# Mastering Scene Rearrangement with Expert-Assisted Curriculum Learning and Adaptive Trade-Off Tree-Search

Zan Wang*, Hanqing Wang*, and Wei Liang†

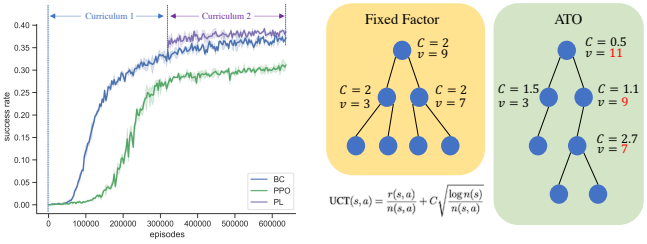
* indicates equal contribution † indicates corresponding author

***Abstract*— Scene Rearrangement Planning (SRP) has recently emerged as a crucial interior scene task; however, current approaches still face two primary issues. First, prior works define the action space of SRP using handcrafted coarse-grained actions, which are inflexible for scene arrangement transition and impractical for real-world deployment. Secondly, the scarcity of realistic indoor scene rearrangement data hinders popular data-hungry learning approaches and quantitative evaluation. To tackle these issues, we propose a fine-grained action space definition and curate a large-scale scene rearrangement dataset to facilitate the training of learning approaches and comprehensive benchmarking. Building upon this dataset, we introduce a novel framework, PLATO, designed for efficient agent training and inference. Our approach features an *exPert-assisted curriculum Learning* (PL) paradigm that possesses a Behavior Cloning (BC) and an offline Reinforcement Learning (RL) curriculum for agent training, along with an advanced tree-search-based planner enhanced by an *Adaptive Trade-Off* (ATO) strategy to improve expert agent performance further. We demonstrate the superior performance of our method over baseline agents through extensive experiments and provide a detailed analysis to elucidate its rationale. Our project website can be accessed at pl-ato.github.io.**

## I. Introduction

Rearrangement involves transforming a physical environment to achieve a pre-defined goal state, facilitating practical applications like table setting [10, 11, 42], object packing [14, 16], and furniture reconfiguration [36, 38, 39]. To achieve these goals, agents must adeptly analyze spatial layouts, plan optimal movement trajectories, and skillfully manipulate objects. Extensive studies have delved into rearrangement planning in robotics, *e.g.*, developing robotic arms to manipulate objects on tabletops through grasping, picking, and nonprehensile actions [21, 22, 31, 35, 41].

Recently, the emergence of Scene Rearrangement Planning (SRP) fills the gap between indoor scene synthesis [24, 40, 43] and layout realization, attracting significant attention from researchers [3, 36, 38, 39]. SRP aims at generating a feasible movement plan for agents to transition from an initial scene layout to a target scene layout by moving furniture. Compared to conventional tabletop rearrangement tasks [9, 15, 19, 42], SRP presents significantly greater complexity and challenges. First, objects typically have regular shapes (*e.g.*, cubes and cylinders) in tabletop settings, whereas furniture objects vary considerably in shape and size in realistic scenes. Second, obstacles and boundaries are usually sparse or absent in tabletop scenarios, while 3D indoor environments feature irregular layouts, resulting in obstacles like walls and pillars. Third, the large scale and intricacy of the scene make the decision sequence much longer, further increasing the challenge.




(a) Training Curve (b) Fixed Factor *vs* ATO on MCTS

Fig. 1: **PL and ATO.** (a) **PL** achieves significant improvement compared to prior learning paradigms. (b) With the **ATO** strategy, tree-search can balance exploitation and exploration by adaptively assigning $C$ to improve search efficiency.

Despite the extensive efforts to address SRP, two critical issues persist. First, prior works shorten the decision sequence by designing action spaces that align with specific movement mechanisms [39] or by relying solely on perfect path planning [36]. However, such limited high-level action definition may restrict transitions between states that require low-level actions, potentially leading to unsolvable scenarios. Second, the quantitative evaluation has been limited to just a few dozen real scenes [36, 38], which undermines the statistical robustness of the evaluation.

To address the above issues, we first define a fine-grained action space to support low-level object manipulation. Additionally, we curate a large-scale dataset of 13,215 diverse furnished rooms for SRP by utilizing 3D-FRONT [7], facilitating the training of data-hungry learning approaches and comprehensive quantitative evaluation. This new large-scale dataset poses a more challenging SRP benchmark, characterized by long decision sequences and diverse environment layouts. To solve the challenging task, we propose a novel framework, **PLATO**, which comprises an *exPert-assisted curriculum Learning* (**PL**) paradigm for efficient agent training and a powerful tree-search-based planner enhanced with *Adaptive Trade-Off* (**ATO**) strategy.

Specifically, our **PL** paradigm comprises two curriculums: (i) expert behavior cloning and (ii) offline RL on expert demonstrations. These two curriculums synergistically address each other's limitations, showcasing superior performance compared to prior learning paradigms, as shown in

All authors are with the Beijing Institute of Technology, Beijing, China {wangzan, hanqingwang, liangwei}@bit.edu.cn.

Fig. 1 (a). In the first curriculum, the agent learns from an expert by imitating the expert behavior, which privileges the agent of fast convergence and mitigates the distributional shift in offline RL. The second curriculum refines the agent's policy towards optimality through the RL objective using expert demonstrations; these demonstrations potentially mitigate issues associated with sparse reward and high training variance in online RL. Additionally, we develop a strong tree-search-based planner integrated with a policy network to address SRP. This planner is further enhanced by an **ATO** strategy, which effectively leverages the problem's intrinsic properties to balance exploitation and exploration within the search algorithm. Our results show a significant improvement in search efficiency, as depicted in Fig. 1 (b).

Our contributions are summarized as follows:

- We propose a more general SRP setting with a flexible and tractable definition of the action space. Additionally, we introduce a large-scale indoor scene dataset comprising $13,215$ diverse and realistic furnished room layouts for benchmarking SRP.
- We introduce a novel curriculum learning paradigm that efficiently trains agents to solve SRP. Extensive experiments on our curated dataset demonstrate this paradigm's superior performance over baseline agents.
- We develop an advanced search strategy designed to enhance the efficiency of the tree-search algorithm, showcase its effectiveness through experimental validation, and discuss the rationale.

## II. Related Work

### A. Rearrangement Planning

Given an initial and target configuration of a specified object set, Rearrangement Planning focuses on generating a feasible action sequence to transform the initial configuration into the target. Robotics researchers have explored this topic for decades, with most efforts concentrated on solving tabletop rearrangement tasks [9–11, 15, 19, 22, 42] using robotic arms [13, 21]. Wang *et al*. [36] were the first to formulate the SRP task, solving it by selecting pre-searched paths to move objects. Weihs et al [38] train agents to return objects in a room to their original states after recording their initial positions; however, the manipulated objects are typically small and lack mutual obstacles. Building on Wang *et al*. [36], our work represents both the initial and target scene layouts as discrete grids, avoiding the use of goal images [11]. Additionally, we follow prior work on object rearrangement [22, 31, 35, 41] by adopting a nonprehensile action space rather than their coarse-grained action definition and curate a large-scale scene rearrangement dataset for SRP agent training and benchmarking.

### B. Offline Reinforcement Learning

Reinforcement Learning (RL) provide a mathematical online learning paradigm to extract policies from the interactive experiences between agents and environments. Nevertheless, such an "online" feature becomes a prohibitive obstacle to deploying Deep RL in many scenarios where continuously collecting experiences with the latest learned policy is inefficient and impractical. To address this challenge, utilizing previously collected offline data to learn policies with RL objectives [8] has become an appealing approach. Recent works leverage the power of data-driven methods by adopting off-policy reinforcement learning techniques for dialogue [17] and robotic manipulation [6]. Unfortunately, many off-policy RL algorithms are vulnerable to distributional shift, *i.e*., the agent is evaluated on a different distribution because the behavior of the learned policy leads to unvisited states. A way to mitigate this issue is to limit the distributional shift by constraining the region of policy update [27, 29]. Some works adopt importance sampling to derive an unbiased estimator of learning objective [18, 33]. Different from those methods, our approach adopts a curriculum learning [23] strategy to mitigate distributional shifts. Specifically, we first train the agent to imitate the action sequences from the collected demonstrations. Subsequently, we refine the agent's policy towards optimality using a RL objective, utilizing the expert demonstrations as well. This two-stage training approach is similar to that of Cheng *et al*. [4], but we additionally include the imitation learning signal in the second stage to mitigate distributional shift.

### C. Dual Policy Iteration

Dual Policy Iteration (DPI) [32] algorithms like Expert Iteration (EI) [1] and AlphaZero [30] have shown impressive performance in solving decision-making problems. This new class of algorithms maintains two policies: a fast learnable policy (*e.g*., a neural network) performs quick rollouts, and a slow policy (*e.g*., a tree-search algorithm [20]) searches the valuable states and plans multiple steps ahead. Those two policies are combined to provide superior demonstrations for training the fast policy, while the updated fast policy in return enhances the performance of the combination. In Expert Iteration, the strong combined policy provides demonstrations to supervise the learnable policy through Imitation Learning (IL). Different from Expert Iteration, our approach trains the policy through Reinforcement Learning, which improves the generalizability of the learnable policy and reduces the learning objective bias caused by mimicking actions of a sub-optimal expert policy.

## III. Preliminaries

### A. Markov Decision Proces

Sequential decision-making procedure is often considered in the Markov Decision Proces (MDP). A discounted infinite-horizon MDP is defined as a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{D}, \mathcal{R}, \gamma)$ [25], where $\mathcal{S}$ is the space of *states*, $\mathcal{A}$ is the space of *actions*; $\mathcal{P}$ is the *transition function*: $\mathcal{P}(s'|s,a)$ is the probability of transforming $s$ to $s'$ by taking action $a$, $s'$ is also written as $\mathcal{P}(s,a)$; $\mathcal{D}$ is the initial state distribution; $\mathcal{R}$ is the *reward function*: $\mathcal{R}(s,a)$ represents the reward received from the environment by taking action $a$ at the state $s$; and $\gamma$ is the discount factor. A distribution over the valid actions $a$ given the state $s$ is called a *policy*, denoted as $\pi(a|s)$. The *trajectory* is a sequence of states and actions of length

$H$, denoted as $\tau = (s_0, a_0, \cdots, s_H, a_H)$. Given a MDP and policy $\pi$, the *trajectory distribution* $p_\pi$ can be derived as $p_\pi(s_0) = \mathcal{D}(s_0)\prod_{t=0}^{H}\pi(a_t|s_t)\mathcal{P}(s_{t+1}|s_t, a_t)$. The value function $V^\pi(s)$ is the expectation of accumulated discounted reward by following $\pi$ starting in state $s$, *i.e.*,

$$V^\pi(s) = \mathbb{E}_{\tau\sim p_\pi(s)}\left[\sum_{t=0}^{H}\gamma^t\mathcal{R}(s_t, a_t)\right]. \tag{1}$$

The optimal policy $\pi^*$ ought to maximize this expectation. Formally, we have $\pi^* = \arg\max_\pi V^\pi(s)$.

### B. Monte Carlo Tree Search

Monte Carlo Tree Search (MCTS) [12, 20] is a useful strategy to address the challenge of selecting/learning policies for MDP. Each node in the search tree corresponds to a state $s$. The root node represents the current state. The edge from $s_1$ to $s_2$ represents the execution of action $a$ by which $s_1$ can be transformed to $s_2$.

In MCTS, four phases are repeated to grow the search tree: a) selection, b) expansion, c) rollout, and d) back-propagation. In the selection phase, a feasible unexpanded edge in the tree is selected according to a *tree policy*. The commonly used tree policy is the Upper Confidence bounds for Trees (UCT) [2], *i.e.*,

$$\mathrm{UCT}(s, a) = \frac{r(s, a)}{n(s, a)} + C\sqrt{\frac{\log n(s)}{n(s, a)}}, \tag{2}$$

where $r(s, a)$ is a prior function that gives suggestions on search. $n(s, a)$ is a counter function that returns the visiting times of this edge; $n(s)$ is the counter function for the node corresponding to state $s$; $C$ is the trade-off parameter to balance the influence of the terms. After a new node with state $s'$ is expanded from the selected edge, in the rollout phase, a quick simulation under the *rollout policy* is performed from the state $s'$ until a terminal condition is satisfied. The results of the simulation are used as the estimate of the node. Finally, the information of nodes is updated bottom-up through the tree until it reaches the root node. When the tree is built, $a^* = \arg\max_a E(s_{\mathrm{R}}, a)$ is the action selected by MCTS, where $s_{\mathrm{R}}$ is the state of the root node and $E(s, a) = \mathcal{R}(s, a) + \gamma V(\mathcal{P}(s, a))$.

## IV. Problem Definition

We formulate SRP as an MDP, where the goal is to transform the current layout of the scene to the target layout. An RL-based agent can achieve this goal by learning a policy to make decisions, aiming to maximize the accumulated discounted reward during the planning.

### A. State Representation

The state in SRP is defined by the scene layout, which includes the shapes, positions, and orientations of the movable objects and the impassable districts, *e.g.*, walls. Following [36], we use the top-down view to characterize the scene layout, as most movable objects rest on the floor plane. The objects and impassable districts are projected onto this floor plane, and the projection is bounded within a square and discretized into a $N \times N$ grid. Suppose a scene has $K$ objects; we use a binary tensor $l, l \in \{0, 1\}^{N\times N\times(K+1)}$ to represent the scene layout. The first two dimensions of the matrix $l$ are the spatial dimensions, and the third dimension is the object dimension. For the $k^{th}$ object, an entry $l_{i,j,k}$ stores whether the location $(i, j)$ is occupied by the object $k$. The third dimension's last channel indicates the occupancy of impassable districts. The state $s_t$ at time step $t$ is the concatenation of the current scene layout $l_t$ and the target scene layout $l_T$ along the third dimension[1], *i.e.*, $s_t \in \{0, 1\}^{N\times N\times(2K+1)}$.

### B. Action Space

The action for SRP is defined as a pair $(o, p)$, where $o \in O$ represents an object and $p \in P$ denotes the possible actions that can manipulate the object. In robotics, more atomic action spaces are preferred due to their tractability (*e.g.*, nonprehensile actions), making them more practical for real-world deployment. In line with this, we define six simple nonprehensile actions to transform the positions and orientations of the objects. Objects can move on cell forward or backward along **up**, **down**, **left** or **right** direction and rotate clockwise or anti-clockwise by 15 degrees (*i.e.*, 24 discrete orientations). This setup is reasonable for smart home scenarios, where furniture with mobile bases can act as intelligent agents rather than relying on external robots for object movement.

### C. Reward Shaping

When an action is executed, the environment returns a reward value immediately. Drawn inspiration from Wang *et al*. [36], our reward shaping consisting four kinds of reward terms. *Distance* reflects the change in distance after an action, which accounts for both the Manhattan distance between the current and target coordinates and the discretized distance between the orientation states. *Arrival* returns a reward of 4 if the current object arrives at its target state after an action. Conversely, *Leave* returns $-4$ if the current object leaves its target state after an action. *Success* offers a significant positive reward of 50 if all objects reach their target states.

## V. Method

To solve the challenging problem, we propose a novel *ex**P**ert-assisted curriculum **L**earning* (**PL**) paradigm trained on demonstrations collected by an expert and an ***A**daptive **T**rade-**O**ff* (**ATO**) strategy to improve the tree-search algorithm. Together, they constitute a powerful agent for SRP.

### A. Expert Demonstration

To perform the learning of the two curriculums paradigm, we first collect a set of trajectory $\mathcal{T}$ through an expert agent. A trajectory $\tau \in \mathcal{T}$ in the dataset is defined as

$$\tau = (s_0, a_0, v_0, r_0(\cdot), s'_0(\cdot), \cdots, s_H, a_H, v_H, r_H(\cdot), s'_H(\cdot)),$$

[1] $l_t$ and $l_T$ share the same impassable district channel.

where $a_t$ is the expert action at time step $t$, $v_t$ is the estimate value of $s_t$ by the expert. $r_t(a)$ is the reward function that returns the reward given $s_t$ and an arbitrary action $a$. $s'_t(a)$ is the transition function that returns the next state given $s_t$ and an arbitrary action $a$. In our problem, $r_t(\cdot)$ and $s'_t(\cdot)$ can be stored in lookup tables since our action space is discrete, and the environment model is available during data collection. They can also be estimated through function approximation [37] and predictive models [6].

The expert we use to collect data is a canonical MCTS algorithm, which predicts both the action and the value when given a state. Note that MCTS is a model-based search algorithm that needs to know the explicit transitions between states. When the model is unavailable, the expert action can be obtained through manual annotation, and the accumulated reward along the trajectory can estimate the value.

### B. Expert Behavior Cloning

In the first curriculum, the agent is taught to learn from the expert. The learning objective is twofold: 1) imitate the expert action, and 2) learn to predict the expert value. To this end, we adopt an actor network $\pi_\theta$ and a value network $v_\phi$ parameterized by $\theta$ and $\phi$, respectively, and train them under expert supervision. Concretely, we use the cross-entropy loss for learning of the actor,

$$\mathcal{L}_p^{\text{IL}} = -\sum_{\tau\in\mathcal{T}}\sum_{t=0}^{H}\log\pi_\theta(a_t|s_t). \tag{3}$$

The learning of the value network is a regression problem where we adopt the Mean-Square-Error (MSE) as the loss function, derived as

$$\mathcal{L}_v^{\text{IL}} = \sum_{\tau\in\mathcal{T}}\sum_{t=0}^{H}(v_\phi(s_t)-v_t)^2. \tag{4}$$

The learning goal of imitating expert action is to approximate the distribution of action sequences in the demonstration dataset given the observed state sequence. According to behavior cloning error bound [26], the expected error of the learned policy in inference is bounded by $C + H^2\epsilon$, where $C$ is the error of expert policy, $H$ is the sequence length, and $\epsilon$ is the generalization error of the learned policy on the training set. In other words, behavior cloning constrains the distribution shift between the training sequences and sequences inferred by the learned policy.

### C. Offline Reinforcement Learning on Expert Demonstration

In contrast to simply imitating expert actions in behavior cloning, the learning objective of RL is finding the best policy to maximize the accumulated discounted reward. Nevertheless, ordinary online RL algorithms often suffer from slow convergence, high variance, and local minima for multiple reasons, one of which is that the agent often explores the sparse-reward regions of the state space. To mitigate those defects, we incorporate the expert demonstration into the RL procedure. The expert demonstration privileges the agent to experience more non-trivial states compared to the data collected through $\epsilon$-greedy strategy by a weak policy, *i.e.*, it illustrates high-reward regions of the state space, which significantly accelerates the convergence. Additionally, these expert demonstrations typically exhibit lower variance, leading to more stable training.

The learning of the policy network is similar to Off-Policy Actor-Critic (Off-PAC) [5], an off-policy RL algorithm. In this way, the loss of the policy network is

$$\mathcal{L}_p^{\text{RL}} = -\sum_{\tau}^{\mathcal{T}}\sum_{t=0}^{H}\mathbb{E}_{a\sim\pi_\theta(\cdot|s_t)}\left[A(s_t,a)\log(\pi_\theta(a|s_t))\right], \tag{5}$$

where $A(s_t,a)$ is the advantage defined as $A(s_t,a) = r_t(a) + \gamma v_\phi(s'_t(a)) - v_\phi(s_t)$.

Conventional AC algorithm updates the critic in a Temporal-Difference (TD) manner. Nevertheless, in the offline setting, TD makes the critic especially prone to suffer from deadly triad [34] due to the out-of-distribution (OOD) issue. Therefore, we use the weights pre-trained in behavior cloning and fix them in offline RL training. The critic is regarded as a baseline function to reduce the variance during training. Since offline RL training of the policy network may increase the defect of distributional shift, we also add the behavior cloning signal in the second curriculum to constrain the update of the actor.

### D. Adaptive Trade-Off Strategy

Due to the vast action space and the exponential growth of search complexity, the spread and depth of the search tree are highly constrained in limited search times, which can make a general MCTS algorithm inefficient. To tackle this problem, we utilize the intrinsic character of our task and propose an **ATO** strategy and significantly improve the search efficiency.

For SRP, it is intuitive that multiple optimal actions are feasible in some states while few optimal actions exist in others. For example, considering a scene where most objects need to be moved, even an optimal policy may choose any one of the objects to move toward its target configuration. Conversely, in a scene with only one object requiring movement, the optimal policy would take the only available optimal action. In other words, the entropy of action probability distribution from an optimal policy varies violently across different states. This character reveals that an ideal balance between exploitation and exploration for a tree-search algorithm is dependent on the state, *i.e.*, the search should exploit more in the states with many optimal actions and explore more in the states with few optimal actions. In MCTS, the trade-off between exploitation and exploration is controlled by a hyperparameter $C$ according to the Upper Confidence bounds for Trees (UCT) [2]. As $C$ increases, MCTS tends to explore (spread) rather than exploit (depth). As the above observation suggests, $C$ can be formalized as a function $C(s)$ depending on the state $s$,

$$C(s) = \beta - \lambda\frac{|\{a|\pi^*(a|s) > \xi\}|}{|\mathcal{A}|}, \tag{6}$$

where $\lambda$ is a scaling coefficient, $\beta$ is a bias coefficient, and $\xi$ is the threshold that filters the optimal actions. Eq. (6)

illustrates that $C$ is proportional to the number of optimal actions. However, calculating $C(s)$ is intractable since $\pi^*$ is unavailable, and $\xi$ needs to be tuned. Instead, we introduce a coarse upper bound $d(s, s_T)$ that holds in most states. $d(s, s_T)$ is the element-wise difference of position and orientation between the current state $s$ and the target state $s_T$. In this way, the trade-off $\hat{C}(s)$ can be derived as:

$$\hat{C}(s) = \beta - \lambda \frac{d(s, s_T)}{|\mathcal{A}|}. \tag{7}$$

The **ATO** strategy can be directly applied in the inference phase of MCTS.

### E. Integration of Policy Network and MCTS

Combining neural networks with search algorithms has achieved great success in many tasks. To further enhance the performance in SRP, we integrate our trained policy networks with the MCTS algorithm. Specifically, during the selection phase of MCTS, we use the actor of the policy, $\pi_\theta(a_t|s_t)$, as the prior function $r(s, a)$ to guide the search. In the rollout phase, conducting a quick simulation to reach a terminal state is intractable for complex tasks like SRP. Therefore, we perform a fixed-step simulation and use the critic network to estimate the value of the final state for back-propagation usage. Such a tree-search planner integrated with a pre-trained policy network possesses improved search efficiency and efficacy.

### F. Implementation

**Network Architecture** We modify the network architecture from [36] to build the actor, which is a CNN encoder appended with an LSTM layer. The hidden state size of the LSTM layer is 512. The critic shares the same CNN encoder with the actor and has an individual FC layer. The input of the two networks is the current state $s$. The output of the actor is a 120-$d$ tensor, which represents the predicted probability of actions. It works for at most 20 objects in a scene. The output of the critic is a scalar, which refers to the estimated value of the current state.

**Reproducibility** Our network is implemented in PyTorch. The batch size is 8. The number of the data processes is 4. The rounds of tree-search for each decision-making is 50. In the test phase, an episode is regarded as failed if it is not finished within 200 steps. The optimizer used in the network training is ADAM. The learning rate is set to $10^{-4}$. The trade-off factor $C$ for canonical MCTS is 2.0. The utilities, including the virtual environment and the tree-search algorithm, are implemented in C++. The full model is trained on 4 NVIDIA RTX 3090 GPUs with 24GB memory in each card. The PL algorithm is illustrated in Alg. 1.

**Algorithm 1: Training of PL**

Initialize $\theta_0$ for actor;
Initialize $\phi_0$ for critic;
Collect a set of trajectory $\mathcal{T} = \{\tau\}$ through an expert;
`// Curriculum 1`
**for** iteration $k \in [0, \cdots, K-1]$ **do**
  sample a batch of trajectory $\mathcal{B}$ from $\mathcal{T} = \{\tau\}$;
  compute loss $\mathcal{L}_p^{\text{IL}}(\mathcal{B}, \theta_k)$ according to Eq. (3);
  compute loss $\mathcal{L}_v^{\text{IL}}(\mathcal{B}, \phi_k)$ according to Eq. (4);
  $\theta_{k+1} \leftarrow \theta_k - \alpha_p \nabla_{\theta_k} \mathcal{L}_p^{\text{IL}}$;
  $\phi_{k+1} \leftarrow \phi_k - \alpha_v \nabla_{\phi_k} \mathcal{L}_v^{\text{IL}}$;
**end for**
`// Curriculum 2`
**for** iteration $l \in [0, \cdots, L-1]$ **do**
  sample a batch of trajectory $\mathcal{B}$ from $\mathcal{T} = \{\tau\}$;
  compute loss $\mathcal{L}_p^{\text{IL}}(\mathcal{B}, \theta_{K+l})$ according to Eq. (3);
  compute loss $\mathcal{L}_p^{\text{RL}}(\mathcal{B}, \theta_{K+l})$ according to Eq. (5);
  compute loss $\mathcal{L}_v^{\text{IL}}(\mathcal{B}, \phi_{K+l})$ according to Eq. (4);
  $\theta_{K+l+1} \leftarrow \theta_k - \alpha_{p,\text{IL}} \nabla_{\theta_{K+l}} \mathcal{L}_p^{\text{IL}} - \alpha_{p,\text{RL}} \nabla_{\theta_k} \mathcal{L}_p^{\text{RL}}$;
  $\phi_{K+l+1} \leftarrow \phi_k - \alpha_v \nabla_{\phi_{K+l}} \mathcal{L}_v^{\text{IL}}$;
**end for**

## VI. Experiments

### A. Dataset

We demonstrate our approach on 3D-FRONT [7], a large-scale indoor scene dataset, where the room's layout is professionally designed and populated with high-quality 3D models. The interior designs of the rooms are transferred from expert creations. This dataset contains $6,813$ distinct houses and $19,062$ furnished rooms, supporting indoor scene tasks like 3D scene understanding, indoor scene synthesis, semantic segmentation, *etc*. The diverse furnished rooms and the abundant labels satisfy the demand of the SRP task. In SRP, the complexity of planning depends heavily on object number and object shape in the scene since more objects and shapes constrain the solution space. We filter the dataset and exclude the scenes with less than 4 objects ($46.1\%$) and more than 15 objects ($1.1\%$), removing the trivial cases and reducing the dataset's bias. It finally results in $13,215$ rooms in total. Fig. 2 shows the statistic of object number in these rooms. We randomly divide the dataset into three splits, `Train` (11899 cases, $90\%$), `Validation Unseen` (259 cases, $2\%$) and `Test` (1057 cases, $8\%$). We sample 1000 cases from `Train` split to form `Validation Seen` set.

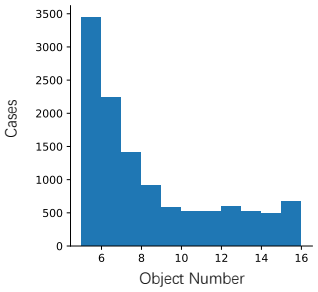


Fig. 2: **Dataset distribution.**

We use the human-designed layout as the target layout and sample a feasible layout with the same objects as the initial layout. To ensure that the initial and target layouts can transform from each other, we sample the initial layout by rounds of random walks starting from the target. In each round, we randomly select an object to perform a random feasible move. Each initial layout is generated by performing 150 rounds of moves using a greedy strategy, making the objects as far as possible from their original positions and poses. We extract the layouts' configurations through the following steps: (i) bounding the view of the scene to the center of a square; (ii) rendering the objects individually; (iii)

TABLE I: **The performance of different agents. SR** is the primary metric. The number in **bold** is the best in comparison.

| | Agent | Validation Seen | | | | | Validation Unseen | | | | | Test | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **SR**↑ | Length↓ | DR↑ | TR↑ | Runtime↓ | **SR**↑ | Length↓ | DR↑ | TR↑ | Runtime↓ | **SR**↑ | Length↓ | DR↑ | TR↑ | Runtime↓ |
| Model-Free | Random | 0 | 200.0 | 3.54 | 4.36 | **0.09s** | 0 | 200.0 | 3.24 | 3.44 | **0.09s** | 0 | 200.0 | 3.49 | 3.90 | **0.09s** |
| | Off-PAC | 0.080 | 192.3 | 17.63 | 65.23 | 0.25s | 0.098 | 191.6 | 17.60 | 70.36 | 0.25s | 0.078 | 193.9 | 17.52 | 65.70 | 0.25s |
| | PPO | 0.306 | 165.6 | **19.54** | 80.28 | 0.33s | 0.314 | 166.3 | **19.48** | 83.28 | 0.33s | 0.307 | 164.8 | **19.48** | 79.58 | 0.33s |
| | EI | 0.236 | 178.1 | 14.37 | 62.87 | 0.24s | 0.206 | 182.5 | 13.40 | 59.80 | 0.24s | 0.187 | 181.7 | 13.25 | 55.97 | 0.25s |
| | PL (**Ours**) | **0.365** | **156.8** | 18.81 | **99.96** | 0.21s | **0.401** | **158.3** | 18.79 | **102.70** | 0.22s | **0.386** | **156.5** | 18.90 | **99.47** | 0.20s |
| Model-Based | Greedy | 0.470 | 152.0 | 21.56 | 54.07 | **1.18s** | 0.470 | 155.3 | 21.53 | 53.19 | **1.19s** | 0.472 | 150.5 | 21.68 | 53.37 | **1.15s** |
| | A* | 0.640 | 131.3 | **22.58** | **139.66** | 178.94s | 0.640 | 131.2 | **22.23** | **138.60** | 162.6s | 0.600 | 140.2 | 21.25 | **142.50** | 169.27s |
| | MCTS | 0.540 | 152.2 | 21.14 | 100.33 | 71.78s | 0.497 | 158.6 | 21.29 | 101.03 | 47.90s | 0.502 | 153.8 | 21.34 | 96.91 | 38.43s |
| | PPO + TS | 0.582 | 138.0 | 21.16 | 103.90 | 16.08s | 0.574 | 139.2 | 21.08 | 104.85 | 16.55s | 0.576 | 137.8 | 20.89 | 104.40 | 16.86s |
| | **PLATO** (**Ours**) | **0.747** | **128.1** | 22.24 | 125.37 | 16.22s | **0.768** | **129.7** | 22.00 | 133.18 | 17.74s | **0.731** | **129.8** | **21.60** | 125.60 | 16.55s |

discretizing the top-down silhouettes to extract the shape and position of each object. We adopt the same process to extract the information of impassable districts. The discretization resolution is $64 \times 64$.

### B. Quantitative Results

**Metrics** To evaluate the agent, we define five metrics, *i.e.*, *Success Rate* (SR) ↑, *Length* ↓, *Discounted Reward* (DR) ↑, *Total Reward* (TR) ↑, and *Runtime* ↓. Up arrow ↑ indicates higher value is better; ↓ is similar. SR is the rate of attaining the target layout. It illustrates the agent's general ability to finish the task and is the primary metric in our experiment. Length refers to the average length of the action sequence. It reflects the efficiency of the action sequence. For the failed cases, the length of the action sequence is the maximum allowed action step (200 in evaluation). DR and TR reveal the agent's soft short-term and long-term performance, respectively. Runtime is the average time consumed in each case. It measures the algorithm's running efficiency.

**Baseline Agents** To have an intuitive understanding of the complexity of SRP and provide a comparison with our approach, we introduce several strong learning-based, rule-based, and search methods as baseline agents:

- *Random*: The agent that follows the random walk policy, where the action is randomly sampled at each step.
- *Greedy*: The agent takes action with the maximum instant reward at each step.
- *A**: The agent performing the A* algorithm to search the action. We use the distance between the current state and the target state as the heuristic estimate. See the definition of distance between states in reward shaping.
- *MCTS*: The agent that performs the canonical Monte Carlo Tree Search algorithm [2] to search the action.
- *Off-PAC*: The agent trained with Off-Policy Actor-Critic algorithm [5]. $\epsilon$-greedy strategy is adopted.
- *PPO*: The agent trained with the Proximal Policy Optimization (PPO) algorithm [28].
- *EI*: A policy trained with Expert Iteration (EI) [1].

Note that *Greedy*, *A**, and *MCTS* are model-based agents that rely on simulation, while the learning-based agents are model-free in inference. For a fair comparison, the networks used in our approach, Off-PAC, PPO, and EI, share the same architecture. The trained network integrated with a tree-search algorithm can build a stronger model-based agent. We take PPO as an example, which is denoted as PPO + TS.

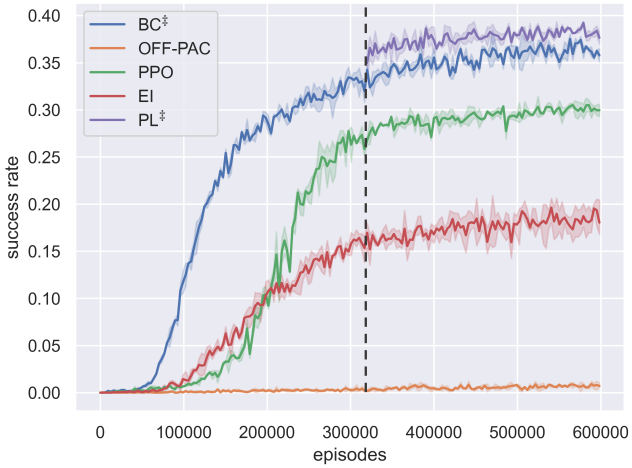



Fig. 3: **The training curve of different agents.** Note that BC and **PL** share the same training process before the dashed line. Notation ‡ denotes that the training is offline.

**Comparison** The performance of the agents on `Validation Seen`, `Validation Unseen` and `Test` sets are shown in Tab. I. The weights of the learnable agents are selected via SR metric on `Validation Unseen` set. The first five rows are model-free agents. The result shows that the Random agent gets 0 SR on all splits, which means that SRP is not trivial. Our approach achieves better SR, Length, and TR than the Off-PAC, PPO, and EI methods across all splits. The following five rows are model-based agents. We can observe that the agent built by our approach also achieves the best performance on SR. It is worth noting that model-free agents require less runtime compared to model-based agents because they do not rely on simulation.

To analyze the training characteristics of the agents, we plot the curve of SR during training. As shown in Fig. 3, the x-axis represents the number of episodes the agent experienced, and the y-axis indicates the SR metric. The curve shows that our approach converges rapidly in the first curriculum and gradually reaches a plateau. In the second curriculum, the agent soon transcends this plateau, which indicates that the learning objective of RL effectively updates the policy network toward a better policy.

### C. Diagnostic Experiment and Analysis

**Problem Complexity** The complexity of planning is partially influenced by the number of objects. To further study the performance of the learnable agents under different levels of problem difficulty, we divide the cases in `Test` set into

TABLE II: **The performance on different object number splits.**

| Agent | 4-7 objects | | | | | 8-11 objects | | | | | 12-15 objects | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **SR**↑ | Length↓ | DR↑ | TR↑ | Runtime↓ | **SR**↑ | Length↓ | DR↑ | TR↑ | Runtime↓ | **SR**↑ | Length↓ | DR↑ | TR↑ | Runtime↓ |
| Off-PAC | 0.117 | 190.9 | 18.01 | 68.88 | 0.25s | 0 | 200.0 | 17.94 | 73.36 | 0.25s | 0 | 200.0 | 14.63 | 40.18 | 0.25s |
| PPO | 0.456 | 147.6 | 20.11 | 84.26 | 0.33s | 0.002 | 199.9 | 18.97 | 79.50 | 0.33s | 0 | 200.0 | 17.16 | 57.16 | 0.33s |
| EI | 0.278 | 172.8 | 15.68 | 70.82 | 0.24s | 0 | 200.0 | 9.68 | 35.24 | 0.24s | 0 | 200.0 | 6.32 | 12.39 | 0.24s |
| **PL** | 0.573 | 135.3 | 20.72 | 111.72 | 0.20s | 0.005 | 199.7 | 15.94 | 84.91 | 0.20s | 0 | 200.0 | 14.09 | 60.13 | 0.20s |

TABLE III: **Ablation study.** Notation ‡ denotes that the training is offline. Notation † indicates that **ATO** is adopted in tree-search.

| Agent | Test | | | | |
|---|---|---|---|---|---|
| | **SR**↑ | Length↓ | DR↑ | TR↑ | Runtime↓ |
| BC‡ | 0.361 | 159.9 | 18.73 | 98.53 | 0.22s |
| Off-PAC‡ | 0 | 200.0 | 3.80 | -3.47 | 0.29s |
| **PL**‡ | 0.386 | 156.5 | 18.90 | 99.47 | 0.20s |
| MCTS | 0.502 | 153.8 | 21.34 | 96.91 | 38.43s |
| MCTS† | 0.513 | 153.8 | 21.26 | 101.11 | 59.67s |
| PPO + TS | 0.576 | 137.8 | 20.89 | 104.40 | 16.86s |
| PPO + TS† | 0.601 | 138.1 | 20.50 | 110.00 | 20.73s |
| **PL** + TS ($C = 0.05$) | 0.716 | 132.0 | 21.57 | 120.94 | 21.05s |
| **PL** + TS ($C = 0.10$) | 0.726 | 131.7 | 21.65 | 125.60 | 14.21s |
| **PL** + TS ($C = 0.15$) | 0.688 | 133.8 | 21.57 | 120.86 | 13.91s |
| **PL** + TS† (**PLATO**) | 0.731 | 129.8 | 21.60 | 125.60 | 16.55s |

three splits according to the object number in each room. In our dataset, object numbers in a room range from 4 to 15; the first split contains cases with 4–7 objects, the second split has cases with 8–11 objects, and the third split has cases of 12–15 objects.

The comparison of different agents with respect to the object number is presented in Tab. II. We observe a significant drop in SR as the object number increases. In particular, the learnable agents achieve nearly 0 SR with 8–11 objects and 12–15 objects. It is important to note that each test case has an optimal solution within 150 steps, which the initial layout generation process guarantees. The evaluation results support our hypothesis that scenarios with more objects impose more constraints, making the problem more challenging. Such a poor performance on the two splits also highlights that this problem is far from being fully solved.

**Curriculums** We study the effects of two curriculums by training agents with only Behavior Cloning (BC) and Off-PAC, respectively. As shown in Tab. III, omitting either curriculum leads to a performance drop. When trained solely with Off-PAC, the agent's performance significantly worsens due to the distributional shift. Fig. 3 further highlights the importance of both curriculums. Interestingly, we observe that the agent trained with EI converges rather slowly compared to BC. This can be attributed to the iterative updates of the expert policy in EI, where the agent attempts to imitate actions from different policies, causing inconsistent learning objectives and a drop in performance.

**Generalizability of ATO** We also study the effectiveness of **ATO** on different search agents. As shown in Tab. III, **ATO** consistently boosts the performance of agents consisting of a network policy and a tree-search planner. Recall that **ATO** controls the balance between exploitation and exploration. Notably, in comparison to the agents **PL** + TS with different fixed trade-off factor $C$, the agent with our **ATO** strategy outperforms all other agents without hyperparameters searching.

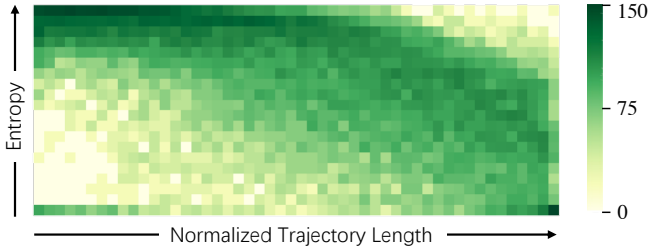



Fig. 4: **Frequency map of action entropy over 259 episodes.**

**Rationality of ATO** To further reveal the rationality of **ATO**, we visualize the statistics of action entropy throughout the episodes. Here, we use MCTS to approximate the optimal policy[2] and record the action entropy along the inferred trajectories on Validation Unseen set. The action probability is derived by computing softmax over $E(s_R, a)$. As illustrated in Fig. 4, at the beginning of an episode, the action entropy of the optimal policy is high, indicating that several actions are feasible. As the episode proceeds, the entropy gradually decreases, reflecting that fewer actions remain optimal. It suggests that the search can rely more on the predicted prior from the network policy (exploitation) when the state has many feasible actions to move, as it has a higher probability of choosing a feasible action. Conversely, the search should try more possibilities (exploration) when the state has few feasible actions. This is exactly the core idea of the proposed **ATO** strategy.

## VII. Conclusion and Future Work

This paper focuses on the SRP task: introducing a more flexible definition of atomic action space, curating a large-scale dataset for benchmarking, and proposing a novel model named **PLATO** to address this challenge. **PLATO** features a curriculum learning paradigm for efficient agent training and a strategy that enhances the efficiency and efficacy of tree-search-based planners. Extensive results and analysis demonstrate the superiority of the proposed methods over various baselines. Additionally, results across different difficulty levels on the test set indicate that the problem remains unsolved as scene layout complexity increases.

Future work should consider the 3D geometry of objects and the involvement of robots for object manipulation in 3D space, making the approach more suitable for real-world deployment. Furthermore, exploring multi-agent cooperation in SRP task is essential, especially when it requires the manipulation of large and heavy objects.

**Acknowledgement** This work is supported by the National Natural Science Foundation of China (NSFC) (62172043).

[2]MCTS is a sound choice because its policy converges toward the optimal policy as the number of search rounds increases.

## REFERENCES


[1] T. Anthony, Z. Tian, and D. Barber, "Thinking fast and slow with deep learning and tree search," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. 2, 6

[2] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine Learning*, vol. 47, pp. 235–256, 2002. 3, 4, 6

[3] D. Batra, A. X. Chang, S. Chernova, A. J. Davison, J. Deng, V. Koltun, S. Levine, J. Malik, I. Mordatch, R. Mottaghi, *et al.*, "Rearrangement: A challenge for embodied ai," *arXiv preprint arXiv:2011.01975*, 2020. 1

[4] C.-A. Cheng, X. Yan, N. Wagener, and B. Boots, "Fast policy learning through imitation and reinforcement," *arXiv preprint arXiv:1805.10413*, 2018. 2

[5] T. Degris, M. White, and R. S. Sutton, "Off-policy actor-critic," in *International Conference on Machine Learning (ICML)*, 2012. 4, 6

[6] F. Ebert, C. Finn, S. Dasari, A. Xie, A. Lee, and S. Levine, "Visual foresight: Model-based deep reinforcement learning for vision-based robotic control," *arXiv preprint arXiv:1812.00568*, 2018. 2, 4

[7] H. Fu, B. Cai, L. Gao, L.-X. Zhang, J. Wang, C. Li, Q. Zeng, C. Sun, R. Jia, B. Zhao, *et al.*, "3d-front: 3d furnished rooms with layouts and semantics," in *International Conference on Computer Vision (ICCV)*, 2021. 1, 5

[8] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine, "D4rl: Datasets for deep data-driven reinforcement learning," *arXiv preprint arXiv:2004.07219*, 2020. 2

[9] K. Gao, D. Lau, B. Huang, K. E. Bekris, and J. Yu, "Fast high-quality tabletop rearrangement in bounded workspace," in *International Conference on Robotics and Automation (ICRA)*, 2022. 1, 2

[10] K. Gao, J. Yu, T. S. Punjabi, and J. Yu, "Effectively rearranging heterogeneous objects on cluttered tabletops," in *International Conference on Intelligent Robots and Systems (IROS)*, 2023. 1, 2

[11] A. Goyal, A. Mousavian, C. Paxton, Y.-W. Chao, B. Okorn, J. Deng, and D. Fox, "Ifor: Iterative flow minimization for robotic object rearrangement," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 1, 2

[12] X. Guo, S. Singh, H. Lee, R. L. Lewis, and X. Wang, "Deep learning for real-time atari game play using offline monte-carlo tree search planning," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2014. 3

[13] J. A. Haustein, I. Arnekvist, J. Stork, K. Hang, and D. Kragic, "Learning manipulation states and actions for efficient non-prehensile rearrangement planning," *arXiv preprint arXiv:1901.03557*, 2019. 2

[14] R. Hu, J. Xu, B. Chen, M. Gong, H. Zhang, and H. Huang, "Tap-net: transport-and-pack using reinforcement learning," *ACM Transactions on Graphics (TOG)*, vol. 39, no. 6, pp. 1–15, 2020. 1

[15] B. Huang, S. D. Han, J. Yu, and A. Boularias, "Visual foresight trees for object retrieval from clutter with nonprehensile rearrangement," *IEEE Robotics and Automation Letters (RA-L)*, vol. 7, no. 1, pp. 231–238, 2021. 1, 2

[16] S. Huang, Z. Wang, J. Zhou, and J. Lu, "Planning irregular object packing via hierarchical reinforcement learning," *IEEE Robotics and Automation Letters (RA-L)*, vol. 8, no. 1, pp. 81–88, 2022. 1

[17] N. Jaques, A. Ghandeharioun, J. H. Shen, C. Ferguson, A. Lapedriza, N. Jones, S. Gu, and R. Picard, "Way off-policy batch deep reinforcement learning of implicit human preferences in dialog," *arXiv preprint arXiv:1907.00456*, 2019. 2

[18] N. Jiang and L. Li, "Doubly robust off-policy value evaluation for reinforcement learning," in *International Conference on Machine Learning (ICML)*, 2016. 2

[19] J. E. King, M. Cognetti, and S. S. Srinivasa, "Rearrangement planning using object-centric and robot-centric action spaces," in *International Conference on Robotics and Automation (ICRA)*, 2016. 1, 2

[20] L. Kocsis and C. Szepesvári, "Bandit based monte-carlo planning," in *European Conference on Machine Learning (ECML)*, 2006. 2, 3

[21] M. C. Koval, J. E. King, N. S. Pollard, and S. S. Srinivasa, "Robust trajectory selection for rearrangement planning as a multi-armed bandit problem," in *International Conference on Intelligent Robots and Systems (IROS)*, 2015. 1, 2

[22] Y. Labbé, S. Zagoruyko, I. Kalevatykh, I. Laptev, J. Carpentier, M. Aubry, and J. Sivic, "Monte-carlo tree search for efficient visually guided rearrangement planning," *IEEE Robotics and Automation Letters (RA-L)*, vol. 5, no. 2, pp. 3715–3722, 2020. 1, 2

[23] S. Narvekar, B. Peng, M. Leonetti, J. Sinapov, M. E. Taylor, and P. Stone, "Curriculum learning for reinforcement learning domains: A framework and survey," *Journal of Machine Learning Research*, vol. 21, no. 181, pp. 1–50, 2020. 2

[24] D. Paschalidou, A. Kar, M. Shugrina, K. Kreis, A. Geiger, and S. Fidler, "Atiss: Autoregressive transformers for indoor scene synthesis," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. 1

[25] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014. 2

[26] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *International Conference on Artificial Intelligence and Statistics (ICAIS)*, 2011. 4

[27] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International Conference on Machine Learning (ICML)*, 2015. 2

[28] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017. 6

[29] K. Sham and L. John, "Approximately optimal approximate reinforcement learning," in *International Conference on Machine Learning (ICML)*, 2002. 2

[30] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, *et al.*, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, 2017. 2

[31] C. Song and A. Boularias, "Object rearrangement with nested nonprehensile manipulation actions," in *International Conference on Intelligent Robots and Systems (IROS)*, 2019. 1, 2

[32] W. Sun, G. J. Gordon, B. Boots, and J. Bagnell, "Dual policy iteration," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. 2

[33] P. Thomas and E. Brunskill, "Data-efficient off-policy policy evaluation for reinforcement learning," in *International Conference on Machine Learning (ICML)*, 2016. 2

[34] H. Van Hasselt, Y. Doron, F. Strub, M. Hessel, N. Sonnerat, and J. Modayil, "Deep reinforcement learning and the deadly triad," *arXiv preprint arXiv:1812.02648*, 2018. 4

[35] E. R. Vieira, D. Nakhimovich, K. Gao, R. Wang, J. Yu, and K. E. Bekris, "Persistent homology for effective non-prehensile manipulation," in *International Conference on Robotics and Automation (ICRA)*, 2022. 1, 2

[36] H. Wang, W. Liang, and L.-F. Yu, "Scene mover: Automatic move planning for scene arrangement by deep reinforcement learning," *ACM Transactions on Graphics (TOG)*, vol. 39, no. 6, pp. 1–15, 2020. 1, 2, 3, 5

[37] X. Wang, Q. Huang, A. Celikyilmaz, J. Gao, D. Shen, Y.-F. Wang, W. Y. Wang, and L. Zhang, "Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 4

[38] L. Weihs, M. Deitke, A. Kembhavi, and R. Mottaghi, "Visual room rearrangement," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 1, 2

[39] G. Xiong, Q. Fu, H. Fu, B. Zhou, G. Luo, and Z. Deng, "Motion planning for convertible indoor scene layout design," *IEEE Transactions on Visualization and Computer Graph (TVCG)*, vol. 27, no. 12, pp. 4413–4424, 2020. 1

[40] L. F. Yu, S. K. Yeung, C. K. Tang, D. Terzopoulos, T. F. Chan, and S. J. Osher, "Make it home: automatic optimization of furniture arrangement," *ACM Transactions on Graphics (TOG)*, vol. 30, no. 4, 2011. 1

[41] W. Yuan, K. Hang, D. Kragic, M. Y. Wang, and J. A. Stork, "End-to-end nonprehensile rearrangement with deep reinforcement learning and simulation-to-reality transfer," *Robotics and Autonomous Systems*, vol. 119, pp. 119–134, 2019. 1, 2

[42] G. Zhai, X. Cai, D. Huang, Y. Di, F. Manhardt, F. Tombari, N. Navab, and B. Busam, "Sg-bot: Object rearrangement via coarse-to-fine robotic imagination on scene graphs," in *International Conference on Robotics and Automation (ICRA)*, 2024. 1, 2

[43] Z. Zhang, Z. Yang, C. Ma, L. Luo, A. Huth, E. Vouga, and Q. Huang, "Deep generative modeling for scene synthesis via hybrid representations," *ACM Transactions on Graphics (TOG)*, vol. 39, no. 2, pp. 1–21, 2020. 1