TAPS: Task-Agnostic Policy Sequencing

Christopher Agia*, Toki Migimatsu*, Jiajun Wu, and Jeannette Bohg
Stanford AI Lab

TOYOTA RESEARCH INSTITUTE

Action dependencies in sequential manipulation

Sequential manipulation tasks often contain geometric dependencies between actions. Attempting to solve sequential manipulation tasks with greedy execution may fail due to these action dependencies.

Greedy execution

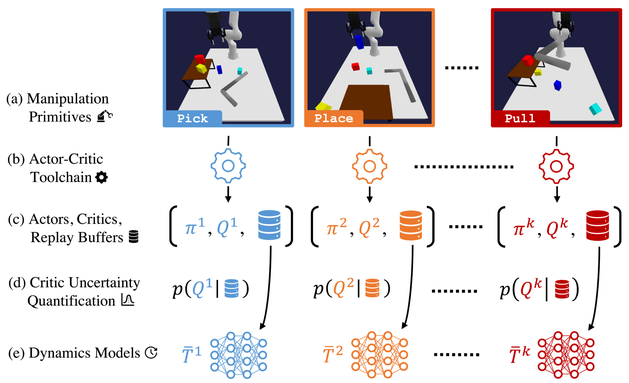
Planning with TAPS (Ours)

TAPS primitives are trained independently

TAPS outlines both a scalable training pipeline and a motion planning framework; planning relies on learning three key components:

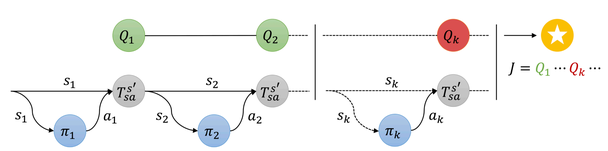
- Actors & Critics via off-the-shelf RL algorithms (e.g., SAC, TD3) for controlling manipulation primitives and estimating affordances;
- Dynamics Models trained on transition experience of the policies;
- Uncertainty Quantification over the Q-networks to detect and out-of-distribution states and actions at plan-time.

Learning skills independently instead of jointly has an advantage—new skills can be added to a library without retraining old ones.



TAPS resolves action dependencies via planning

Given any sequence of policies, TAPS performs sampling-based optimization to obtain actions that maximize the probability of task success, estimated by the product of downstream Q-values.



Planning objective: maximize long-horizon task success

$$
\begin{aligned}
p(\text{task success} \mid \bar{s}_1, a_{1:H}) &= p(r_1 = 1, \ldots, r_H = 1 \mid \bar{s}_1, a_{1:H}) \\
&= \Pi_{h=1}^{H} p(r_h = 1 \mid \bar{s}_1, a_{1:h}) \\
&= E_{\bar{s}_2 \sim \bar{T}_1(\bar{s}_1, a_1), \ldots, \bar{s}_H \sim \bar{T}_{H-1}(\bar{s}_{H-1}, a_{H-1})} [\Pi_{h=1}^{H} p(r_h = 1 \mid \bar{s}_h, a_h)] \\
&= E_{\bar{s}_2 \sim \bar{T}_1(\bar{s}_1, a_1), \ldots, \bar{s}_H \sim \bar{T}_{H-1}(\bar{s}_{H-1}, a_{H-1})} [\Pi_{h=1}^{H} Q_h(\Psi_h(\bar{s}_h), a_h)]
\end{aligned}
$$

Hook Reach

Greedy execution

Planning with TAPS (Ours)

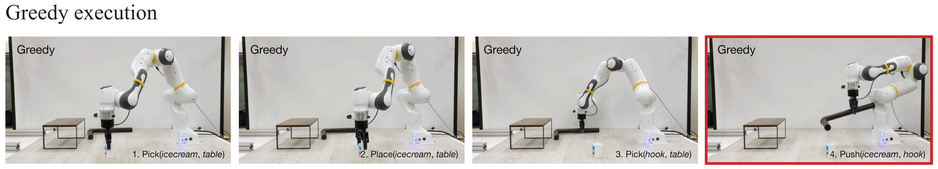
Constrained Packing

Greedy execution

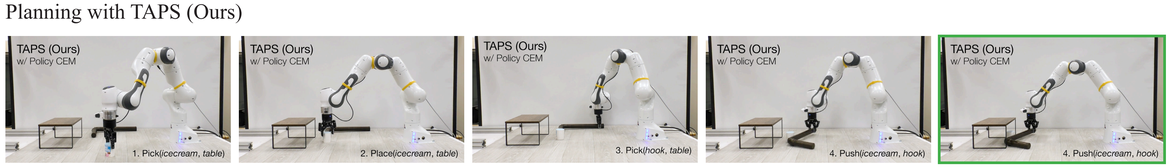
Planning with TAPS (Ours)

Rearrangement Push

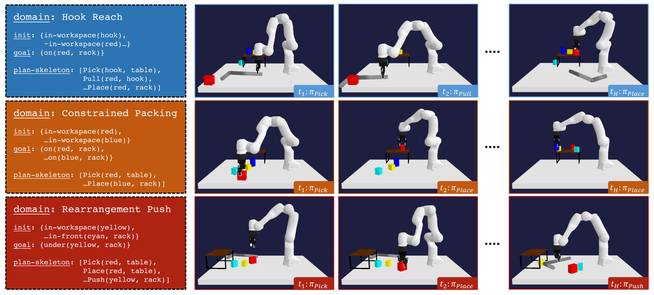
Greedy execution



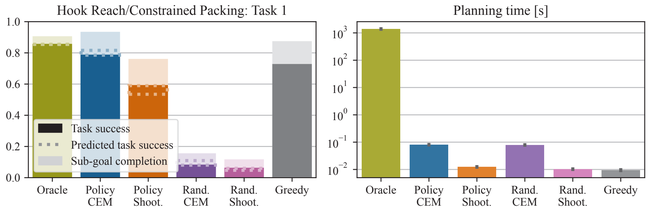
Planning with TAPS (Ours)



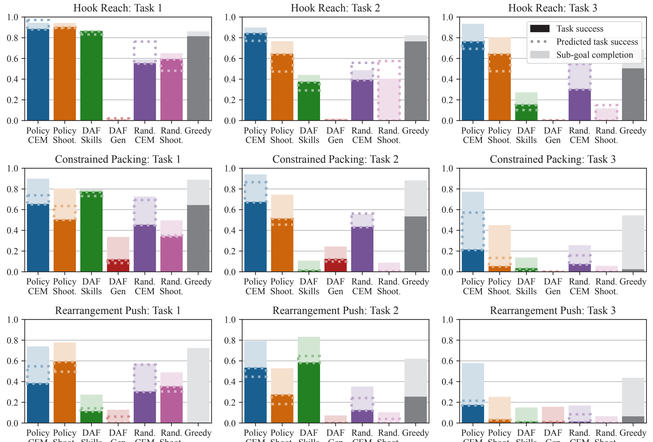
Sequential manipulation tasks



Maximizing product of Q-functions → task success



TAPS primitives generalize to unseen tasks



TAPS + task planning can solve TAMP problems