A Limitations

We discuss the limitations of BTPG from the following aspects:

Node Representation. The current representation using first-order predicates is simple and intuitive, but it has significant limitations. Firstly, when there are many objects with the same name in the scenario, all these objects need to be added to the object library, which greatly increases the problem's scale. Additionally, this method does not effectively handle the dynamic changes in action costs; currently, the cost of a node can only be preset to a fixed value. Lastly, in our current work, only objects are categorized. In fact, condition nodes and action nodes can also be abstracted and categorized, which could potentially greatly enhance the efficiency of planning algorithms.

STRIPS-style Action Model. The popularity of STRIPS-style planning in BTs is due to its decomposition of the effects of actions into add and delete sets, rather than a single postcondition. This approach simplifies backward planning from the goal, aligning well with the robustness required in BTs, which continuously explore paths to the goal. However, this representation also has its limitations, such as difficulty in representing the quantities of objects. Therefore, in the future, it might be beneficial to consider adopting more advanced world models to formalize the BT planning problem.

Scenarios. Although we aim for the proposed four scenarios to comprehensively cover the activities of everyday service robots, there are undoubtedly areas that are lacking. In the future, we can also incorporate more simulators and design more complex scenarios to meet the needs of BT planning for robots in various fields.

B Details about the Platform

B.1 Scenarios

B.2 BTML

In most existing literature, XML format is commonly used to store BTs. However, this format was not specifically designed to represent BTs, resulting in considerable redundancy and making it less readable and adjustable for humans. In BTPG, we designed a BT representation format similar to Python syntax, called BT Markup Language (BTML). This format uses indentation to indicate the hierarchical relationships between BT nodes and is built to recognize sequence, fallback (selector), action, and condition nodes. It offers high readability and modifiability and can easily be converted to and from the existing XML format. For example, a BT can be represented in BTML as follows:

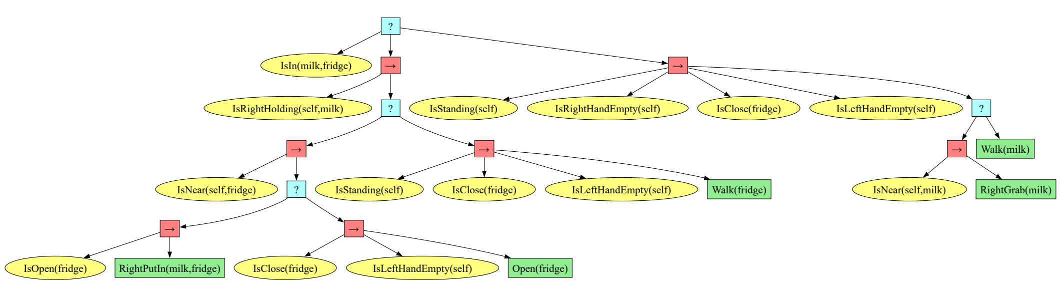


Figure 1: An example of BT, the robot need to put the milk into the refrigerator.

BTML

```
selector
  cond IsIn(milk,fridge)
  sequence
    cond IsRightHolding(self,milk)
    selector
      sequence
        cond IsNear(self,fridge)
        selector
          sequence
            cond IsOpen(fridge)
            act RightPutIn(milk,fridge)
          sequence
            cond IsClose(fridge)
            cond IsLeftHandEmpty(self)
            act Open(fridge)
      sequence
        cond IsStanding(self)
        cond IsClose(fridge)
        cond IsLeftHandEmpty(self)
        act Walk(fridge)
  sequence
    cond IsStanding(self)
    cond IsRightHandEmpty(self)
    cond IsClose(fridge)
    cond IsLeftHandEmpty(self)
    selector
      sequence
        cond IsNear(self,milk)
        act RightGrab(milk)
      act Walk(milk)
```

C Details about the Benchmark

C.1 The Dataset Generator

In the process of constructing the dataset, we adopted a systematic approach that involved creating a comprehensive dataset, where each data record is primarily composed of three parts: *Environment State*, *Goal Conditions*, and *Optimal Paths*. Initially, we established a simulated home environment and designed five distinct environmental scenarios to mimic various locations and states of objects.

We then standardized the description format of goals by defining a variety of object state descriptions and combining them with a specific set of objects (e.g., IsOn(<GRABBABLE>,<SURFACES>), IsOpen(<CAN_OPEN>), etc.) to randomly generate goal descriptions that conform to grammatical rules. During the goal composition phase, we assembled the goals into *Goal Conditions* containing one to six goals based on a probability distribution, while ensuring logical consistency, such as the requirement that a goal set containing IsSwitchedOn must also include the corresponding IsPlugged state.

Finally, we randomly selected an environmental scenario as *Environment State* and utilized OBTEA to generate a corresponding action sequence, which is described in the dataset as *Optimal Paths*, thereby forming a complete data record. By repeating this process multiple times, we successfully constructed a dataset with rich and diverse data.

Dataset Example

Environment State:

IsUnplugged(wallphone), IsUnplugged(coffeemaker), IsUnplugged(lightswitch), IsUnplugged(cellphone), IsUnplugged(fridge), IsUnplugged(toaster), IsUnplugged(tablelamp), IsUnplugged(microwave), IsUnplugged(tv), IsUnplugged(clock), IsUnplugged(radio), IsUnplugged(washingmachine), IsUnplugged(mouse), IsUnplugged(keyboard), IsUnplugged(printer), IsSwitchedOff(coffeemaker), IsSwitchedOff(cellphone), IsSwitchedOff(candle), IsSwitchedOff(faucet), IsSwitchedOff(washingmachine), IsSwitchedOff(printer), IsSwitchedOff(wallphone), IsSwitchedOff(remotecontrol), IsSwitchedOff(computer), IsSwitchedOff(toaster), IsSwitchedOff(microwave), IsSwitchedOff(dishwasher), IsSwitchedOff(clock), IsSwitchedOff(radio), IsSwitchedOff(lightswitch), IsSwitchedOff(fridge), IsSwitchedOff(tablelamp), IsSwitchedOff(stove), IsSwitchedOff(tv), IsClose(coffeemaker), IsClose(cookingpot), IsClose(toothpaste), IsClose(coffeepot), IsClose(kitchencabinet), IsClose(washingmachine), IsClose(window), IsClose(printer), IsClose(curtains), IsClose(closet), IsClose(box), IsClose(microwave), IsClose(hairproduct), IsClose(dishwasher), IsClose(radio), IsClose(fridge), IsClose(toilet), IsClose(book), IsClose(garbagecan), IsClose(magazine), IsClose(nightstand), IsClose(cabinet), IsClose(milk), IsClose(desk), IsClose(stove), IsClose(door), IsClose(folder), IsClose(clothespile), IsClose(bathroomcabinet)

Goal Conditions:

IsIn(breadslice,kitchencabinet) The bread slice is inside the kitchen cabinet
IsClose(kitchencabinet) The kitchen cabinet is closed

Optimal Paths:

Walk(breadslice) Navigate to the location of the bread slice
RightGrab(breadslice) Use the right hand to grasp the bread slice
Walk(kitchencabinet) Navigate to the kitchen cabinet
Open(kitchencabinet) Open the kitchen cabinet
RightPutIn(breadslice,kitchencabinet) Use the right hand to place the bread slice inside the kitchen cabinet
Close(kitchencabinet) Close the kitchen cabinet

This dataset example showcases a simulated household environment (*Environment State*) in which a series of actions (*Optimal Paths*) are generated and executed based on given goals (*Goal Conditions*). In this instance, the goals require placing a bread slice inside a kitchen cabinet and ensuring that the cabinet door is closed upon completion. The optimal action sequence details each step required from the initial state to the goal state. Additionally, the vital action predicates and objects that are crucial for understanding and executing the actions are also listed.

C.2 The Task Distribution

Figure 2 provides a detailed depiction of the frequency distribution of each component for a single target task within the dataset. By analyzing these data, we can clearly observe that the complexity of single-goal tasks is relatively low, as all tasks can be completed in 6 actions or fewer. It is worth noting that, in the Predicates, `Walk` appears most frequently, which fully indicates that the movement of objects is an extremely common and repetitive action during robot task execution, which is highly consistent with our daily understanding of robot behavior. Furthermore, from the distribution of Objects, objects of the `GRABBLE` appear most frequently, indicating that robots face more handling tasks in this context. In the specific context of the VirtualHome environment, `HAS_SWITCH` appears most frequently, which is likely due to the complexity and diversity of electrical operations, which typically require specific interaction and switching by robots.

Figure 3 presents in detail the frequency distribution of each component in multi-goal tasks. From the graph, it can be observed that compared to single target tasks, multi-goal tasks usually require more than 7 actions to complete, highlighting the complexity and challenge of multitasking processing. Although the frequency of `Walk` still dominates in multi-goal tasks, its advantage is no longer as significant compared to other *Predicates*. This may be due to the diversity of tasks, which requires

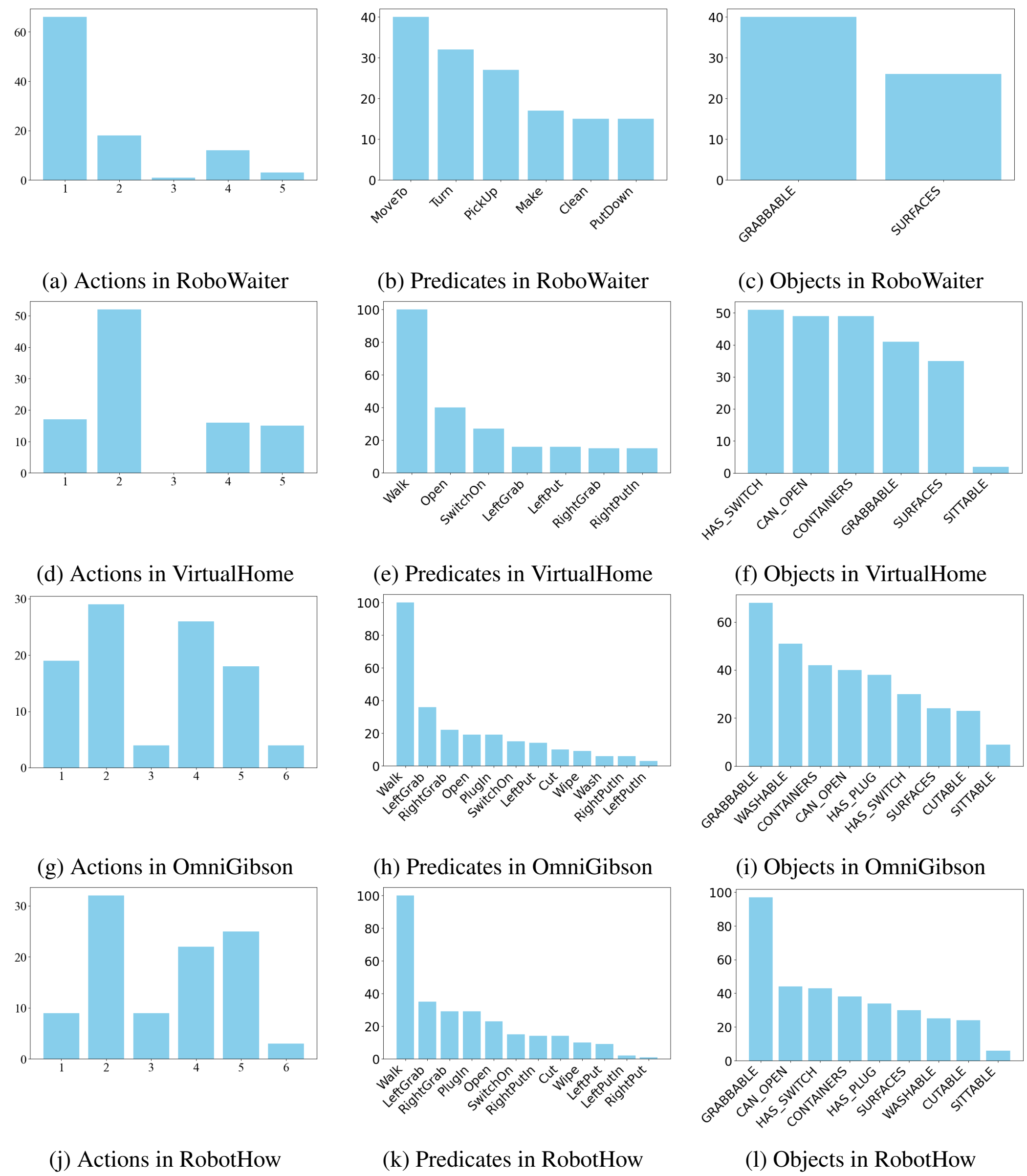
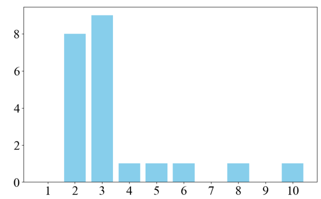


Figure 2: The distribution of single-goal tasks in different scenarios.

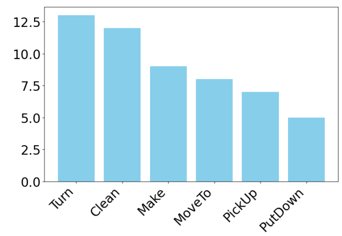
robots to stay in the same location for more work while performing different tasks, thereby reducing the need for frequent movement.

C.3 Details about Common Metrics

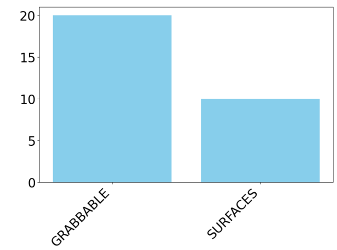
We utilized a dataset generator to create 100 instances each of single-goal and multi-goal datasets across 4 scenarios. We assessed 5 baseline algorithms on these datasets using common metrics such as success rate, planning efficiency, and execution efficiency. The complete experimental results are presented in Table C.3. It should be noted that there is an inherent unfairness in the measurement of planning time, as we have excluded data where the planning time exceeds 5 seconds. Similarly, execution cost, path length, and ticks also exclude data from failed planning attempts. These metrics are only considered fair when the success rate of the compared algorithms is 100%.



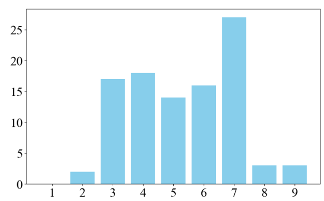
(a) Actions in RoboWaiter



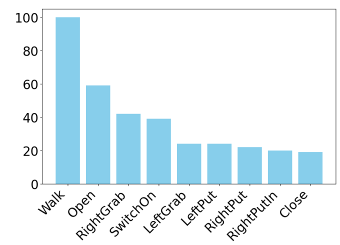
(b) Predicates in RoboWaiter



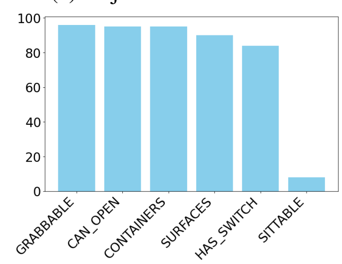
(c) Objects in RoboWaiter



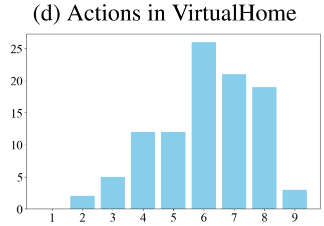
(d) Actions in VirtualHome



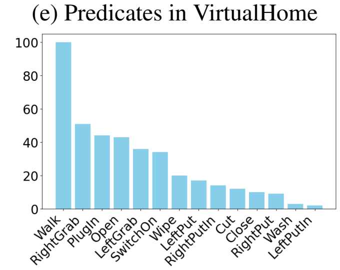
(e) Predicates in VirtualHome



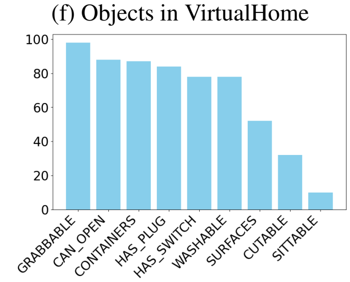
(f) Objects in VirtualHome



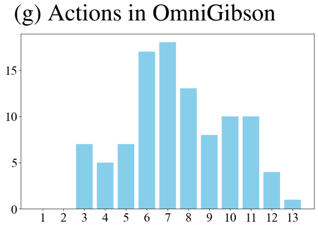
(g) Actions in OmniGibson



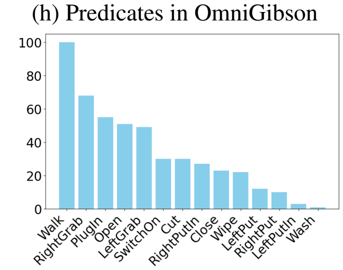
(h) Predicates in OmniGibson



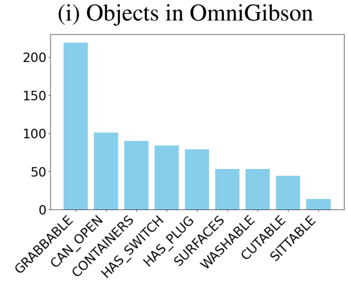
(i) Objects in OmniGibson



(j) Actions in RobotHow



(k) Predicates in RobotHow



(l) Objects in RobotHow

Figure 3: The distribution of multi-goal tasks in different scenarios.

Table 1: The comparison of BT planning algorithms in common metrics.

Algorithm	SR	Timeout Rate	Explored Conditions	Planning Time	Execution Cost	Path Length	Ticks
Single-goal							
Scenario: RoboWaiter							
Reactive Planning	85%	0%	2.88	0.000	8.66	1.24	14.49
BT Expansion	100%	0%	6.97	0.011	15.31	1.68	126.60
OBTEA	100%	0%	2.95	0.001	15.31	1.68	22.31
HBTP	100%	0%	3.23	0.001	15.31	1.68	23.53
HBTP-Oracle	100%	0%	2.83	0.000	15.31	1.68	24.47
Scenario: VirtualHome							
Reactive Planning	69%	0%	3.96	0.000	19.22	1.75	27.09
BT Expansion	100%	0%	75.27	0.244	32.27	3.05	2409.59
OBTEA	100%	0%	7.52	0.005	27.02	2.60	63.45
HBTP	100%	0%	4.29	0.005	27.02	2.60	45.12
HBTP-Oracle	100%	0%	4.05	0.004	27.02	2.60	44.35
Scenario: OmniGibson							
Reactive Planning	48%	0%	4.20	0.000	18.48	1.60	24.52
BT Expansion	100%	0%	27.42	0.121	34.48	3.25	818.45
OBTEA	90%	10%	12.08	0.005	30.53	2.86	130.12
HBTP	100%	0%	17.63	0.010	32.38	3.07	176.95
HBTP-Oracle	100%	0%	5.15	0.002	32.38	3.07	63.03
Scenario: RobotHow							
Reactive Planning	41%	0%	4.46	0.004	20.02	1.78	27.24
BT Expansion	59%	41%	70.52	2.196	25.58	2.32	402.25
OBTEA	73%	27%	130.32	0.746	30.70	2.89	89.21
HBTP	87%	13%	62.04	0.732	33.64	3.23	65.02
HBTP-Oracle	100%	0%	5.42	0.116	34.27	3.31	68.09
Multi-goal							
Scenario: RoboWaiter							
Reactive Planning	0%	0%	2.00	0.000	-	-	-
BT Expansion	100%	0%	92.91	0.165	29.86	3.55	1778.68
OBTEA	100%	0%	17.73	0.004	29.14	3.45	171.05
HBTP	100%	0%	14.18	0.004	29.14	3.45	170.09
HBTP-Oracle	100%	0%	10.68	0.003	29.14	3.45	128.05
Scenario: VirtualHome							
Reactive Planning	2%	0%	2.74	0.000	18.00	2.00	32.00
BT Expansion	41%	59%	379.25	3.387	43.34	4.00	4713.71
OBTEA	96%	4%	435.00	0.424	54.16	5.21	5815.55
HBTP	98%	2%	218.70	0.402	55.07	5.31	1925.74
HBTP-Oracle	100%	0%	10.16	0.025	55.22	5.33	213.57
Scenario: OmniGibson							
Reactive Planning	3%	0%	3.76	0.000	24.00	2.33	40.67
BT Expansion	71%	29%	717.04	2.505	61.17	5.76	8999.79
OBTEA	86%	14%	279.57	0.180	62.99	5.94	4763.50
HBTP	100%	0%	152.73	0.143	64.40	6.12	1696.84
HBTP-Oracle	100%	0%	15.37	0.013	63.95	6.09	336.67
Scenario: RobotHow							
Reactive Planning	0%	0%	4.36	0.005	-	-	-
BT Expansion	6%	94%	112.31	4.818	34.33	3.00	5809.33
OBTEA	7%	93%	871.26	4.530	38.86	3.29	183.43
HBTP	32%	68%	276.38	3.785	63.12	6.19	493.53
HBTP-Oracle	93%	7%	28.12	1.777	74.56	7.32	632.35