A Environment Details

We choose two tasks, TABLE_LACK and CHAIR_INGOLF, from the IKEA furniture assembly environment [17] for our experiments. For reinforcement learning, we use a heavily shaped multi-phase dense reward from [17]. For the robotic agent, we use the 7-DoF Rethink Sawyer robot operated via joint velocity control. For imitation learning, we collected 200 demonstrations for each furniture part assembly with a programmatic assembly policy. Each demonstration for single-part assembly consists of around 200-900 steps long due to the long-horizon nature of the task.

The observation space includes robot observations (29 dim), object observations (35 dim), and task phase information (8 dim). The robot observations consist of robot joint angles (7 dim), joint velocities (7 dim), gripper state (2 dim), gripper position (3 dim), gripper quaternion (4 dim), gripper velocity (3 dim), and gripper angular velocity (3 dim). The object observations contain the positions (3 dim) and quaternions (4 dim) of all five furniture pieces in the scene. In addition, the 8-dimensional task information is an one-hot embedding representing the phase of the state (*e.g.* reaching, grasping, lifting, moving, aligning).

In our experiments, we define a subtask as assembling one part to another; thus, we have *four* subtasks for each task. Subtasks are independently trained on the initial states sampled from the environment with random noise ranging from $[-2\text{cm}, 2\text{cm}]$ and $[-3°, 3°]$ in the (x, y)-plane. Moreover, this subtask decomposition is given, *i.e.*, the environment can be initialized for each subtask and the agent is informed when the subtask is completed. Once two parts are attached, the corresponding subtask is completed and the robot arm moves back to its initial pose in the center of the workplace.

B Network Architectures

For fair comparison, we use the same network architecture for our method and the baseline methods. The policy and critic networks consist of two fully connected layers of 128 and 256 hidden units with ReLU nonlinearities, respectively. The output layer of the actor network outputs an action distribution, which consists of the mean and standard deviation of a Gaussian distribution. The critic network outputs only a single critic value. The discriminator for GAIL [29] and initiation set discriminator for our proposed method use a two-layer fully connected network with 256 hidden units. $\tanh$ is used as a nonlinear activation function. These discriminators' outputs are clipped between $[0, 1]$ following Peng et al. [53].

C Training Details

During policy learning, we sample a batch of 128 elements from the expert (for GAIL) and agent experience buffers. To acquire subtask polices for the policy sequencing baseline [6] and our method, we run from 20M to 25M steps depending on the difficulty of each subtask. We train all methods except BC for 48 hours which collect 200M steps for PPO, GAIL, GAIL + PPO; and 5M steps for SPiRL. We train the policy sequencing baseline and our method for additional 100M steps; thus, in total 200M steps. For the final evaluation, we evaluate a trained policy every 50 updates and report the best performing checkpoints due to unstable adversarial training. Most of the runs are converged within these training steps.

BC [22]: The policy is trained with batch size 256 for 1000 epochs using the Adam optimizer with learning rate 1e-4 and momentum (0.9, 0.999).

PPO [54]: Any reinforcement learning algorithm can be used for policy optimization, but we choose to use Proximal Policy Optimization (PPO) [54] and we use the default hyperparameters of PPO (see Table 2).

GAIL [29]: We tested different variants of GAIL implementation and found the AMP [53] formulation is most stable. Please refer to the paper [53] for implementation

Table 2: PPO hyperparameters.

Hyperparameter	Value
# Workers	16
Rollout Size	1,024
Learning Rate	0.0003
Learning Rate Decay	Linear decay
Mini-batch Size	128
# Epochs per Update	10
Discount Factor	0.99
Entropy Coefficient	0.003
Reward Scale	0.05
State Normalization	True

details about GAIL training and GAIL reward. In this paper, we specifically use an agent states s to discriminate agent and expert trajectories, instead of state-action pairs (s, a).

GAIL + PPO: We use the same setup with the GAIL baseline, but with the environment reward. We use $\lambda_1 = \lambda_2 = 0.5$.

SPiRL [55]: We use the official implementation of the original paper.

Policy Sequencing [6]: We use the same implementation and hyperparameters as ours, but without the terminal state regularization. To prevent catastrophic forgetting, we sample an initial state from the environment for 20% and sample from the Gaussian distribution for 80%.

T-STAR (Ours): We use $\lambda_3 = 10000$ for the terminal state regularization. Similar to PS, we sample an initial state from the environment for 20% and sample from the terminal state buffer of the previous skill for 80%.