BTPG: A Platform and Benchmark for Behavior Tree Planning in Everyday Service Robots

Xinglin Chen, **Yishuai Cai***, **Minglong Li***, **Yunxin Mao**, **Zhou Yang**, **Wenjing Yang**, **Weixia Xu** and **Ji Wang**
College of Computer Science and Technology, National University of Defense Technology
{chenxinglin, caiyishuai, liminglong10, maoyunxin, yangzhou, wenjing.yang, xuweixia, wj}@nudt.edu.cn

Abstract

Behavior Trees (BTs) are a widely used control architecture in robotics, renowned for their robustness and safety, which are especially crucial for everyday service robots. Recently, several methods have been proposed to automatically plan BTs to accomplish specific tasks. However, existing research in BT planning lacks two main aspects: (1) the absence of a standard platform for modeling and planning BTs, along with testing benchmarks; and (2) insufficient metrics for a comprehensive evaluation of BT planning algorithms. In this paper, we propose Behavior Tree Planning Gym (BTPG), the first platform and benchmark for BT planning in everyday service robots. In BTPG, behavior nodes are represented by predicate logic, and objects are categorized to better define the predicate domains and action models. The BT planning problem is then formulated in the STRIPS style. We support four environments and three simulators with different action models, which cover most of the needs of everyday service activities. We design a dataset generator for each environment and test three state-of-the-art BT planning algorithms, as well as one proposed by us, using various common metrics. In addition, we design three advanced metrics, planning progress, region distance, and execution robustness, to gain deeper insights into these BT planning algorithms. With a standard test benchmark, we hope BTPG can inspire and accelerate progress in the field of BT planning. Our codes are available at https://github.com/DIDS-EI/BTPG.

1 Introduction

One of the main goals of robotics and AI is to develop intelligent robots capable of autonomously making decisions and executing behaviors to accomplish various tasks. This requires not only intelligent planning algorithms, but also a safe and robust control architecture, which is particularly crucial for everyday service robots. Behavior Trees (BTs) have become a popular control architecture for robots exactly due to their ability to ensure safety and robustness [Colledanchise and Ögren, 2018; Ögren and Sprague, 2022]. Because of their modular and adaptable tree structure, BTs can effectively perform various tasks and handle uncertain environments [Colledanchise *et al.*, 2019a]. In addition, their clear and interpretable control flows enhance the reliability and predictability of robot behavior, making BT systems easy for humans to design, deploy, and scale.

However, despite the advantages of BTs, researches on automatic BT generation has been relatively slow to progress and yet far from practical application. Recently, BT planning [Cai *et al.*, 2021; Chen *et al.*, 2024; Cai *et al.*, 2025b; Colledanchise *et al.*, 2019a; Cai *et al.*, 2025a] has emerged as a popular approach to automatically construct BTs toward specific goals (as illustrated in Figure 1). This approach is based on easily designed action models and can theoretically guarantee the success of planned BTs. Typically, when using interpretable representations of BT nodes, such as predicate logic, the goal conditions and heuristics for BT planning can be well interpreted and reasoned about using Large Language Models (LLMs) [Chen *et al.*, 2024; Cai *et al.*, 2025b]. These advantages make BT planning a promising approach for automatic BT generation in everyday service robots.

However, to the best of our knowledge, existing BT planning methods mainly face two significant challenges: (1) Prior BT planning research constructs experiments on various platforms, and their simulation environments are quite naive. This makes it difficult to conduct a unified evaluation of existing BT algorithms. Therefore, a standard platform for real-life environments and evaluation datasets is urgently needed for this field to progress. (2) Current performance evaluation metrics focus primarily on planning efficiency and execution efficiency. However, some important attributes of the planned BTs, such as the attraction region and robustness, are less emphasized.

To address these two issues, we propose Behavior Tree Planning Gym (BTPG), the first platform and benchmark for BT planning in everyday service robots. To establish BTPG as a standard for different BT planning methods, we focus on the following three efforts: (1) We unify the BT representation by adopting the popular predicate logic form [Chen *et al.*, 2024] for BT nodes. To efficiently define the domain of each action and condition predicate, we group objects into categories based on their attributes. Subsequently, the BT

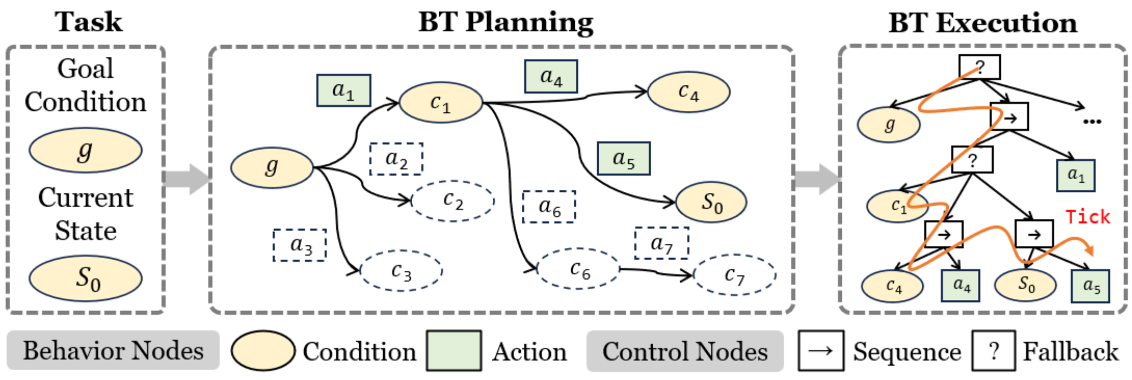

Figure 1: BT planning often searches from the goal condition backward to the current state using the action model. The explored condition and action nodes ultimately form the BT that achieves the goal.

planning problems are formulated using STRIPS-style action models, which are general enough to model common environments in everyday service scenarios. (2) We build BT systems in four environments: RoboWaiter [Chen *et al.*, 2024], VirtualHome [Puig *et al.*, 2018], OmniGibson [Li *et al.*, 2023], and RobotHow [Liao *et al.*, 2019]. The first three environments each include a simulator with which the robot can actually interact, while the last one is used for large-scale computational tests. We design action models in each environment for BT planning respectively. (3) We design a dataset generator for each environment that can automatically generate random task goals while guaranteeing their reasonableness. Then we perform comprehensive evaluations of three state-of-the-art BT planning algorithms, as well as one proposed by us, using various common metrics. Furthermore, we design three advanced metrics – planning progress, region distance, and execution robustness – to gain deeper insights into these BT planning algorithms. Through the efforts described above, we hope BTPG can become a standard platform and benchmark that can inspire and accelerate progress in BT planning research.

2 Background

A BT is a directed rooted tree where the behavior nodes (leaves) manage the robot's perception and execution, while control nodes (internals) manage the ticking logic flow to determine which action should be executed in the current state [Colledanchise and Ögren, 2018]. Current BT planning approaches focus mainly on four types of BT nodes:

- Condition. A behavior node that checks whether the specified condition holds in the current state, and returns either `success` or `failure`.
- Action. A behavior node that controls the robot to perform the specified action, and returns either `success`, `failure`, or `running`.
- Sequence. A control node that ticks its children from left to right and returns `success` when all its children succeed. Otherwise, it returns `failure` or `running` according to the first `non-success` status it encounters.
- Fallback. A control node with opposite return logic to the sequence node, which means it ticks its children from left to right and returns `failure` when all its children fail. Otherwise, it returns `success` or `running` according to the first `non-failure` status it encounters. The typical structure of the BT is illustrated in Figure 1.

Predicate Logic for Node Representation. In the predicate logic representation [Chen *et al.*, 2024; Cai *et al.*, 2025b], condition and action nodes are represented by a tuple $< \mathcal{R}, \mathcal{Q}, \mathcal{O} >$, where $\mathcal{R}$ is the condition predicate set, $\mathcal{Q}$ is the action predicate set, and $\mathcal{O}$ is the object set. In this formulation, a condition node can be denoted as $c = r(o_1, ..., o_i), r \in \mathcal{R}$ while an action node can be denoted as $a = q(o_1, ..., o_j), q \in \mathcal{Q}$.

STRIPS-style Action Model. In the STRIPS-style planning formulation [Fikes and Nilsson, 1971]. $\mathcal{S}$ is the finite set of environment states, $\mathcal{A}$ is the finite set of actions, $\mathcal{M}$ is the action model. Both the state $s \in \mathcal{S}$ and the condition c are represented by a set of atom conditions. $c \subseteq s$ means condition c holds in the state s. The state transition affected by action $a \in \mathcal{A}$ can be defined as a triplet $\mathcal{M}(a) =< pre(a), add(a), del(a) >$, consisting of the precondition, add effects, and delete effects of the action. After the action's execution, the subsequent state $s' = s \cup add(a) \setminus del(a)$.

Problem Formalization. The BT planning problem is a tuple: $< \mathcal{S}, \mathcal{A}, \mathcal{M}, D, s_0, g >$, where $D : \mathcal{A} \mapsto \mathbb{R}^+$ is the cost function, s_0 is the initial state, g is the goal condition. For each BT $\mathcal{T}$ includes a path $p \in \mathcal{T}, p = (a_1, a_2, ..., a_n)$ that achieves g from s_0, BT planning aims to find the optimal BT $\mathcal{T}_*$ with minimal execution cost: $\mathcal{T}_* = \arg\min_{\mathcal{T} \in \mathscr{T}} D(\mathcal{T}) = \arg\min_{\mathcal{T} \in \mathscr{T}} \sum_{i=1}^{n} D(a_i)$.

3 Related Work

3.1 Behavior Tree Planning

Current studies on automatic BT generation generally have certain limitations. Evolutionary computing [Neupane and Goodrich, 2019; Colledanchise *et al.*, 2019b; Lim *et al.*, 2010], reinforcement learning [Banerjee, 2018], imitation learning [French *et al.*, 2019], and MCTS [Scheide *et al.*, 2021] require implicit goals of BT, such as fitness functions or reward functions, which are difficult to translate from human instructions. Formal synthesis [Tadewos *et al.*, 2022; Neupane and Goodrich, 2023] requires complex environment modeling and has rather high computational complexity. BT generation directly using LLMs [Izzo *et al.*, 2024; Li *et al.*, 2024; Lykov and Tsetserukou, 2023; Lykov *et al.*, 2023] is adaptable to the diversity of human instructions but cannot guarantee the reliability of BTs.

In contrast, BT planning has explicit goal conditions, easily designed action models, and the theoretical guarantee of the success and reliability of planned BTs. The state-of-the-art BT planning methods mainly include: (1) BT Expansion [Cai *et al.*, 2021], a sound and complete algorithm for BT planning where planned BTs are guaranteed to be finite time successful. (2) OBTEA [Chen *et al.*, 2024], Optimal Behavior Tree Expansion Algorithm. It is based on BT Expansion but designed to plan optimal BTs with minimal execution cost. However, the computation complexity is high when the action space becomes large. (3) HBTP [Cai *et al.*, 2025b], Heuristic Optimal Behavior Tree Expansion Algorithm. It uses LLMs to provide action space pruning and planning heuristics through commonsense reasoning. This approach has advantages in both planning and execution efficiency, but places high demands on the task understanding and heuristics generation capabilities of LLMs.

3.2 Benchmark Environments for Robot Planning

The most common benchmarks for robot planning are designed for low-level control tasks. For path planning, there are Plannie [Rocha and Vivaldini, 2022], MRPB 1.0 [Wen *et al.*, 2021], and PathBench [Toma *et al.*, 2021]. For manipulation planning, there are MotionBenchMaker [Chamzas *et al.*, 2022], the motion planning pipeline [Liu and Liu, 2022], and the grasp planning benchmark [Bekiroglu *et al.*, 2020]. In recent years, learning methods like reinforcement learning [Bai *et al.*, 2023; Bai *et al.*, 2025b; Bai *et al.*, 2025a] have become popular approaches for robot control planning, with many benchmark environments proposed, such as Safety Gymnasium [Ji *et al.*, 2023] and RLBench [Ning *et al.*, 2023]. For high-level behavior planning tasks, existing works are mostly in the field of embodied intelligence, and LLMs are often used to generate plans [Valmeekam *et al.*, 2023; Singh *et al.*, 2022; Chen *et al.*, 2023]. However, the lack of reliability is a critical flaw when LLMs directly generate plans, not to mention the safety and robustness during the execution of these plans. BTPG fills this gap by providing a platform and benchmark for BT planning, where reliable and robust BTs for high-level behavior control can be automatically generated, without needing to access low-level control of the robots.

4 Behavior Tree Planning Gym

In this section, we first briefly introduce the object categorization that assists in modeling the BT planning problem. We then provide detailed information on the BT simulation platforms and the BT planning benchmarks. The framework of BTPG is illustrated in Figure 2.

Object Categorization. In BTPG, behavior nodes are represented using predicate logic, and action models are represented in the STRIPS style. Consequently, defining the domains of predicates and the state transitions for each action instance becomes more challenging as the number of available objects in the environment increases. Object categorization [Puig *et al.*, 2018] is a useful technique to improve the reusability of predicate and action model templates, based on the shared attributes of objects. This approach utilizes the abstraction of the world through human common sense. One inspiration from it is that conditions and actions can also be categorized, potentially leading to the development of more efficient BT planning algorithms based on this hierarchical representation in the future.

4.1 Behavior Tree Simulation Platform

The BT systems in BTPG are all built in *Python*, which can communicate with external simulators such as Unity or Unreal Engine to implement low-level robot control. We design four environments in which various BT planning tasks can be tested. These environments have different themes and problem scales. Detailed information on these environments is listed in the Appendix.

RoboWaiter (RW). In this environment, the robot acts as a waiter to perform tasks according to customer needs, such as delivering coffee or cleaning tables. The planned BTs can be executed in a simulator developed based on Unreal Engine 5 by Dataa [Chen *et al.*, 2024]. The low-level robot control for action nodes is based on scripted APIs. The task difficulty in this environment is relatively low.

VirtualHome (VH). In this environment, the robot needs to accomplish various household tasks, such as washing fruits or turning on the TV. The planned BTs can be executed in the VirtualHome [Puig *et al.*, 2018] simulator. The low-level robot control is implemented using programmed scripts in a specific format. The task difficulty in this environment is medium.

OmniGibson (OG). In this environment, the robot executes everyday household activities in IsaacSim, a more realistic simulator developed by Nvidia. The robot and object sources are from the BEHAVIOR-1K dataset distributed with OmniGibson [Li *et al.*, 2023] platform. The low-level robot control for action nodes is implemented using rule-based navigation and manipulation algorithms. The task difficulty in this environment is medium.

RobotHow (RH). This environment is inspired by the RobotHow knowledge base [Liao *et al.*, 2019] which includes a number of activity programs for household robots. This environment is designed for large-scale computational tests, and the planned BTs can be executed in a headless environment without simulation deployment for now. The task difficulty in this environment is high.

In BTPG, we propose the Behavior Tree Markup Language (BTML) to represent BTs, which is more concise and readable than XML in terms of BT representation. Details about BTML can be found in Appendix.

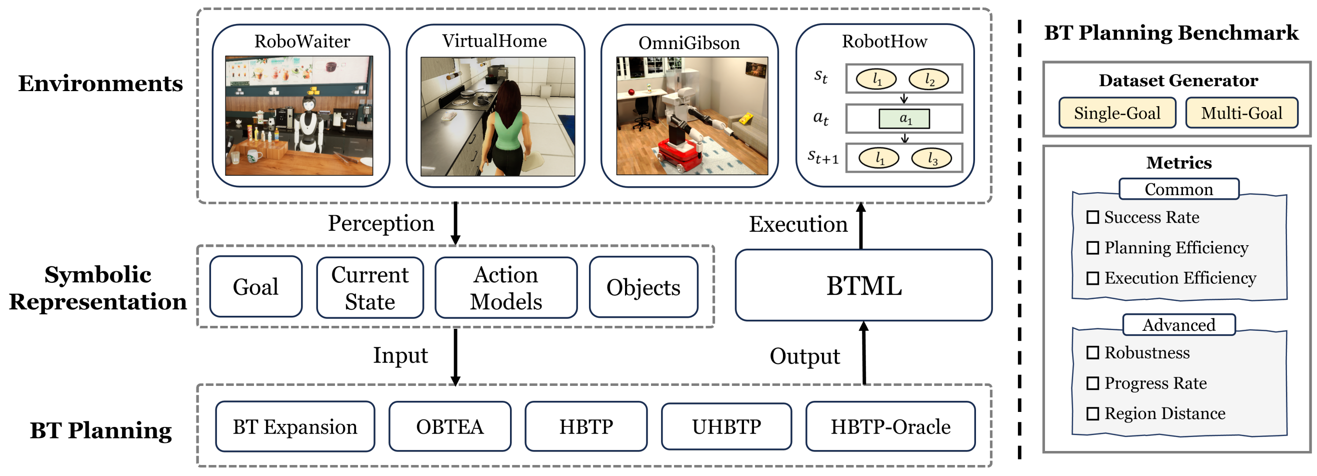

Figure 2: Overview of the BTPG architecture. We support four environments and five BT planning algorithms. The tasks are represented using symbolic representation for BT planning, and the produced BTs are converted to BTML format for execution. For benchmarking, we design a dataset generator for each environment and use common and advanced metrics to evaluate the performance of the planned BTs.

Items		RW	VH	OG	RH
Predicates	Conditions	8	11	13	13
	Actions	6	11	16	16
Objects	Categories	3	5	10	10
	Total	31	36	17	126
Instances	Conditions	306	293	122	4914
	Actions	174	480	155	7329

Table 1: The detailed problem scales of four environments.

4.2 Behavior Tree Planning Benchmark

Universal Heuristic Behavior Tree Planning

We implement all three state-of-the-art BT planning algorithms in BTPG as mentioned in Section 3.1. Additionally, we further propose a new BT planning algorithm inspired by classical planning. We notice that current heuristic BT planning algorithms [Cai *et al.*, 2025b] mainly use domain-dependent heuristics produced by LLMs. However, domain-independent heuristics have been widely studied in the field of classical planning [Hoffmann and Nebel, 2001] but are lacking in BT planning algorithms. In this paper, we simply replace the heuristics used in HBTP with domain-independent heuristics and propose Universal Heuristic Behavior Tree Planning (UHBTP).

We start with the delete relaxation used in GRAPHPLAN [Hoffmann and Nebel, 2001]. Consider a relaxed problem where, for each action, $del(a) = \emptyset$. In this case, we start from the initial state and apply all actions whose preconditions are satisfied in the current state at each step, resulting in a sequence of literal sets $\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_n$, where $\mathcal{L}_1 = s_0$ and $g \subseteq \mathcal{L}_n$. We let $\mathcal{L}_{n+1} = \mathcal{L}$, then we subtract these literal sets layer-by-layer to form heuristic plan layers: $\mathcal{L}'_1 = \mathcal{L}_1, \mathcal{L}'_{i+1} = \mathcal{L}_{i+1} \setminus \mathcal{L}_i, i = 1, 2, ..., n$. For these heuristic plan layers, we have:

$$\bigcap_{i=1}^{n+1} \mathcal{L}'_i = \emptyset, \quad \bigcup_{i=1}^{n+1} \mathcal{L}'_i = \mathcal{L} \tag{1}$$

The heuristic for each literal can be defined as:

$$h(l) \doteq i, l \in \mathcal{L}'_i \tag{2}$$

The heuristic for conditions is the sum of their literals:

$$h(c) \doteq \sum_{l \in c} h(l) \tag{3}$$

Although the above heuristics are simple, they are efficient in guiding BT planning by prioritizing the exploration of potentially shorter action paths, thereby accelerating the planning process. The comparison results of the performance of UHBTP with other existing BT planning algorithms are shown in the Experiment section.

Dataset Generator

Given an action model and human prior knowledge, planning tasks within the environments can be automatically generated. In BTPG, we designed a dataset generator for each environment to randomly generate batches of tasks to test the performance of planning algorithms. Typically, the dataset generator first randomly generates the environment state and goal conditions according to certain distributions to form the planning task. These distributions can be set to uniform distributions if there are no special requirements. Then, the optimal path is obtained using OBTEA as a reference for evaluating other planning algorithms. The final task goals can include a single goal condition or a combination of multiple goals, with the number of goal conditions controlling the difficulty of the planning tasks. Details of the dataset generator and the data format are provided in the Appendix.

Metrics Design

Common Metrics. Our benchmark includes the following common evaluation metrics to evaluate the performance of

Algorithms	Planning Time(ms)	Timeout Rate	Actions	Costs	Tree Sizes	Ticks
Environment: RoboWaiter						
BT Expansion	18.0	0.8%	1.9	17.1	50.6	183.5
OBTEA	0.6	0.0%	2.0	17.8	5.6	49.1
HBTP	0.6	0.0%	2.0	17.8	5.2	50.0
UHBTP	0.7	0.0%	2.0	18.0	8.7	60.4
HBTP-Oracle	0.5	0.0%	2.0	17.8	4.2	43.1
Environment: VirtualHome						
BT Expansion	406.5	33.5%	2.9	31.7	1582.1	797.4
OBTEA	92.3	1.5%	3.8	39.6	191.0	2245.2
HBTP	70.4	2.0%	3.9	40.7	89.8	799.9
UHBTP	18.9	0.5%	4.0	44.0	154.3	406.6
HBTP-Oracle	7.5	0.0%	4.0	43.3	7.2	130.3
Environment: OmniGibson						
BT Expansion	371.4	27.0%	3.6	39.7	719.0	2203.6
OBTEA	48.3	0.0%	4.4	46.5	137.9	2168.1
HBTP	28.7	0.0%	4.6	49.4	71.3	917.6
UHBTP	22.2	0.0%	4.5	49.4	81.7	334.5
HBTP-Oracle	4.6	0.0%	4.6	51.6	10.3	199.5
Environment: RobotHow						
BT Expansion	729.7	68.5%	2.3	26.7	3515.3	714.4
OBTEA	505.7	47.0%	2.9	31.4	153.4	114.8
HBTP	469.4	39.0%	3.6	39.3	63.4	110.7
UHBTP	381.3	22.0%	4.3	46.4	3078.8	6261.7
HBTP-Oracle	336.6	18.0%	4.4	49.9	13.5	169.6

Table 2: The comparison of BT planning algorithms in common metrics. The results are averaged over 5 seeds.

BTs: (1) Planning efficiency, including planning time and timeout rate; (2) Execution efficiency, including executed actions, costs, tree sizes, and the number of ticks during execution.

In addition to the above common metrics, we propose the following three advanced metrics in this paper. These metrics aim to provide deeper insights into BT planning algorithms and enable a more comprehensive evaluation.

Planning Progress (PP). For the planning efficiency of BT planning algorithms, simple metrics like planning time and the timeout rate may be insufficient. It would be helpful if we could understand the specific advance of BT planning at each exploration and estimate the overall progress of the whole planning procedure. Therefore, we propose the Planning Progress metric. This metric is based on the optimal paths provided in the datasets. Assuming we have an optimal path set $\mathcal{P}^*$ for a given task, the planning progress at time t can be calculated as follows:

$$PP_t = \max_{p^* \in \mathcal{P}^*} \max_{p_t \in \mathcal{T}_t} \frac{\sum_{a \in \mathcal{A}(p^*)} \max(I(p_t, a), I(p^*, a))}{\sum_{a \in \mathcal{A}(p^*)} I(p^*, a)} \quad (4)$$

where $\mathcal{T}_t$ is the BT planned at the current time, $\mathcal{A}(p^*)$ is the set of actions shown in path p^*, $I(p, a)$ is the action count that shows how many times action a appears in the path p.

With this metric, we can estimate the ability of the planning algorithm to finish the task before it completes the search. This is especially helpful when the problem scale is large and the algorithm cannot finish within the expected time limit. However, since it is impossible to maintain all optimal paths in the datasets, this estimation could be inaccurate when multiple optimal paths are available.

Region Distance (RD). It is also helpful to know the distribution of path lengths starting from each condition within the attraction region of the planned BT, which we refer to as Region Distance. This metric provides two main insights: it allows us to understand the depths on which the search process primarily focuses, and it gives us a rough idea of which tasks the final planned BT can accomplish. Given an interval $[\alpha_1, \alpha_2)$, we can count the number of conditions whose corresponding path lengths fall within it:

$$RD_{[\alpha_1, \alpha_2)} = |\{\alpha_1 \le |p(\mathcal{T}, c)| < \alpha_2 | c \in \mathcal{C}(\mathcal{T})\}| \quad (5)$$

where $p(\mathcal{T}, c)$ is the path from condition c to goal g within the BT $\mathcal{T}$, and $\mathcal{C}(\mathcal{T})$ is the set of conditions in the attraction region of $\mathcal{T}$.

With the same number of explorations, shorter path lengths within the attraction region indicate that the BT primarily explores the region near the goal, making it more likely to fail in long-horizontal tasks. Therefore, we generally expect a larger Region Distance for the planned BTs.

Execution Robustness (ER). Robustness is one of the most important advantages that distinguish BTs from other control architectures. However, it is rarely addressed in current BT planning literature. In this paper, we test the robustness of planned BTs by introducing noise into the environment. We first define a noise function $\mathcal{N}$ with a noise parameter $\epsilon \in [0, 1]$, which indicates the probability of each condition in the environment being disturbed. We set up disturbance functions for each environment to ensure the disturbances are reasonable and reflect real-world conditions. We have the initial noise parameter ϵ_1 and the transition noise parameter ϵ_2, which respectively test the BT's ability to execute tasks under conditions not targeted in the planning and its robustness under uncertainties on the environment transition. Under the effect of the noise function, the environment transitions as follows:

$$\begin{aligned} &s'_0 = \mathcal{N}(s_0; \epsilon_1), a_{i+1} = \mathcal{T}(s'_i), \\ &s_{i+1} = \mathcal{M}(s'_i, a_{i+1}), s'_{i+1} = \mathcal{N}(s_{i+1}; \epsilon_2), \\ &i = 0, 1, 2, 3, \cdots \end{aligned}$$

We measure the execution robustness of a BT by the probability that it successfully reaches the goal in a noisy environment. We assume that the planned BTs succeed in a finite time in the absence of noise, meaning that when the environment state falls in the attraction region of the BT, it can follow the predetermined path to the goal. Thus, as long as the BT has not returned a `failure`, it is considered to be progressing towards the goal. Execution Robustness can then be defined as follows:

$$ER = 1 - \prod_{i=0}^{\infty} P(\texttt{failure} = \mathcal{T}(s'_i) | s_0, \epsilon_1, \epsilon_2, \mathcal{N}, \mathcal{T}, \mathcal{M}) \quad (6)$$

where $\texttt{failure} = \mathcal{T}(s'_i)$ indicates that the BT $\mathcal{T}$ returns a `failure` under state s'_i at time t.

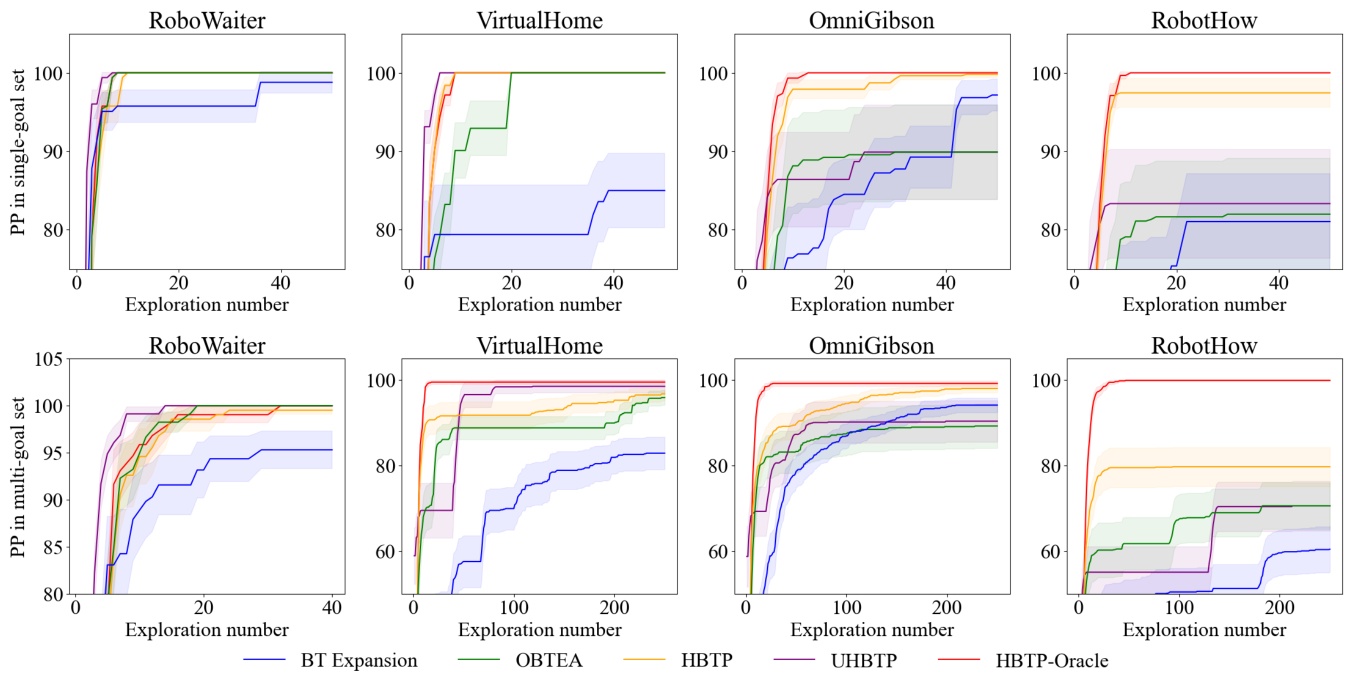

Figure 3: The changes in the Planning Progress metric with the increase in the number of explorations.

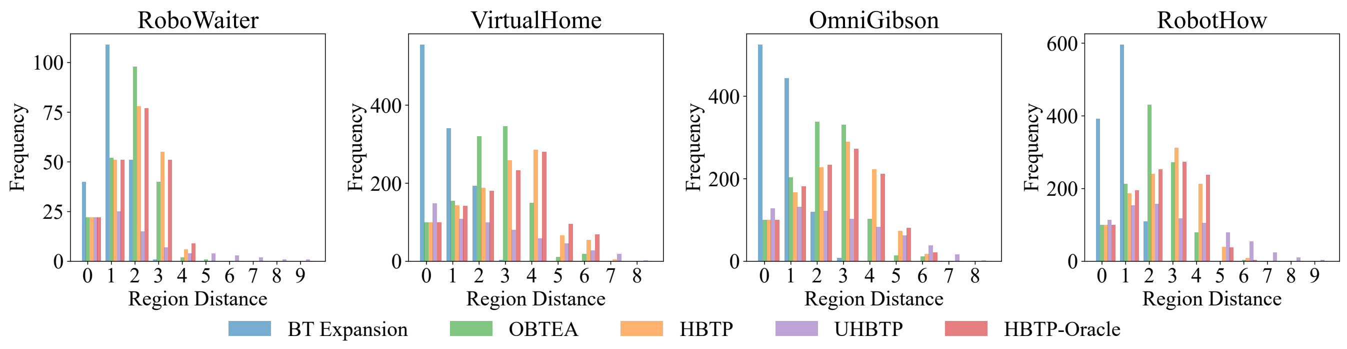

Figure 4: The comparison of Region Distance in the multi-goal set.

Achieving greater robustness in the planned BTs typically involves higher planning and execution expenses. Therefore, the trade-off between efficiency and robustness is a key challenge that future BT planning algorithms need to address.

5 Experiments

To evaluate the performance of BT planning algorithms in our benchmark, for each environment, we randomly generated 100 single-goal tasks under a uniform distribution, as well as 100 multi-goal tasks where the number of goal conditions ranged from 2 to 3. We implemented the following five BT planning algorithms in BTPG: BT Expansion, OBTEA, HBTP, UHBTP and HBTP-Oracle, where in HBTP-Oracle we assume that the heuristics provided by LLMs are always accurate. All our experiments were performed on an AMD Ryzen 9 5900X 12-Core Processor (3.70 GHz).

5.1 Common Metrics

Planning Efficiency. We set a planning time limit of 1 second for each algorithm. If the time limit is exceeded, planning is stopped and its execution efficiency is not recorded. As we can see from Table 2, BT Expansion has the lowest planning efficiency while HBTP-Oracle has the highest. The heuristics used by the HBTP algorithm significantly improved planning efficiency compared to OBTEA, but there is still a considerable gap compared to HBTP-Oracle. More surprisingly, even UHBTP, which uses only domain-independent heuristics is faster than HBTP. This indicates that there is still much room for improving the reasoning abilities of LLMs to enhance the BT planning efficiency.

Execution Efficiency. As shown in Table 2, when the timeout rate is close to 0, we can observe that algorithms using heuristics generally have higher executed actions and costs than OBTEA. However, compared to the significant improve-

Settings	Algorithms	Single-Goal				Multi-Goal			
		RW	VH	OG	RH	RW	VH	OG	RH
$\epsilon_1 = 0$ $\epsilon_2 = 0.2$	BT Expansion	81	74	78	55	17	27	26	7
	OBTEA	82	81	72	62	14	48	53	10
	HBTP	77	76	74	69	12	58	51	23
	UHBTP	82	72	72	64	16	52	41	23
	HBTP-Oracle	83	77	80	69	13	51	51	13
$\epsilon_1 = 0$ $\epsilon_2 = 0.5$	BT Expansion	55	58	63	48	8	20	19	7
	OBTEA	57	60	59	46	6	31	34	9
	HBTP	54	58	56	49	5	24	26	16
	UHBTP	57	57	56	45	7	22	21	12
	HBTP-Oracle	53	57	61	47	4	24	31	11
$\epsilon_1 = 1$ $\epsilon_2 = 0$	BT Expansion	34	71	68	60	1	40	46	16
	OBTEA	34	70	67	62	1	56	60	19
	HBTP	33	68	69	65	0	46	47	36
	UHBTP	35	65	67	59	1	53	46	33
	HBTP-Oracle	35	68	88	64	1	48	52	23
$\epsilon_1 = 1$ $\epsilon_2 = 0.2$	BT Expansion	33	65	67	50	0	36	37	18
	OBTEA	32	66	65	54	1	53	49	18
	HBTP	33	61	65	55	0	37	38	29
	UHBTP	37	61	64	56	2	37	32	22
	HBTP-Oracle	33	66	64	55	0	35	38	19

Table 3: The success rates (%) of planned BT in noisy environments, which reflect the Execution Robustness metric.

ment in planning speed they achieve, these minor sacrifices in execution efficiency are acceptable. Moreover, heuristics can lead to smaller tree sizes and lower tick counts, indicating that the generated tree structures are more concise. However, when the timeout rate increases, the assessment results for Execution Efficiency will lose statistical significance, as many tasks fail to generate an executable BT.

5.2 Advanced Metrics

Evaluation of Planning Progress. As shown in Figure 3, we can observe a clear phenomenon where the PP in the early stage is generally positively correlated with the final planning efficiency. This indicates that our PP metric can effectively reflect the planning efficiency of an algorithm early in the planning procedure, without waiting for the entire planning to complete. This is particularly helpful in complex environments, such as RH. Furthermore, most of the planning progresses are achieved early on, and as planning continues, the increase in PP slows down significantly. This indicates that the difficulty of planning in the later stages is much higher than in the early stages. This is because the number of combinations between conditions increases dramatically as more conditions are explored. This insight suggests that improving planning efficiency in the later stages of planning might be a worthwhile research direction. Another phenomenon regarding the planning efficiency of OBTEA and BT Expansion has been observed. In the OG environment, the planning progress of BT Expansion eventually surpasses that of OBTEA, while in other environments, it is generally lower. This suggests that the planning efficiency of the algorithms may depend on the specific action model of the environment.

Evaluation of Region Distance. We conducted Region Distance experiments on the multi-goal dataset, where each algorithm was run for only 10 explorations. Despite the limited number of explorations, we can still observe differences between the algorithms. As shown in Figure 4, the Region Distance metric indicates that BT Expansion's Region Distance is generally distributed within a smaller range. The Region Distance for HBTP and UHBTP is relatively larger, which suggests that with the help of heuristics, the algorithms tend to explore states in more distant regions. Additionally, the performance of HBTP is not significantly different from its theoretical bound, HBTP-Oracle. From this experiment, we can preliminarily conclude that higher planning efficiency often implies that the algorithm explores regions at sufficiently far distances.

Evaluation of Execution Robustness. To test the robustness of BTs generated by different algorithms, we set various noise parameter settings in each environment, ran each algorithm with 5 random seeds, and calculated the average completion rate for the BTs generated by each algorithm. As shown in Table 3, when only the environment transition noise ϵ_2 is present, these BTs have a relatively high success rate on the single-goal set. However, the success rate significantly drops on the multi-goal set. This is because the robot needs to complete multiple goals simultaneously, and noise interference in any single goal can lead to the failure of the entire task. Next, when comparing the second and third settings, we found that in VH, OG, and RH, the success rate with $\epsilon_1 = 1, \epsilon_2 = 0$ is higher than with $\epsilon_1 = 0, \epsilon_2 = 0.5$. However, in the RW environment, the opposite is true. This indicates that the robustness of BTs generated by planning algorithms is depends on the type of tasks in different environments. Finally, we observed that in all random perturbation environments, the robustness of the BTs generated by all algorithms is generally low. This highlights the severe deficiencies of current BT planning algorithms in terms of the robustness of the generated BTs. Improving the robustness of planned BTs is a significant challenge that needs to be addressed in future work.

6 Conclusion

This paper presents BTPG, the first platform and benchmark for Behavior Tree (BT) planning in everyday service robots. In BTPG, we adopt a predicate logic representation for behavior nodes, formulate the BT planning problem using a STRIPS-style action model, and use object categorization to aid definition. We create four environments with three simulators that encompass most everyday service activities. Additionally, we design a dataset generator for each environment. We implement and test three state-of-the-art BT planning algorithms, as well as UHBTP proposed by us, using various common metrics and three advanced metricsto gain deeper insights into these algorithms. With a standard test benchmark, we hope BTPG can inspire and accelerate progress in the field of BT planning. Future work includes continuing to improve the success rate of BT execution in the simulators, incorporating more BT planning algorithms, and providing a more comprehensive user manual and community support.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant Nos. 62032024 and 62372459) and the Independent Innovation Science Fund Project of NUDT (Grant No. 23-ZZCXKXKY-07).

Contribution Statement

Yishuai Cai and Minglong Li are corresponding authors.

References

[Bai *et al.*, 2023] Fengshuo Bai, Hongming Zhang, Tianyang Tao, Zhiheng Wu, Yanna Wang, and Bo Xu. Picor: Multi-task deep reinforcement learning with policy correction. *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 37(6), Jun. 2023.

[Bai *et al.*, 2025a] Fengshuo Bai, Yu Li, Jie Chu, Tawei Chou, Runchuan Zhu, Ying Wen, Yaodong Yang, and Yuanpei Chen. Retrieval dexterity: Efficient object retrieval in clutters with dexterous hand. *arXiv preprint arXiv:2502.18423*, 2025.

[Bai *et al.*, 2025b] Fengshuo Bai, Runze Liu, Yali Du, Ying Wen, and Yaodong Yang. Rat: Adversarial attacks on deep reinforcement agents for targeted behaviors. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 39, 2025.

[Banerjee, 2018] Bikramjit Banerjee. Autonomous Acquisition of Behavior Trees for Robot Control. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3460–3467, 2018.

[Bekiroglu *et al.*, 2020] Yasemin Bekiroglu, Naresh Marturi, Máximo A. Roa, Komlan Jean Maxime Adjigble, Tommaso Pardi, Cindy Grimm, Ravi Balasubramanian, Kaiyu Hang, and Rustam Stolkin. Benchmarking Protocol for Grasp Planning Algorithms. *IEEE Robotics and Automation Letters*, 5(2):315–322, 2020.

[Cai *et al.*, 2021] Zhongxuan Cai, Minglong Li, Wanrong Huang, and Wenjing Yang. BT Expansion: A Sound and Complete Algorithm for Behavior Planning of Intelligent Robots with Behavior Trees. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 6058–6065. AAAI Press, 2021.

[Cai *et al.*, 2025a] Yishuai Cai, Xinglin Chen, Zhongxuan Cai, Yunxin Mao, Minglong Li, Wenjing Yang, and Ji Wang. Mrbtp: Efficient multi-robot behavior tree planning and collaboration. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 14548–14557, 2025.

[Cai *et al.*, 2025b] Yishuai Cai, Xinglin Chen, Yunxin Mao, Minglong Li, Shaowu Yang, Wenjing Yang, and Ji Wang. Hbtp: Heuristic behavior tree planning with large language model reasoning. *Proceedings of the International Conference on Robotics and Automation (ICRA)*, 2025.

[Chamzas *et al.*, 2022] Constantinos Chamzas, Carlos Quintero-Peña, Zachary Kingston, Andreas Orthey, Daniel Rakita, Michael Gleicher, Marc Toussaint, and Lydia E. Kavraki. MotionBenchMaker: A Tool to Generate and Benchmark Motion Planning Datasets. *IEEE Robotics and Automation Letters*, 7(2):882–889, 2022.

[Chen *et al.*, 2023] Yaran Chen, Wenbo Cui, Yuanwen Chen, Mining Tan, Xinyao Zhang, Dongbin Zhao, and He Wang. RoboGPT: An intelligent agent of making embodied long-term decisions for daily instruction tasks. *arXiv preprint arXiv:2311.15649*, 2023.

[Chen *et al.*, 2024] Xinglin Chen, Yishuai Cai, Yunxin Mao, Minglong Li, Wenjing Yang, Weixia Xu, and Ji Wang. Integrating Intent Understanding and Optimal Behavior Planning for Behavior Tree Generation from Human Instructions. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2024.

[Colledanchise and Ögren, 2018] Michele Colledanchise and Petter Ögren. *Behavior Trees in Robotics and AI: An Introduction*. CRC Press, 2018.

[Colledanchise *et al.*, 2019a] Michele Colledanchise, Diogo Almeida, and Petter Ogren. Towards Blended Reactive Planning and Acting using Behavior Trees. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pages 8839–8845, Montreal, QC, Canada, May 2019. IEEE.

[Colledanchise *et al.*, 2019b] Michele Colledanchise, Ramviyas Parasuraman, and Petter Ögren. Learning of Behavior Trees for Autonomous Agents. *IEEE Transactions on Games*, 11(2):183–189, 2019.

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

[French *et al.*, 2019] Kevin French, Shiyu Wu, Tianyang Pan, Zheming Zhou, and Odest Chadwicke Jenkins. Learning behavior trees from demonstration. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 7791–7797. IEEE, 2019.

[Hoffmann and Nebel, 2001] Jörg Hoffmann and Bernhard Nebel. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14:253–302, 2001.

[Izzo *et al.*, 2024] Riccardo Andrea Izzo, Gianluca Bardaro, and Matteo Matteucci. Btgenbot: Behavior tree generation for robotic tasks with lightweight llms. *arXiv preprint arXiv:2403.12761*, 2024.

[Ji *et al.*, 2023] Jiaming Ji, Borong Zhang, Jiayi Zhou, Xuehai Pan, Weidong Huang, Ruiyang Sun, Yiran Geng, Yifan Zhong, Josef Dai, and Yaodong Yang. Safety Gymnasium: A Unified Safe Reinforcement Learning Benchmark. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS)*, volume 36, pages 18964–18993. Curran Associates, Inc., 2023.

[Li *et al.*, 2023] Chengshu Li, Ruohan Zhang, Josiah Wong, Cem Gokmen, Sanjana Srivastava, Roberto Martín-

Martín, Chen Wang, Gabrael Levine, Michael Lingelbach, Jiankai Sun, Mona Anvari, Minjune Hwang, Manasi Sharma, Arman Aydin, Dhruva Bansal, Samuel Hunter, Kyu-Young Kim, Alan Lou, Caleb R Matthews, Ivan Villa-Renteria, Jerry Huayang Tang, Claire Tang, Fei Xia, Silvio Savarese, Hyowon Gweon, Karen Liu, Jiajun Wu, and Li Fei-Fei. BEHAVIOR-1K: A Benchmark for Embodied AI with 1,000 Everyday Activities and Realistic Simulation. In Karen Liu, Dana Kulic, and Jeff Ichnowski, editors, *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 80–93. PMLR, December 2023.

[Li *et al.*, 2024] Fu Li, Xueying Wang, Bin Li, Yunlong Wu, Yanzhen Wang, and Xiaodong Yi. A Study on Training and Developing Large Language Models for Behavior Tree Generation. *arXiv preprint arXiv:2401.08089*, 2024.

[Liao *et al.*, 2019] Yuan-Hong Liao, Xavier Puig, Marko Boben, Antonio Torralba, and Sanja Fidler. Synthesizing environment-aware activities via activity sketches. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[Lim *et al.*, 2010] Chong-U Lim, Robin Baumgarten, and Simon Colton. Evolving behaviour trees for the commercial game DEFCON. In *Applications of Evolutionary Computation: EvoApplicatons 2010: EvoCOMPLEX, EvoGAMES, EvoIASP, EvoINTELLIGENCE, EvoNUM, and EvoSTOC, Istanbul, Turkey, April 7-9, 2010, Proceedings, Part I*, pages 100–110. Springer, 2010.

[Liu and Liu, 2022] Shuai Liu and Pengcheng Liu. Benchmarking and optimization of robot motion planning with motion planning pipeline. *The International Journal of Advanced Manufacturing Technology*, 118(3):949–961, January 2022.

[Lykov and Tsetserukou, 2023] Artem Lykov and Dzmitry Tsetserukou. LLM-BRAIn: AI-driven Fast Generation of Robot Behaviour Tree based on Large Language Model. *arXiv preprint arXiv:2305.19352*, 2023.

[Lykov *et al.*, 2023] Artem Lykov, Maria Dronova, Nikolay Naglov, Mikhail Litvinov, Sergei Satsevich, Artem Bazhenov, Vladimir Berman, Aleksei Shcherbak, and Dzmitry Tsetserukou. LLM-MARS: Large Language Model for Behavior Tree Generation and NLP-enhanced Dialogue in Multi-Agent Robot Systems. *arXiv preprint arXiv:2312.09348*, 2023.

[Neupane and Goodrich, 2019] Aadesh Neupane and Michael Goodrich. Learning Swarm Behaviors using Grammatical Evolution and Behavior Trees. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 513–520, 2019.

[Neupane and Goodrich, 2023] Aadesh Neupane and Michael A Goodrich. Designing Behavior Trees from Goal-Oriented LTLf Formulas. *arXiv preprint arXiv:2307.06399*, 2023.

[Ning *et al.*, 2023] Kun-Peng Ning, Hu Xu, Kun Zhu, and Sheng-Jun Huang. Co-Imitation Learning without Expert Demonstration. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2023.

[Ögren and Sprague, 2022] Petter Ögren and Christopher I Sprague. Behavior trees in robot control systems. *Annual Review of Control, Robotics, and Autonomous Systems*, 5:81–107, 2022.

[Puig *et al.*, 2018] Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. Virtualhome: Simulating household activities via programs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8494–8502, 2018.

[Rocha and Vivaldini, 2022] Lidia Rocha and Kelen Vivaldini. Plannie: A Benchmark Framework for Autonomous Robots Path Planning Algorithms Integrated to Simulated and Real Environments. In *International Conference on Unmanned Aircraft Systems (ICUAS)*, 2022.

[Scheide *et al.*, 2021] Emily Scheide, Graeme Best, and Geoffrey A. Hollinger. Behavior Tree Learning for Robotic Task Planning through Monte Carlo DAG Search over a Formal Grammar. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pages 4837–4843, Xi'an, China, May 2021. IEEE.

[Singh *et al.*, 2022] Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. ProgPrompt: Generating Situated Robot Task Plans using Large Language Models. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*. Proceedings of the International Conference on Robotics and Automation (ICRA), September 2022.

[Tadewos *et al.*, 2022] Tadewos G Tadewos, Abdullah Al Redwan Newaz, and Ali Karimoddini. Specification-guided behavior tree synthesis and execution for coordination of autonomous systems. *Expert Systems with Applications*, 201:117022, 2022.

[Toma *et al.*, 2021] Alexandru-Iosif Toma, Hao-Ya Hsueh, Hussein Ali Jaafar, Riku Murai, Paul H.J. Kelly, and Sajad Saeedi. PathBench: A Benchmarking Platform for Classical and Learned Path Planning Algorithms. In *2021 18th Conference on Robots and Vision (CRV)*, pages 79–86, 2021.

[Valmeekam *et al.*, 2023] Karthik Valmeekam, Matthew Marquez, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. Planbench: An extensible benchmark for evaluating large language models on planning and reasoning about change. *Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS)*, 36, 2023.

[Wen *et al.*, 2021] Jian Wen, Xuebo Zhang, Qingchen Bi, Zhangchao Pan, Yanghe Feng, Jing Yuan, and Yongchun Fang. Mrpb 1.0: A unified benchmark for the evaluation of mobile robot local planning approaches. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pages 8238–8244. IEEE, 2021.