Hierarchical Task and Motion Planning in the Now

By: Leslie Pack Kaelbling and Tomás Lozano-Pérez

Presenter: Yoonwoo Kim

[Oct. 17. 2023]

Motivation: Laundry

What series of actions should I take?

Motivation: Laundry

What series of actions should I take?

1. Take the clothes
2. Put it into the washer
3. Wash it
4. Move it to the dryer
5. Dry it
6. Move it to the closet

Motivation: Laundry

Hey Robot, can you…

Motivation: Laundry

Hey Robot, can you…

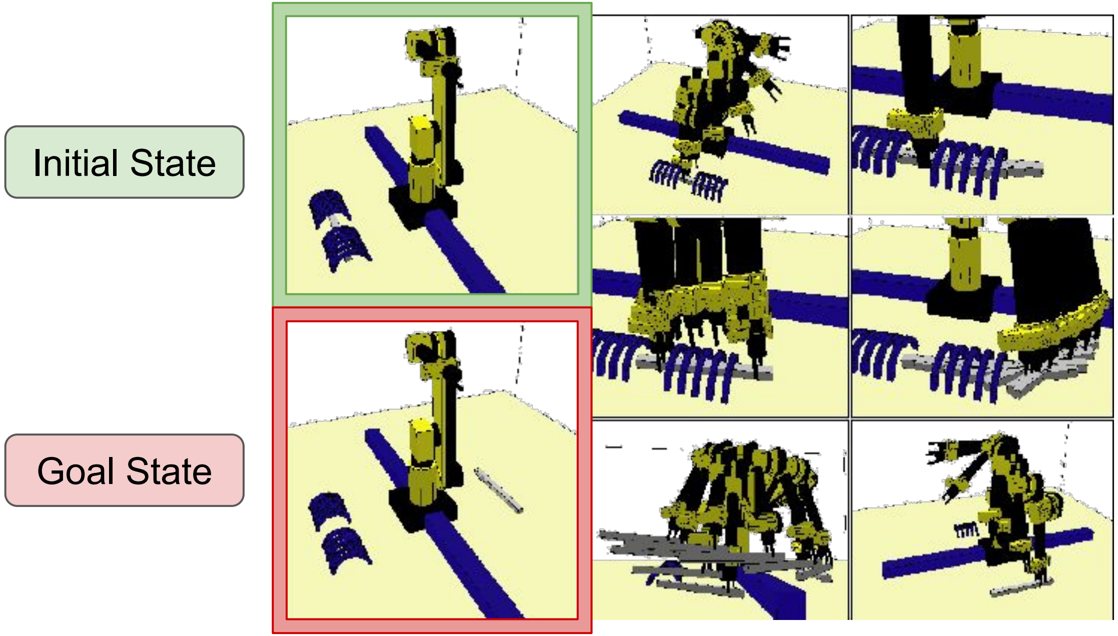
HARD!!!

- Non-determinism in the environment or in the low-level motion planner.
- Integration of task planner and geometric planners is non-trivial.

Related Work

Manipulation planning

- Generate robot motion sequences allowing the **manipulation** of movable objects among obstacles.



Siméon, Thierry, et al. "Manipulation planning with probabilistic roadmaps." *The International Journal of Robotics Research* 23.7-8 (2004): 729-746.

Related Work

Manipulation planning

- Generate robot motion sequences allowing the **manipulation** of movable objects among obstacles.

Integrating symbolic and motion planning

- Integrates **symbolic** task planner **geometric** motion planner.

Hierarchical planning

- **Symbolic** task planner which utilizes hierarchy.

Problem Setting: Representation

Fluents: Symbolic predicate that characterizes the logics aspect of the domain.

Example)

- In(O, R)
- Overlaps(O, R)
- ClearX(R, Os)
- Holding()
- Clean(O)

Problem Setting: Representation

Fluents: Symbolic predicate that characterizes the logics aspect of the domain.

World State: Detailed description of the environment

Example)

- Geometric (configuration of the robot, pose and shape of each objects)
- Fluents (grasped, clean)

Problem Setting: Representation

Fluents: Symbolic predicate that characterizes the logics aspect of the domain.

World State: Detailed description of the environment

Goals: Conjunction of fluents with values

Example)

- In(A, storage) = True $\wedge$ Clean(A) = True

Problem Setting: Representation

Fluents: Symbolic predicate that characterizes the logics aspect of the domain.

World State: Detailed description of the environment

Goals: Conjunction of fluents with values

Operators: Characterized by primitive actions

Example)

- Pick(O)
- Place(O, R)
- Wash()

Problem Setting: Operators

STRIPS-style

Target Fluent

$$\boxed{F(A_1, \ldots, A_n) = V:}$$

exists: $B_1, \ldots, B_k$
pre: $\phi_1, \ldots, \phi_m$
sideEffects: $\psi_1, \ldots, \psi_l$
prim: π
cost: c

If the **primitive action** is executed in any world state where all of the **preconditions** hold, the **target fluent** and **side effect fluents** will have the values specified while everything else will remain the same.

Problem Setting: Operators Example

$Holding() = O$:

define: $Ts = \{T : ClearX(T, X) \in \textbf{goal} \wedge O \notin X\}$

exists: $L \in \{Location(O, \textbf{start}),$
$\boxed{SuggestParking(O, Ts, \textbf{start})\}}$

$P \in \boxed{SuggestPaths(O, L, home, \textbf{start})}$

pre:
$\boxed{0.}$ $Holding() = nothing$
$\boxed{0.}$ $In(O, L) = True$
$\boxed{2.}$ $ClearX(sweptVol(P), [O]) = True$

sideEffects: $\forall L'.In(O, L') = False$

prim: $Pick(O)$

Abstraction Level

Suggestors

- Allows operating on continuous domain
- Restrict domains for existential variables

Problem Setting: More Operators

$Holding() = O$:

define: $Ts = \{T : ClearX(T, X) \in \textbf{goal} \wedge O \notin X\}$
exists: $L \in \{Location(O, \textbf{start}), SuggestParking(O, Ts, \textbf{start})\}$
$P \in SuggestPaths(O, L, home, \textbf{start})$
pre: 0. $Holding() = nothing$
0. $In(O, L) = True$
2. $ClearX(sweptVol(P), [O]) = True$
sideEffects: $\forall L'.In(O, L') = False$
prim: $Pick(O)$

$In(O, R) = True$:

define: $Ts = \{T : ClearX(T, X) \in \textbf{goal} \wedge O \notin X\}$
exists: $P \in SuggestPaths(O, R, home, \textbf{start})$
pre: 1. $Holding() = O$
2. $ClearX(sweptVol(P), [O]) = True$
sideEffects: $Holding() = nothing$
prim: $Place(R)$

$ClearX(R, Os) = True$:

pre: 1. $\forall X \in Objects - Os : Overlaps(X, R) = False$
prim: none

$Overlaps(O, R) = False$:

define: $Ts = \{T : ClearX(T, X) \in \textbf{goal} \wedge O \notin X\} \cup \{R\}$
exists: $L = SuggestParking(O, Ts, \textbf{start})$
pre: 1. $In(O, L) = True$, $ClearX(L, [O]) = True$
prim: none

$Clean(O) = True$:

pre: 1. $In(O, \text{WASHER})$
prim: $Wash()$

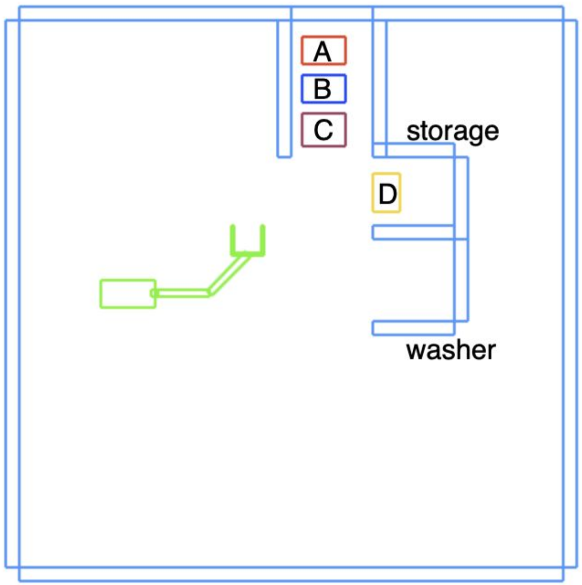
Key Idea

1. **Given the goal, what action can I take now?**
2. **Create Subgoals until primitive action is found**
3. **Execute primitive action**

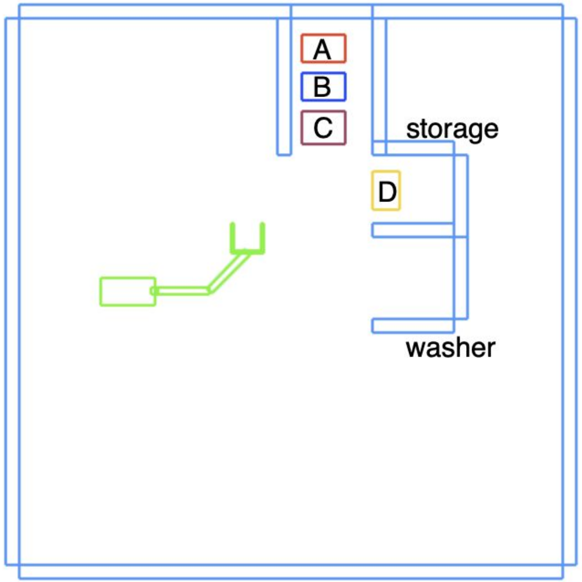
Recursive DFS with an order of preconditions to decide which child node to descend first.

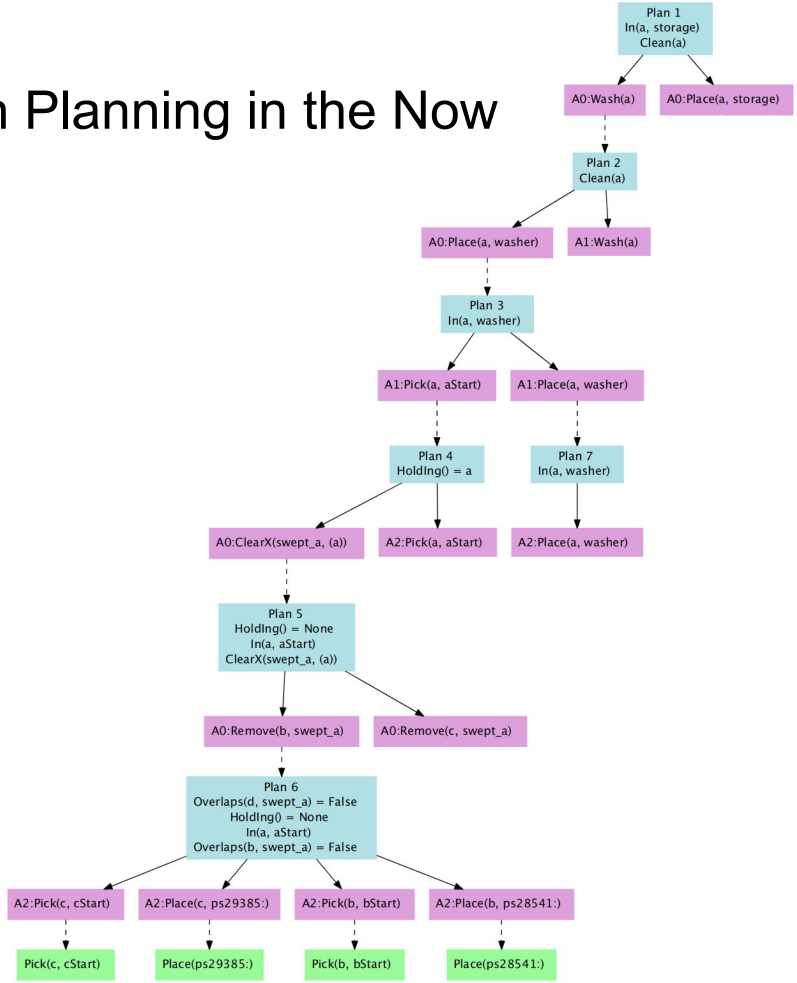
Plan 1
In(a, storage)
Clean(a)

Hierarchical Task and Motion Planning in the Now

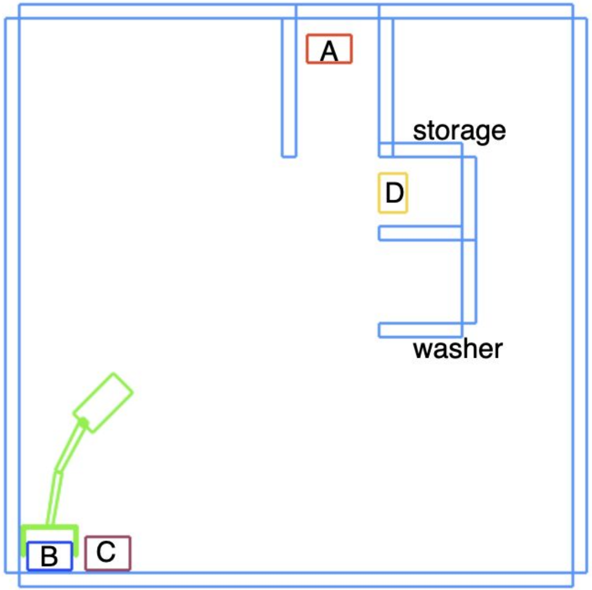


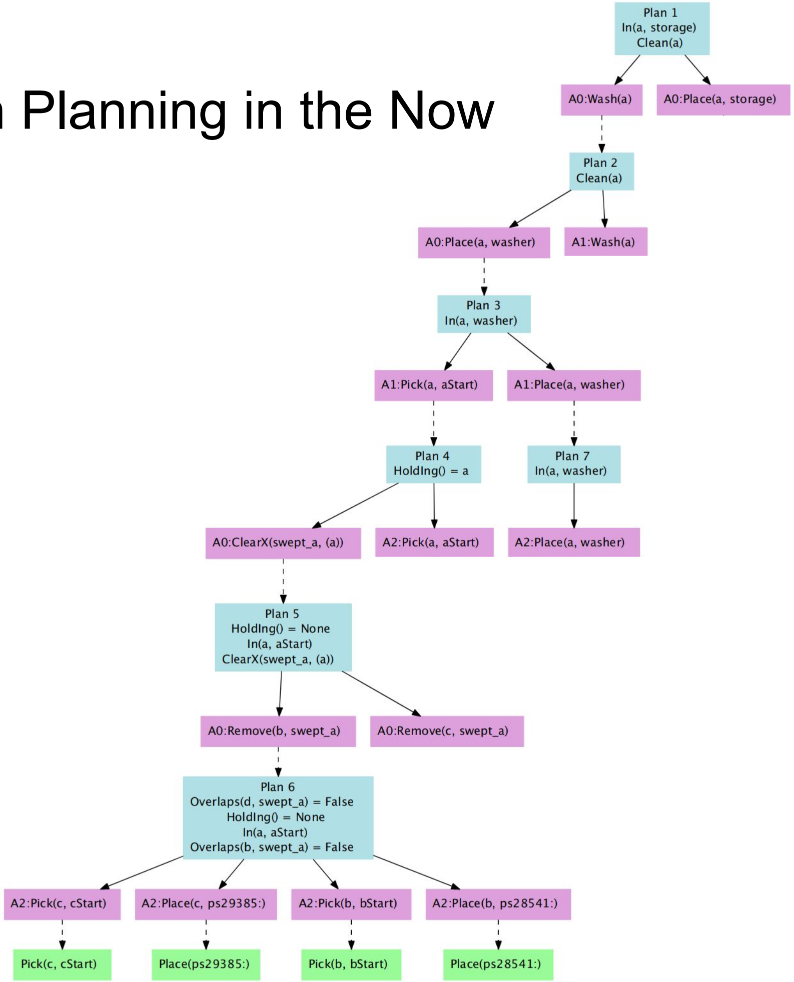
Hierarchical Task and Motion Planning in the Now



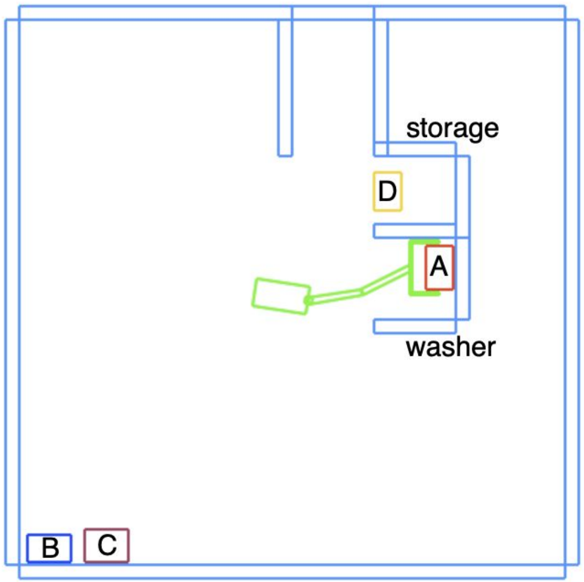


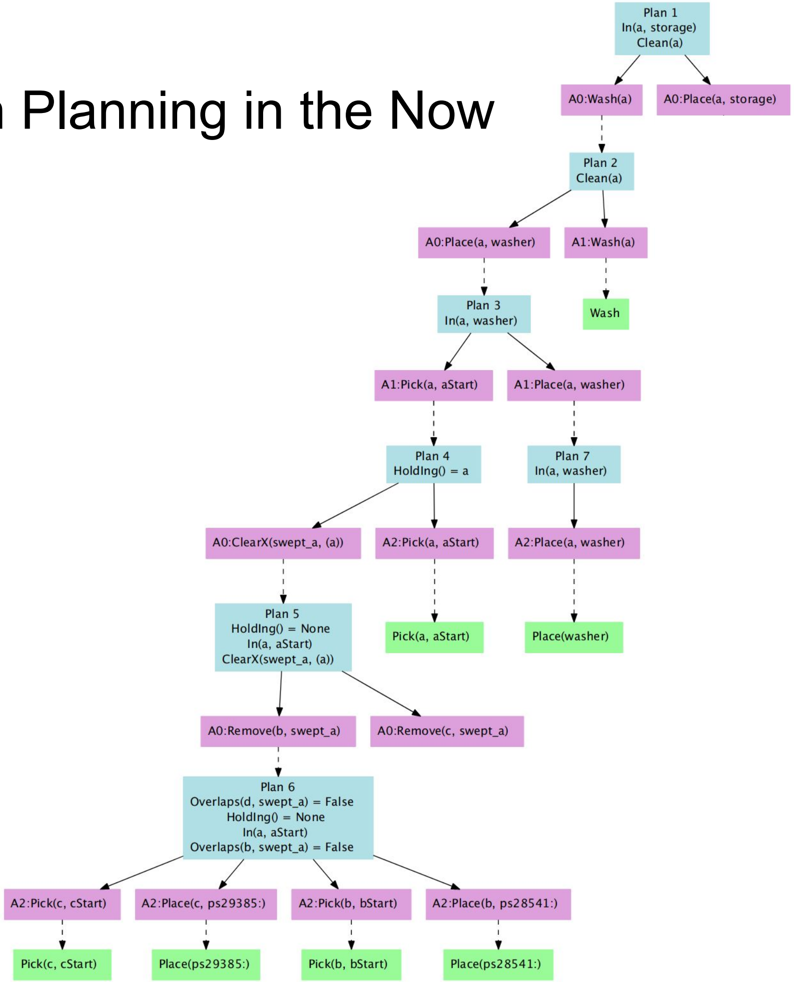
Hierarchical Task and Motion Planning in the Now



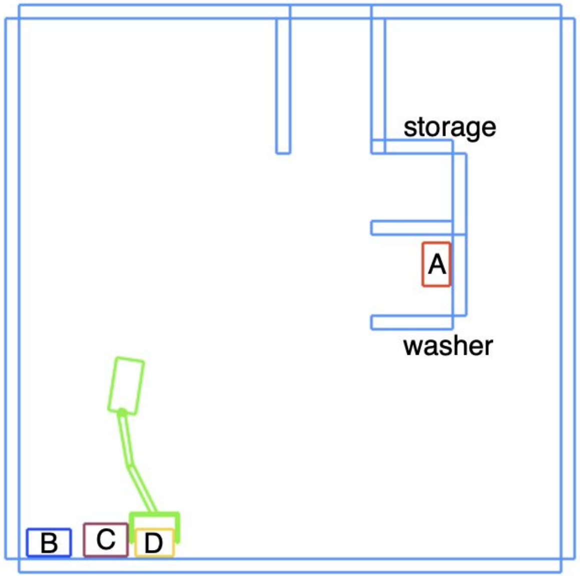


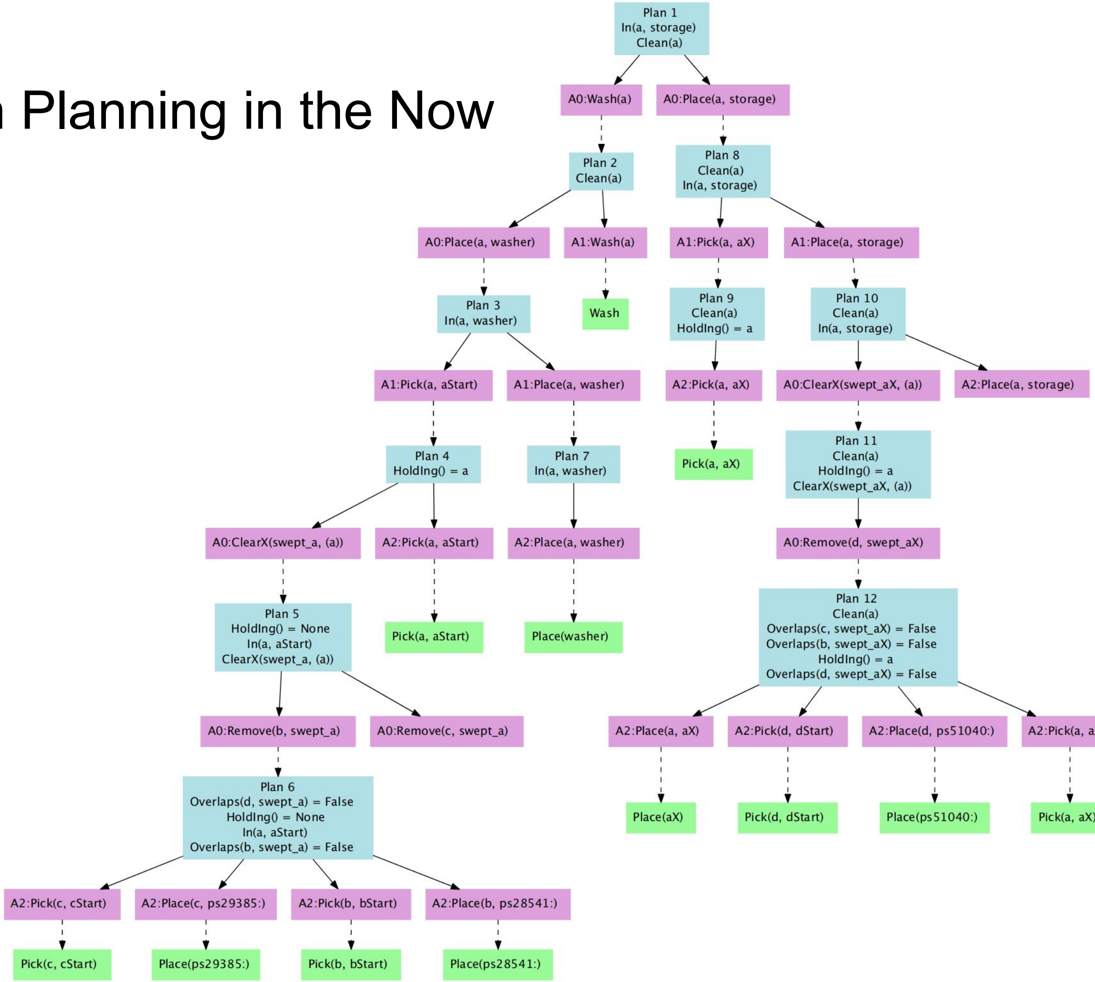
Hierarchical Task and Motion Planning in the Now



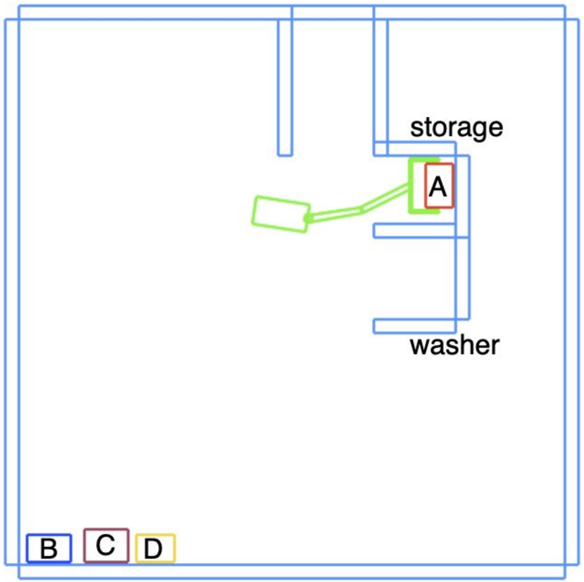


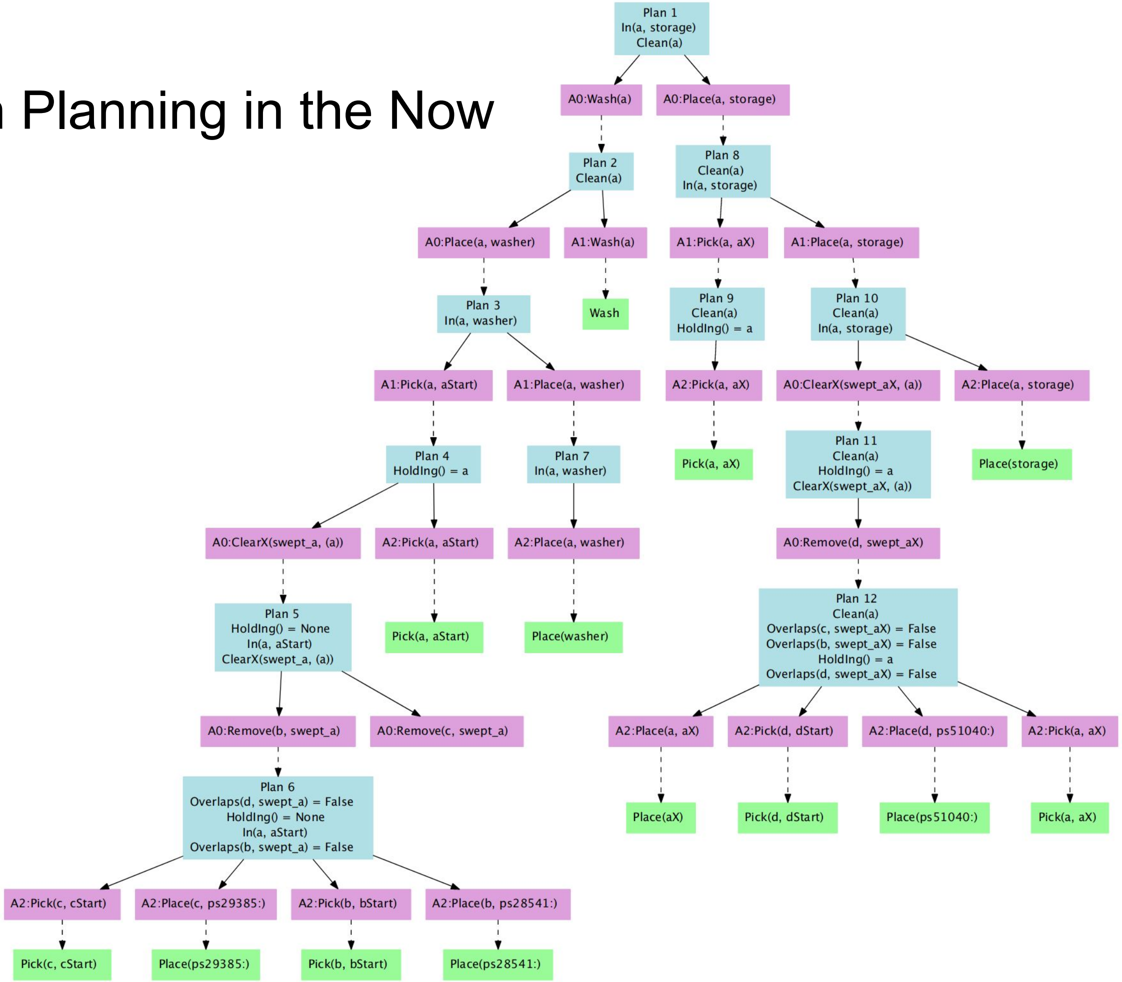
Hierarchical Task and Motion Planning in the Now





Hierarchical Task and Motion Planning in the Now





Hierarchy

Postponing consideration of some or all preconditions of an operator.

pre: $p_1, \ldots, p_n$
prim: o

$$\downarrow$$

pre: $p_1, \ldots, p_{n-1}$
prim: achieve p_n maintaining $p_1, \ldots, p_{n-1}$; o

$Clean(O) = True$:
  pre: 1. $In(O, \text{WASHER})$
  prim: $Wash()$

$$\downarrow$$

$Clean(O) = True$:
  pre:
  prim: $In(O, \text{WASHER})$ $Wash()$

Hierarchy

Postponing consideration of some or all preconditions of an operator.

$Clean(O) = True$:

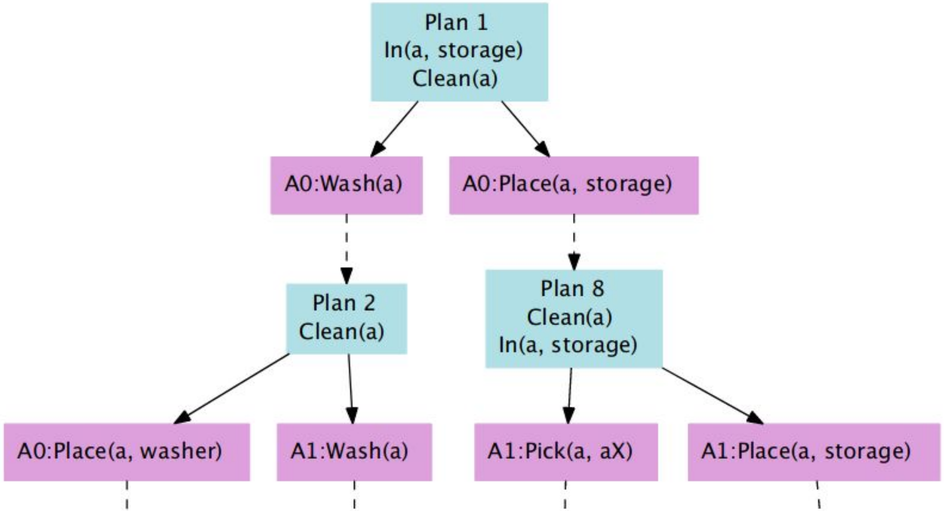
pre: 1. $In(O, \text{WASHER})$

prim: $Wash()$

$Clean(O) = True$:

pre:

prim: $In(O, \text{WASHER})\ Wash()$



HPN Algorithm

HPN($currentState$, $goal$, $operators$, $absLevel$, $world$):
    if holds($goal$, $currentState$):
        return TRUE
    else p = PLAN($currentState$, $goal$, $operators$, $absLevel$)
        for (o_i, g_i) **in** p
            if prim(o_i):
                $currentState = world.execute(o_i)$
            else HPN($currentState$, g_i, $operators$,
                    NEXTLEVEL($absLevel$, o_i), world)

Return a list $\left[\left(o_1, g_1\right), \left(o_2, g_2\right), \ldots, \left(o_n, g_n\right)\right]$ where o_i are operator instances, g_n is the goal, g_i is the weakest precondition of g_{i+1} under o_{i+1}

Theory: Theorem

If:

1) PDD specified by operators ***ops*** at abstraction level ***H*** is complete and correct formalization of the primitive actions of domain ***w***
2) **Start** has static connectivity in that domain
3) **Goal** is reachable from start

Then

HPN(**start**, **Goal**, **ops**, **H**, **w**) will cause world w be in state s $\in$ G

Guarantees if a goal state was reachable from the starting state under some sequence of operations, HPN will eventually cause the system to reach a goal state.

Experimental Results

Domains:

- Wash
- Household
- Swap

Achievements:

- Can handle different domains with
- Plans with no or few redundant steps

Domain	Num	Longest	Steps
swap	22	4	8
wash	14	4	13
wash all	26	6	22
clean house	89	4	36
clean and tidy	169	7	65

Limitations

- Empirical results were not benchmarked against any baseline algorithms.
- Selecting a hierarchical formalization is non-trivial
    - Requires good domain knowledge.
- Does not leverage “Learning”
    - Same amount of computation even if same problem is given.

Future Work for Paper / Reading

- How can we leverage learning for Task and Motion Planning?
    - What should we learn?
- Selecting hierarchical formalization requires domain-dependent choices. LLMs seem to have domain knowledge of everyday human environment. Can we leverage LLMs to formulate the hierarchy? If we can what should change? if we cannot why not?
- Can we use learned policy as primitive actions for contact rich or more complicated actions? If we do, how should the formulation of operators change?

Extended Readings

Task and Motion Planning in Belief Space

- Kaelbling, Leslie Pack, and Tomás Lozano-Pérez. "Integrated task and motion planning in belief space." *The International Journal of Robotics Research* 32.9-10 (2013): 1194-1227.

Review of Task and Motion Planning which contains more recent work

- Garrett, Caelan Reed, et al. "Integrated task and motion planning." *Annual review of control, robotics, and autonomous systems* 4 (2021): 265-293.

Learning for Task and Motion Planning

- Yang, Zhutian, Caelan Reed Garrett, and Dieter Fox. "Sequence-Based Plan Feasibility Prediction for Efficient Task and Motion Planning." *arXiv preprint arXiv:2211.01576* (2022).

Imitation learning with TAMP planner

- Dalal, Murtaza, et al. "Imitating Task and Motion Planning with Visuomotor Transformers." *arXiv preprint arXiv:2305.16309* (2023).

Summary

Problem the reading is discussing

- Integrating task and motion planning

Why is it important and hard

- Non-determinism in the environment or in the low-level motion planner.
- Integration of task planner and geometric planners is non-trivial.

What is the key limitation of prior work

- Manipulation planning didn’t consider high-level symbolic planning.
- Integrated symbolic and motion planning didn’t consider non-determinism in the environment
- Hierarchical planning didn’t consider integrating continuous geometric motion planning.

What is the key insights

- Hierarchical planning for TAMP has the potential for dramatic speedups.
- Proper design of the suggesters can constrain task planning by limiting suggested states