. 2003. SHOP2: An HTN planning system. *J. Artific. Intell. Res.* 20 (2003), 379–404.

[135] Abdullah Al Redwan Newaz and Tauhidul Alam. 2021. Hierarchical task and motion planning through deep reinforcement learning. In *Proceedings of the 5th IEEE International Conference on Robotic Computing (IRC'21).* IEEE, 100–105.

[136] Nils J. Nilsson et al. 1984. Shakey the Robot. Tech. Rep. 323, Artificial Intelligence Center, SRI International.

[137] Jian Niu, Zhengqiong Liu, Zhizhong Ding, and Momiao Zhou. 2021. Velocity planning for autonomous vehicle. In *Proceedings of the 3rd International Conference on Information Technology and Computer Communications.* 57–62.

[138] Mahda Noura and Martin Gaedke. 2019. An automated cyclic planning framework based on plan-do-check-act for web of things composition. In *Proceedings of the 10th ACM Conference on Web Science.* 205–214.

[139] Jun Ota. 2004. Rearrangement of multiple movable objects-integration of global and local planning methodology. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'04)*, Vol. 2. IEEE, 1962–1967.

[140] Èric Pairet, Constantinos Chamzas, Yvan Petillot, and Lydia E. Kavraki. 2021. Path planning for manipulation using experience-driven random trees. *IEEE Robot. Autom. Lett.* 6, 2 (2021), 3295–3302.

[141] Tianyang Pan, Andrew M. Wells, Rahul Shome, and Lydia E. Kavraki. 2021. A general task and motion planning framework for multiple manipulators. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'21)*. IEEE, 3168–3174.

[142] Yudha Pane, Vahid Mokhtari, Erwin Aertbeliën, Joris De Schutter, and Wilm Decré. 2021. Autonomous runtime composition of sensor-based skills using concurrent task planning. *IEEE Robot. Autom. Lett.* 6, 4 (2021), 6481–6488.

[143] David Paulius, Kelvin Sheng Pei Dong, and Yu Sun. 2021. Task planning with a weighted functional object-oriented network. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'21)*. IEEE, 3904–3910.

[144] Chris Paxton, Vasumathi Raman, Gregory D. Hager, and Marin Kobilarov. 2017. Combining neural networks and tree search for task and motion planning in challenging environments. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'17)*. IEEE, 6059–6066.

[145] Judea Pearl. 1996. Decision making under uncertainty. *ACM Comput. Surv.* 28, 1 (1996), 89–92.

[146] Camille Phiquepal and Marc Toussaint. 2019. Combined task and motion planning under partial observability: An optimization-based approach. In *Proceedings of the International Conference on Robotics and Automation (ICRA'19)*. IEEE, 9000–9006.

[147] Martin L. Puterman. 1990. Markov decision processes. *Handbooks in Operations Research and Management Science* 2 (1990), 331–434.

[148] Ahmed Hussain Qureshi, Jiangeng Dong, Asfiya Baig, and Michael C. Yip. 2021. Constrained motion planning networks x. *IEEE Trans. Robot.* 38, 2 (2021), 868–886.

[149] Gayathri Rajendran, V. Uma, and Bettina O'Brien. 2022. Unified robot task and motion planning with extended planner using ROS simulator. *J. King Saud Univ.-Comput. Info. Sci.* 34, 9 (2022), 7468–7481.

[150] Tianyu Ren, Georgia Chalvatzaki, and Jan Peters. 2021. Extended task and motion planning of long-horizon robot manipulation. Retrieved from https://arXiv:2103.05456.

[151] Tomáš Rouček, Martin Pecka, Petr Čížek, Tomáš Petříček, Jan Bayer, Vojtěch Šalanskỳ, Daniel Heřt, Matěj Petrlík, Tomáš Báča, Vojěch Spurnỳ et al. 2019. Darpa subterranean challenge: Multi-robotic exploration of underground environments. In *Proceedings of the International Conference on Modelling and Simulation for Autonomous Systems*. Springer, 274–290.

[152] Francesco Rovida, Bjarne Grossmann, and Volker Krüger. 2017. Extended behavior trees for quick definition of flexible robotic tasks. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'17)*. IEEE, 6793–6800.

[153] Evgenii Safronov, Michele Colledanchise, and Lorenzo Natale. 2020. Task planning with belief behavior trees. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'20)*. IEEE, 6870–6877.

[154] Evgenii Safronov, Michael Vilzmann, Dzmitry Tsetserukou, and Konstantin Kondak. 2019. Asynchronous behavior trees with memory aimed at aerial vehicles with redundancy in flight controller. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'19)*. IEEE, 3113–3118.

[155] Sayan Saha and Anak Agung Julius. 2017. Task and motion planning for manipulator arms with metric temporal logic specifications. *IEEE Robot. Autom. Lett.* 3, 1 (2017), 379–386.

[156] Naman Shah and Siddharth Srivastava. 2022. Using deep learning to bootstrap abstractions for hierarchical robot planning. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*. 1183–1191.

[157] Naman Shah, Deepak Kala Vasudevan, Kislay Kumar, Pranav Kamojjhala, and Siddharth Srivastava. 2020. Anytime integrated task and motion policies for stochastic environments. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'20)*. IEEE, 9285–9291.

[158] Naman Shah, Pulkit Verma, Trevor Angle, and Siddharth Srivastava. 2022. JEDAI: A system for skill-aligned explainable robot planning. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*. 1917–1919.

[159] Akshay Sharma, Piyush Rajesh Medikeri, and Yu Zhang. 2021. Domain concretization from examples: Addressing missing domain knowledge via robust planning. *IEEE Robot. Autom. Lett.* 7, 2 (2021), 1032–1039.

[160] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton et al. 2017. Mastering the game of go without human knowledge. *Nature* 550, 7676 (2017), 354–359.

[161] Tom Silver, Rohan Chitnis, Joshua Tenenbaum, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. 2021. Learning symbolic operators for task and motion planning. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'21)*. IEEE, 3182–3189.

[162] Siddharth Srivastava, Eugene Fang, Lorenzo Riano, Rohan Chitnis, Stuart Russell, and Pieter Abbeel. 2014. Combined task and motion planning through an extensible planner-independent interface layer. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'14).* IEEE, 639–646.

[163] Sebastian Stock, Masoumeh Mansouri, Federico Pecora, and Joachim Hertzberg. 2015. Online task merging with a hierarchical hybrid task planner for mobile service robots. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'15).* IEEE, 6459–6464.

[164] Alejandro Suárez-Hernández, Guillem Alenyà, and Carme Torras. 2018. Interleaving hierarchical task planning and motion constraint testing for dual-arm manipulation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'18).* IEEE, 4061–4066.

[165] Ioan A. Şucan and Lydia E. Kavraki. 2012. Accounting for uncertainty in simultaneous task and motion planning using task motion multigraphs. In *Proceedings of the IEEE International Conference on Robotics and Automation.* IEEE, 4822–4828.

[166] Richard S. Sutton and Andrew G. Barto. 2018. *Reinforcement Learning: An Introduction.* MIT press.

[167] Kartik Talamadupula, J. Benton, Subbarao Kambhampati, Paul Schermerhorn, and Matthias Scheutz. 2010. Planning for human-robot teaming in open worlds. *ACM Trans. Intell. Syst. Technol.* 1, 2 (2010), 1–24.

[168] Antony Thomas, Fulvio Mastrogiovanni, and Marco Baglietto. 2021. MPTP: Motion-planning-aware task planning for navigation in belief space. *Robot. Auton. Syst.* 141 (2021), 103786.

[169] Wil Thomason and Ross A. Knepper. 2019. A unified sampling-based approach to integrated task and motion planning. In *Proceedings of the International Symposium of Robotics Research.* Springer, 773–788.

[170] Sebastian Thrun. 2002. Probabilistic robotics. *Commun. ACM* 45, 3 (2002), 52–57.

[171] Alejandro Torreno, Eva Onaindia, Antonín Komenda, and Michal Štolba. 2017. Cooperative multi-agent planning: A survey. *ACM Comput. Surv.* 50, 6 (2017), 1–32.

[172] Marc A. Toussaint, Kelsey Rebecca Allen, Kevin A. Smith, and Joshua B. Tenenbaum. 2018. Differentiable physics and stable modes for tool-use and manipulation planning. https://www.engineeringvillage.com/app/doc/?.

[173] Florence Tsang, Tristan Walker, Ryan A. MacDonald, Armin Sadeghi, and Stephen L. Smith. 2021. LAMP: Learning a motion policy to repeatedly navigate in an uncertain environment. *IEEE Trans. Robot.* 38, 3 (2021), 1638–1652.

[174] Alessandro Umbrico, Amedeo Cesta, Marta Cialdea Mayer, and Andrea Orlandini. 2017. PLATINU m: A new framework for planning and acting. In *Proceedings of the Conference of the Italian Association for Artificial Intelligence.* Springer, 498–512.

[175] Stylianos Loukas Vasileiou, William Yeoh, Tran Cao Son, Ashwin Kumar, Michael Cashmore, and Dianele Magazzeni. 2022. A logic-based explanation generation framework for classical and hybrid planning problems. *J. Artific. Intell. Res.* 73 (2022), 1473–1534.

[176] Vasileios Vasilopoulos, Sebastian Castro, William Vega-Brown, Daniel E. Koditschek, and Nicholas Roy. 2022. Technical report: A hierarchical deliberative-reactive system architecture for task and motion planning in partially known environments. Retrieved from https://arXiv:2202.01385.

[177] Weiwei Wan, Takeyuki Kotaka, and Kensuke Harada. 2022. Arranging test tubes in racks using combined task and motion planning. *Robot. Auton. Syst.* 147 (2022), 103918.

[178] Lirui Wang, Xiangyun Meng, Yu Xiang, and Dieter Fox. 2022. Hierarchical policies for cluttered-scene grasping with latent plans. *IEEE Robot. Autom. Lett.* 7, 2 (2022), 2883–2890.

[179] Zi Wang, Caelan Reed Garrett, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. 2018. Active model learning and diverse action sampling for task and motion planning. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'18).* IEEE, 4107–4114.

[180] Zi Wang, Caelan Reed Garrett, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. 2021. Learning compositional models of robot skills for task and motion planning. *Int. J. Robot. Res.* 40, 6-7 (2021), 866–894.

[181] Andrew M. Wells, Neil T. Dantam, Anshumali Shrivastava, and Lydia E. Kavraki. 2019. Learning feasibility for task and motion planning in tabletop environments. *IEEE Robot. Autom. Lett.* 4, 2 (2019), 1255–1262.

[182] Tongfeng Weng, Xu Zhou, Kenli Li, Peng Peng, and Keqin Li. 2022. Efficient distributed approaches to core maintenance on large dynamic graphs. *IEEE Trans. Parallel Distrib. Syst.* 33, 1 (2022), 129–143. DOI: https://doi.org/10.1109/TPDS.2021.3090759

[183] Tongfeng Weng, Xu Zhou, Kenli Li, Kian-Lee Tan, and Keqin Li. 2023. Distributed approaches to butterfly analysis on large dynamic bipartite graphs. *IEEE Trans. Parallel Distrib. Syst.* 34, 2 (2023), 431–445. DOI: https://doi.org/10.1109/TPDS.2022.3221821

[184] Martin Weser, Dominik Off, and Jianwei Zhang. 2010. HTN robot planning in partially observable dynamic environments. In *Proceedings of the IEEE International Conference on Robotics and Automation.* IEEE, 1505–1510.

[185] Jan Oliver Winkler and Michael Beetz. 2015. Generalized plan design for autonomous mobile manipulation in open environments. In *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'15).* 1891–1892.

[186] Guoqing Xiao, Kenli Li, Yuedan Chen, Wangquan He, Albert Y. Zomaya, and Tao Li. 2021. CASpMV: A customized and accelerative SpMV framework for the sunway TaihuLight. *IEEE Trans. Parallel Distrib. Syst.* 32, 1 (2021), 131–146. DOI: https://doi.org/10.1109/TPDS.2019.2907537

[187] Shuo Yang, Xinjun Mao, and Wanwei Liu. 2020. Towards an extended pomdp planning approach with adjoint action model for robotic task. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC'20).* IEEE, 1412–1419.

[188] Shuo Yang, Xinjun Mao, Shuo Wang, Huaiyu Xiao, and Yuanzhou Xue. 2021. Towards adjoint sensing and acting schemes and interleaving task planning for robust robot plan. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'21).* IEEE, 13791–13797.

[189] Shuo Yang, Xinjun Mao, Yuanzhou Xue, Huaiyu Xiao, and Shuo Wang. 2021. Towards a hybrid-ASP planning approach with adjoint observation for incomplete task-relevant information. *IEEE Robot. Autom. Lett.* 7, 1 (2021), 494–501.

[190] Yuanyuan Zeng, Kenli Li, Xu Zhou, Wensheng Luo, and Yunjun Gao. 2022. An efficient index-based approach to distributed set reachability on small-world graphs. *IEEE Trans. Parallel Distrib. Syst.* 33, 10 (2022), 2358–2371. DOI: https://doi.org/10.1109/TPDS.2021.3139111

[191] Chongjie Zhang and Julie A. Shah. 2016. Co-optimizing task and motion planning. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'16).* IEEE, 4750–4756.

[192] Xiaohan Zhang, Yifeng Zhu, Yan Ding, Yuke Zhu, Peter Stone, and Shiqi Zhang. 2022. Visually grounded task and motion planning for mobile manipulation. Retrieved from https://arXiv:2202.10667.

[193] Yifan Zhang, Jinghuai Zhang, Jindi Zhang, Jianping Wang, Kejie Lu, and Jeff Hong. 2022. Integrating algorithmic sampling-based motion planning with learning in autonomous driving. *ACM Trans. Intell. Syst. Technol.* 13, 3 (2022), 1–27.

[194] Wenrui Zhao and Weidong Chen. 2021. Hierarchical POMDP planning for object manipulation in clutter. *Robot. Auton. Syst.* 139 (2021), 103736.

[195] Yingshen Zhao, Philippe Fillatreau, Linda Elmhadhbi, Mohamed Hedi Karray, and Bernard Archimede. 2022. Semantic coupling of path planning and a primitive action of a task plan for the simulation of manipulation tasks in a virtual 3D environment. *Robot. Comput.-Integr. Manufact.* 73 (2022), 102255.

[196] Yingshen Zhao, Philippe Fillatreau, Mohamed Hedi Karray, and Bernard Archimède. 2018. An ontology-based approach towards coupling task and path planning for the simulation of manipulation tasks. In *Proceedings of the IEEE/ACS 15th International Conference on Computer Systems and Applications (AICCSA'18).* IEEE, 1–8.

[197] Kai Zhong, Zhibang Yang, Guoqing Xiao, Xingpei Li, Wangdong Yang, and Kenli Li. 2022. An efficient parallel reinforcement learning approach to cross-layer defense mechanism in industrial control systems. *IEEE Trans. Parallel Distrib. Syst.* 33, 11 (2022), 2979–2990. DOI: https://doi.org/10.1109/TPDS.2021.3135412

[198] Boyu Zhou, Yichen Zhang, Xinyi Chen, and Shaojie Shen. 2021. FUEL: Fast UAV exploration using incremental frontier structure and hierarchical planning. *IEEE Robot. Autom. Lett.* 6, 2 (2021), 779–786.

[199] Gang Zhu and Nigel Shadbolt. 1994. A hybrid approach to the automatic planning of textual structures. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING'94).*

[200] Yifeng Zhu, Jonathan Tremblay, Stan Birchfield, and Yuke Zhu. 2021. Hierarchical planning for long-horizon manipulation with geometric and symbolic scene graphs. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'21).* IEEE, 6541–6548.

Received 14 August 2022; revised 22 January 2023; accepted 30 January 2023