2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

Supportive Relationships-aware Hierarchical Reinforcement Learning for Efficient Ex-situ Object Rearrangement

Leibing Xiao[1], Xuemei Wang[1], Zhongqiang Zhao[1], Yachao Wang[1], Jin Liu[2], and Chaoqun Wang[1,*]

***Abstract*— In ex-situ object rearrangement tasks within open environments, robots face significant challenges due to the increased cost of moving objects over large workspaces. To address this issue, we propose a hierarchical reinforcement learning-based approach that takes into account the supportive relationships and semantic correlations between objects. The robot groups and stacks objects with compatible supportive capabilities, moving them together to their target locations to optimize task execution. Specifically, we use a large language model to assess the supportive relationships and semantic correlations between objects. In the high-level decision-making process, objects are grouped based on their supportive capabilities, while the low-level process refines these groupings using a graph capsule convolutional network. Experimental results demonstrate that our approach not only reduces the number of movements required but also improves task efficiency and significantly decreases task completion time by approximately 50%, compared to methods that do not consider supportive relationships.**

I. Introduction

Object rearrangement is a challenging task for robots, both in everyday environments [1]–[3] and industrial settings [4]–[6]. For example, a robot responsible for organizing items needs to rearrange scattered tableware from the tabletop and neatly place it at a designated location, as shown in Figure 1. To complete this task, the robot must not only transport the tableware to the target area but also arrange it systematically in the correct positions. Robots must possess capabilities in visual perception [7], semantic understanding [8], motion planning [9], and manipulation control [10], etc. Nowadays, most existing methods for object rearrangement rely on visual perception, typically using cameras or depth sensors to gather object information. These methods can be categorized into three main types: physics-based, pose estimation-based, and large model-based approaches [11]. Physics-based methods rely on a geometric model of the object and positional data to perform the rearrangement [6], [12]. Pose estimation-based methods detect the object's pose for rearrangement, eliminating the need for a geometric model but requiring precise pose information [13]. Large model-based methods leverage large vision models to facilitate rearrangement [14]–[16]. Although significant progress has been made, most research has focused on closed scenarios, such as desktop or box environments. By ignoring the robot's movement, this setting significantly simplifies the task but also restricts the applicability of these methods in more complex real-world environments [1].

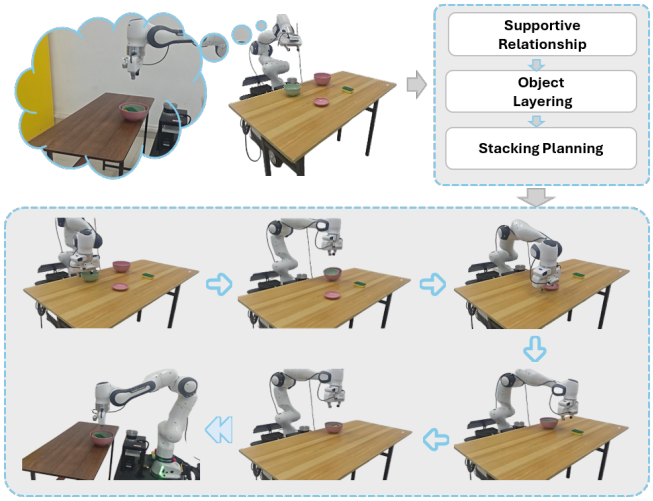

Fig. 1. An example of ex-situ multi-object rearrangement in an open environment. Because the objects' initial and goal locations are far apart, the robot first stacks compatible items according to their supportive relationships, then transports the stack to the target location, and finally places the items.

To enhance the robot's applicability in real-world environments, many researchers have investigated ex-situ object rearrangement in open settings, such as the living room or the kitchen [1], [2]. In these settings, the robot first uses prior environmental knowledge to locate the object. It then performs the grasping operation using perception and control modules, moves the object to the target position, and places it. Most research focuses on control planning to achieve the rearrangement goal, while some studies train the robot to autonomously determine the target state and complete the task. However, these studies often overlook the robot's decision-making process during intermediate steps. This results in inefficiency when the robot processes multiple objects, particularly in ex-situ object rearrangement tasks. Existing object rearrangement research typically involves the robot handling each object individually, which is inefficient, especially in open environments. As the robot's workspace expands in such a setting, the cost of moving an object from its initial position to the target location increases. Therefore, improving the efficiency of object rearrangement in open environments remains a significant challenge.

To improve the robot's efficiency in ex-situ object rearrangement tasks, we propose a hierarchical reinforcement learning (HRL) based framework that optimizes both

[1]School of Control Science and Engineering, Shandong University, Jinan, China.
[2]School of Computer Science and Technology / School of Artificial Intelligence, China University of Mining and Technology, Xuzhou, China.
*Corresponding author (e-mail: chaoqunwang@sdu.edu.cn).

decision-making and execution. HRL aims to decompose a complex problem into smaller sub-problems at different levels, addressing high-level planning and low-level execution through the training of neural networks [17]–[19]. One of the main advantages of HRL is its ability to reduce training complexity and improve efficiency. HRL has been widely applied to various tasks [20]–[23].

The method first inputs the object information into a large model, such as ChatGPT [24], to determine the supportive relationships and semantic correlations between objects. Then, the robot feeds the supportive relationships into the high-level decision process of HRL to group the target objects based on their supportive capabilities. In the low-level decision process, the robot inputs the results from the high-level decision, along with semantic and supportive relationship information, into a model based on Graph Capsule Convolutional Networks (GCCNs) to determine the final grouping. Finally, the robot performs the actual rearrangement tasks. This approach enables the robot to optimize object placement while minimizing execution time and unnecessary movement. The main contributions of this paper are as follows:

- We propose a comprehensive framework to improve the efficiency of multi-object rearrangement in open environments, allowing the robot to transport multiple objects by stacking them, thereby enhancing operational efficiency.
- We incorporate both the supportive relationships and semantic information between objects, utilizing these insights to significantly enhance the robot's decision intelligence.
- Experiments show that our approach effectively enhances the robot's efficiency and reduces task completion time in both simulation and real-world scenarios.

II. Methodology

In this section, we present the details of the HRL framework. The details are illustrated in Figure 2.

A. Problem Statement

We consider an object rearrangement scenario in a cluttered environment. A set of m objects, such as plates, cups, lids, and other items, must be transported by the robot to a designated cabinet located on the opposite side of the table. In this scenario, the robot first acquires information about the objects on the countertop, including their category, position, shape, and other relevant attributes. The robot must then determine how to group the objects and decide the order in which they should be moved. The objective is to minimize the robot's task execution time and improve its overall efficiency in completing the rearrangement task.

B. Modeling Supportive Relationships and Semantic Relevance

In cluttered environments, objects often exhibit complex supportive dependencies. To ensure safety and efficiency in rearrangement tasks, object relationships are modeled from two complementary perspectives: supportive relationships and semantic relevance.

We first introduce the support matrix $\mathbb{S} \in \{0,1\}^{m\times m}$, where m denotes the total number of objects in the scene. Its elements are defined as

$$\mathbb{S}_{ij} = \begin{cases} 1, & \text{if object } i \text{ supports object } j, \\ 0, & \text{otherwise,} \end{cases} \qquad \mathbb{S}_{ii} = 0. \tag{1}$$

Supportive relationships are assumed to be unidirectional, i.e., $\mathbb{S}_{ij} = 1 \not\Rightarrow \mathbb{S}_{ji} = 1$. This assumption eliminates ambiguity due to mutual dependence and guarantees the structural stability of stacks.

Beyond supportive relationships, semantic relevance is also considered, e.g., the association between a cup and its lid. To formalize this, let $\mathbb{N} = \{n_i\}_{i=1}^{m}$ denote the set of objects, and $\mathbb{E} \subseteq \mathbb{N} \times \mathbb{N}$ denote the set of candidate object pairs. The candidate relation set $\Gamma = \{\texttt{above}, \texttt{below}\}$ focuses on vertical relations. The mapping $\phi : \mathbb{E} \to \Gamma$ assigns specific semantic labels to candidate pairs. Finally, a LLM is employed to select the most appropriate relation from Γ, thus extracting semantically meaningful object pairs and providing structured priors for downstream reinforcement learning policies.

C. Object Layering Based on Supportive Capability

Based on the supportive relationship matrix introduced in Section II-B, we transform the sparse matrix $\mathbb{S} \in \{0,1\}^{m\times m}$ into a Directed Acyclic Graph (DAG) to enable hierarchical reasoning for subsequent tasks.

The DAG is denoted as $\mathbb{G} = (\mathbb{N}, \hat{\mathbb{E}})$, where $\mathbb{N}$ is the set of object nodes and $\hat{\mathbb{E}}$ is the set of selected edges representing supportive relations. The conversion process consists of two stages: (*i*) **node layering**, which assigns objects to layers by iteratively removing nodes without incoming edges, and (*ii*) **edge connecting**, which converts supportive relations into directed edges (n_i, n_j) with the restriction that each child node retains at most one parent. The detailed procedure is summarized in Algorithm 1.

Once the DAG is constructed, we define the *high-level state space* S^H as the collection of edge attributes:

$$S^H = \{\, (\hat{n}_i, \underline{n}_i, d_i, t_i) \mid e_i = (\hat{n}_i \to \underline{n}_i) \in \hat{\mathbb{E}} \,\}, \tag{2}$$

where $\hat{n}_i$ and $\underline{n}_i$ denote the parent and child nodes of edge e_i, d_i is the depth of the parent node in the layered structure, and $t_i \in \{0,1\}$ is a binary flag indicating whether the edge has been selected (0 for selected, 1 for unselected).

The attribute vector of edge e_i is then defined as

$$x_i^H = [\, p(\hat{n}_i),\ p(\underline{n}_i),\ d_i \,], \tag{3}$$

where $p(\hat{n}_i), p(\underline{n}_i) \in \mathbb{R}^2$ denote the spatial positions of the parent and child nodes, respectively.

This vector is first passed through a sequence of linear mapping layers followed by normalization:

$$\tilde{h}_i^H = \mathrm{Norm}\big(W^H x_i^H + b^H\big), \tag{4}$$

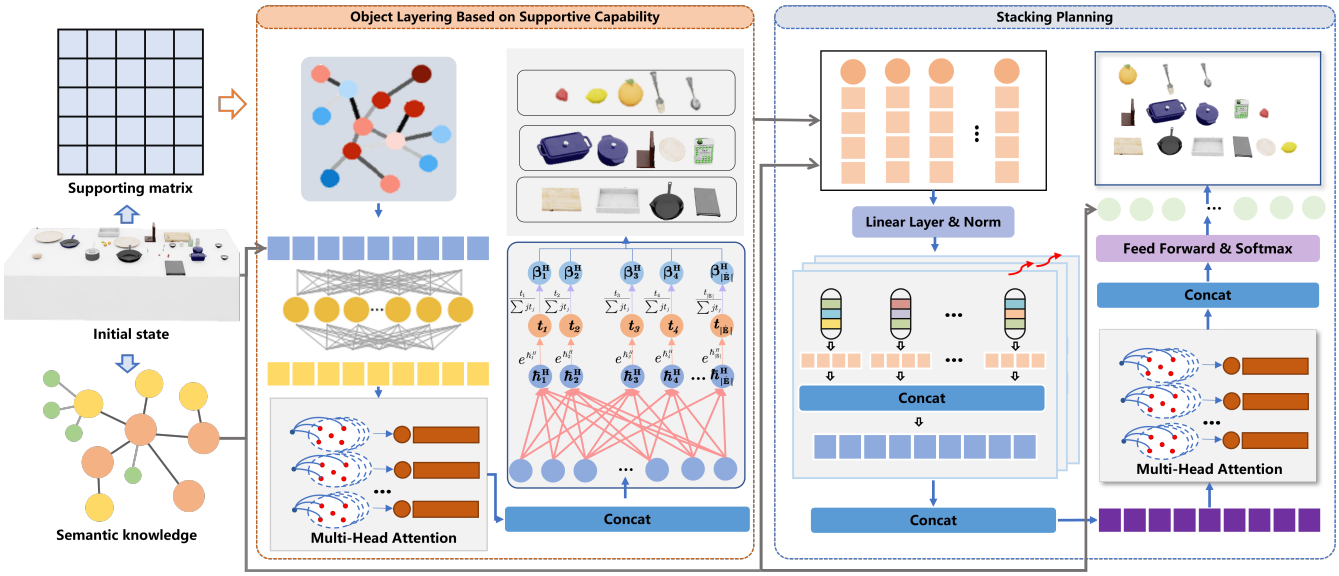

Fig. 2. The HRL framework for multi-object rearrangement is as follows. The objects are initially scattered on a white tabletop, and the goal is to move them to a distant wooden cabinet and arrange them neatly. First, the robot uses a large language model (LLM) to identify the supportive relationships and semantic correlations between the objects. In the high-level decision process of reinforcement learning, the robot groups the objects based on their supportive relationships. The low-level decision process then refines this grouping by incorporating both semantic and supportive information before executing the final task.

where W^H and b^H are learnable parameters at the high level, and $\mathrm{Norm}(\cdot)$ denotes batch normalization or layer normalization.

Subsequently, a multi-head attention mechanism is applied to model contextual dependencies among edges. Given the normalized edge feature $\tilde{h}_i^H$, we project it into a query vector $q_i^{H,r}$, a key vector $k_i^{H,r}$, and a value vector $v_i^{H,r}$ for each attention head $r = 1, \ldots, R$:

$$q_i^{H,r} = W_Q^{H,r}\,\tilde{h}_i^H, \quad k_i^{H,r} = W_K^{H,r}\,\tilde{h}_i^H, \quad v_i^{H,r} = W_V^{H,r}\,\tilde{h}_i^H, \tag{5}$$

where $W_Q^{H,r}, W_K^{H,r}, W_V^{H,r}$ are parameters of head r.

The compatibility score between edge e_i and edge e_j in head r is computed as

$$s_{ij}^{H,r} = \frac{(q_i^{H,r})^\top k_j^{H,r}}{\sqrt{d_k^{H,r}}}, \tag{6}$$

where $d_k^{H,r}$ is the dimension of the key vectors. The attention weight is then obtained using a softmax over all candidate edges:

$$\alpha_{ij}^{H,r} = \frac{\exp\left(s_{ij}^{H,r}\right)}{\sum_{j'=1}^{|\hat{\mathbb{E}}|} \exp\left(s_{ij'}^{H,r}\right)}. \tag{7}$$

The output of head r for edge e_i is the weighted sum of value vectors:

$$z_i^{H,r} = \sum_{j=1}^{|\hat{\mathbb{E}}|} \alpha_{ij}^{H,r}\, v_j^{H,r}. \tag{8}$$

Finally, the outputs from all R heads are concatenated and linearly transformed to obtain the final edge embedding:

$$z_i^H = W_O^H\,[\,z_i^{H,1} \,\|\, z_i^{H,2} \,\|\, \cdots \,\|\, z_i^{H,R}\,] + b_O^H, \tag{9}$$

Algorithm 1: Matrix-to-DAG Conversion Algorithm

Input : Supportive relationship matrix $\mathbb{S} \in \{0,1\}^{m\times m}$, object set $\mathbb{N} = \{n_i\}_{i=1}^m$, a virtual empty node $n_\perp$

Output: Directed acyclic graph $\mathbb{G} = (\mathbb{N}, \hat{\mathbb{E}})$

$\mathbb{S}_0 \leftarrow \mathbb{S}$; $\mathbb{L} \leftarrow \emptyset$; $\hat{\mathbb{E}} \leftarrow \mathbf{0}_{(m+1)\times(m+1)}$; cur $\leftarrow 1$; $\mathbb{N} \leftarrow \mathbb{N} \cup \{n_\perp\}$;

while $\mathbb{S} \neq \emptyset$ **do**

  $\mathbb{L}[\mathrm{cur}] \leftarrow \mathcal{N}_{\mathrm{cur}} \leftarrow \{\, n_j \mid \sum_{i=1}^n \mathbb{S}_{ij} = 0 \,\}$;

  remove from $\mathbb{S}$ the rows/columns indexed by $\mathcal{N}_{\mathrm{cur}}$;

  cur $\leftarrow$ cur $+ 1$;

for $l = 2$ **to** $|\mathbb{L}|$ **do**

  for $n_j \in \mathbb{L}[l]$ **do**

    $\mathcal{P}_j \leftarrow \{\, n_i \in \mathbb{L}[l-1] \mid (\mathbb{S}_0)_{ij} = 1 \,\}$;

    if $\mathcal{P}_j \neq \emptyset$ **then**

      $I_j \leftarrow \{\, i \in \{1, \ldots, m\} \mid n_i \in \mathcal{P}_j \,\}$;

      $i^\star \leftarrow \min I_j$;

      $\hat{\mathbb{E}}_{i^\star j} \leftarrow 1$;

for $n_i \in \mathbb{L}[\,|\mathbb{L}|-1\,]$ **do**

  $\hat{\mathbb{E}}_{i,\,m+1} \leftarrow 1$;

return $\mathbb{G} = (\mathbb{N}, \hat{\mathbb{E}})$;

where W_O^H and b_O^H are learnable parameters, and $\|\cdot\|$ denotes concatenation.

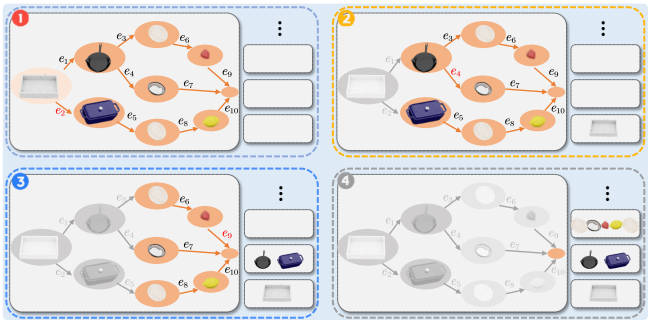

Fig. 3. The high-level decision-making process. The robot first constructs a DAG based on the supportive relationships among objects. It then employs a multi-head attention mechanism to capture contextual dependencies and selects edges iteratively to form hierarchical layers of objects.

Finally, a Softmax function is applied:

$$\begin{aligned} \hbar_i^H &= (w^H)^\top z_i^H + c^H, \\ \beta_i^H &= \frac{\exp(\hbar_i^H)}{\sum_{j=1}^{|\hat{\mathbb{E}}|} \exp(\hbar_j^H)}, \end{aligned} \tag{10}$$

where w^H and c^H are learnable parameters, and β_i^H represents the normalized probability of selecting edge e_i.

After obtaining the edge-level probability distribution from the multi-head attention and Softmax module, the edge selection at step k is formulated as a constrained optimization problem:

$$\begin{aligned} & i^{(k)} = \arg\max_i \beta_i^H, \\ & \text{s.t.,}\ \ t_{i^{(k)}}^{(k-1)} = 1, \\ & \quad d_{i^{(k)}} \le \max_{j=1}^{|\hat{\mathbb{E}}|} d_j, \\ & \quad (k=1) \ \vee \ \left| d_{i^{(k)}} - d_{i^{(k-1)}} \right| \le \delta, \\ & \quad \ell(\underline{n}_{i^{(k)}}) = \ell(\hat{n}_{i^{(k)}}) + 1, \end{aligned} \tag{11}$$

where δ is a predefined threshold controlling the maximum depth difference, and $\ell(\cdot)$ denotes the node layer index.

The layering process terminates when the chosen edge reaches the maximum parent depth, i.e., $d_{i^{(k)}} = d_{\max}$. As a result, the object set $\mathbb{N}$ is partitioned into hierarchical layers:

$$\mathbb{N} = \bigcup_{\ell=1}^{L} \mathbb{N}^{(\ell)}, \qquad \mathbb{N}^{(\ell)} \cap \mathbb{N}^{(\ell')} = \emptyset \ \ (\ell \neq \ell'), \tag{12}$$

where $\mathbb{N}^{(\ell)}$ contains the objects assigned to layer ℓ based on the selected supportive relations. The overall hierarchical decision-making process is illustrated in Figure 3.

D. Stacking Planning

In Section II-C, the object set $\mathbb{N}$ has been divided into $\{\mathbb{N}^{(1)}, \mathbb{N}^{(2)}, \cdots, \mathbb{N}^{(k)}\}$, where objects in adjacent layers exhibit a clear top-down supportive relationship. At this stage, the objects in each layer are further partitioned into several groups. During stacking planning, the object set is represented as the state space S^ℓ. Each object is modeled as a node in a graph, and the attributes of node i include its position p_i, weight w_i, layer index l_i, semantic information Γ_i, allocation flag t_i^ℓ, and the global supportive relationship matrix $\mathbb{S}$. The low-level state space is defined as

$$S^\ell = \{\, (p_i, w_i, l_i, t_i^\ell, \Gamma_i, \mathbb{S}) \mid n_i \in \mathbb{N}^{(\ell)} \,\}. \tag{13}$$

To capture the relationships among objects, a GCCN is employed to extract structural features. By incorporating the concept of capsules, GCCNs capture hierarchical and spatial relations between nodes, thereby enhancing the representational capacity of the model. The initial feature representation of node i is defined as

$$\begin{aligned} & x_i^\ell = [\, p_i, w_i, l_i], \\ & w_i \in \mathbb{R}, \quad l_i \in [1, H], \quad p_i \in \mathbb{R}^2. \end{aligned} \tag{14}$$

Let $X \in \mathbb{R}^{m \times d}$ denote the node feature matrix, where m is the number of nodes and d is the feature dimension. The node features are first projected by a linear mapping layer to obtain the initial embeddings $\tilde{h}_i^\ell$.

Each embedding $\tilde{h}_i^\ell$ is then passed through a series of graph capsule layers. The output of the $(u-1)$-th layer is used to compute the p-th capsule matrix $f_p^{(u)}(X, L)$ using a polynomial graph convolutional filter of order $\mathbb{K}$:

$$\begin{aligned} f_p^{(u)}(X, L) &= \sigma\left(\sum_{\mathfrak{k}=0}^{\mathbb{K}} L^{\mathfrak{k}} \, F_{(u-1)}(X, L) \, W_{p\mathfrak{k}}^{(u)} \right), \\ L &= D - A, \end{aligned} \tag{15}$$

where D is the degree matrix, A is the adjacency matrix, and L is the graph Laplacian. $F_{(l-1)}(X, L)$ denotes the output of the $l-1$-th layer, $W_{p\mathfrak{k}}^{(l)}$ is the learnable parameter of the l-th layer, and $\sigma(\cdot)$ is a nonlinear activation function, which $\mathfrak{k}$ is set to 2 in our experiments.

The outputs of all capsule channels are concatenated to form the overall output of the l-th layer:

$$F_u(X, L) = [f_1^{(u)}(X, L), f_2^{(u)}(X, L), \ldots, f_P^{(u)}(X, L)], \tag{16}$$

where P is the number of capsules. After u layers of graph capsule convolution, $F_u(X, L)$ is passed through a linear mapping layer to produce the final node embeddings h_i^ℓ.

The node embeddings h_i^ℓ are then fed into a multi-head attention mechanism to capture contextual dependencies. Each h_i^ℓ is projected into query, key, and value vectors, and the attention weight between nodes i and j is denoted as α_{ij}^ℓ. The aggregated representation of node i is computed as z_i^ℓ. For the multi-head attention mechanism, results are concatenated and linearly transformed. Finally, a Softmax function is applied to obtain a normalized probability distribution β_i^ℓ.

Given the node probability distribution β_i^ℓ, the selection of the first node in group $\mathcal{G}_k$ is formulated as

$$\begin{aligned} & i^{(\ell,k_1)} = \arg\max_i \beta_i^\ell, \\ & \text{s.t.,}\ \ t_{i^{(\ell,k_1)}}^{(\ell,k_1)} = 1, \\ & \quad l_{i^{(\ell,k_1)}} = \min\{\, l_j \mid n_j \in \mathbb{N},\ t_j = 1 \,\}, \\ & \quad w_{i^{(\ell,k_1)}} \le W_{\max}, \end{aligned} \tag{17}$$

where $W_{\max}$ is the maximum allowable weight for a group, $t^{(\ell,k_1)}_{i^{(\ell,k_1)}}$ is the allocation flag of node $i^{(\ell,k_1)}$ when selecting the first node of group $\mathcal{G}_k$.

After a node is selected, its flag is updated to 0, indicating that it has been chosen. The subsequent nodes of group $\mathcal{G}_k$ are then selected. If the $(j-1)$-th node $n_{i^{(\ell,k_{j-1})}}$ of $\mathcal{G}_k$ has semantic correlations with other nodes,

$$\begin{aligned} & i^{(\ell,k_j)} = \arg\max_i \beta_i^\ell, \\ & s.t.,\ t^{(\ell,k_j)}_{i^{(\ell,k_j)}} = 1, \\ & \quad \mathbb{S}_{i^{(\ell,k_{j-1})},i^{(\ell,k_j)}} = 1, \\ & \quad \sum_{\kappa=1}^{|\mathcal{G}_k|} w_{i^{(\ell,k_\kappa)}} + w_{i^{(\ell,k_j)}} \le W_{\max}, \end{aligned} \tag{18}$$

otherwise,

$$\begin{aligned} & i^{(\ell,k_j)} = \arg\max_i \beta_i^\ell, \\ & s.t.,\ t^{(\ell,k_j)}_{i^{(\ell,k_j)}} = 1, \\ & \quad \mathbb{S}_{i^{(\ell,k_{j-1})},i^{(\ell,k_j)}} = 1, \\ & \quad l_{i^{(\ell,k_j)}} = \min_{q\in\mathbb{Q}} l_q, \\ & \quad \mathbb{Q} = \{q : l_q < l_{i^{(\ell,k_{j-1})}}\}, \\ & \quad \sum_{\kappa=1}^{|\mathcal{G}_k|} w_{i^{(\ell,k_\kappa)}} + w_{i^{(\ell,k_j)}} \le W_{\max}. \end{aligned} \tag{19}$$

If no node satisfying the constraints can be found among the unallocated nodes, the current group ends and the selection for the next group begins, repeating Equations 17, 18, and 19 until all nodes in the current layer are assigned. Finally, the objects are partitioned into groups.

E. Reward Function and Network Training

The robot's execution time consists of grasping/stacking time t_1 and transportation time t_2. For a plan π that partitions objects into $\{\mathcal{G}_1,\dots,\mathcal{G}_K\}$ with $|\mathcal{G}_k| = n_k$, the total time is

$$\begin{aligned} T^\pi &= t_1^\pi + t_2^\pi \\ &= \sum_{k=1}^{K} \Big[C_1(n_k - 1) + \tfrac{1}{v_1}\mathbb{D}_k \Big] + \tfrac{(2K-1)L_1}{v_2}. \end{aligned} \tag{20}$$

where C_1 is the time cost for each additional object in a group, v_1 and v_2 are the average speeds of the robot arm and mobile base, respectively, $\mathbb{D}_k$ is the short distance between objects in group $\mathcal{G}_k$, and L_1 is the distance from the initial position to the target area. The reward is defined as

$$R = -T^\pi, \tag{21}$$

which encourages minimization of execution time.

Since the reward signal is delayed in this sequential decision-making process, we adopt the REINFORCE algorithm for policy optimization. To stabilize training of the hierarchical framework and accelerate convergence, we employ a two-stage training strategy. In the first stage, a heuristic stacking planner is used to train the grouping module based on supportive capability. After obtaining a pre-trained model, the heuristic method is replaced with this model, and the neural network for stacking planning is further optimized. During parameter updates, policies are updated at different time scales to ensure effective coordination between the high-level and low-level agents.

III. Experiments and Results

A. Experimental Settings

This section presents the quantitative analysis and experimental evaluation of our proposed framework. Since this paper investigates the problem of object rearrangement in open scenes, we first design a simulation experiment to validate the effectiveness of our approach. To generate the experimental data, we randomly place n everyday objects on the gray tabletop. We set the movement of the robot in the scene to be one unit length per second, and the time for the robot to operate on an object is set to one second. The straight-line distance between the desktop for generating objects and the target position is set to 7 units. During the experiments, the scale of the training dataset is set to 10,000, and the scale of the test dataset is set to 1,000. Since the supportive relationships between objects in the scene have a significant impact on the experimental results, we need to calculate the maximum stack height of the objects in the generated scene.

To evaluate the algorithm, we first compare the proposed method with an approach that does not consider the supportive relationships. Then, under the consideration of object supportiveness, we compare our grouping decision algorithm with other methods, including random, heuristic algorithms, Ant Colony Optimization (ACO), Attention Mechanism-based RL (AM-RL) [25], and Capsule Attention Mechanism (CapAM) [26]. We perform the experiments on an Ubuntu 20.04 system with an NVIDIA GeForce RTX 3090 GPU.

B. Evaluation Metrics

To evaluate the performance of the algorithm, we selected the following metrics:

- **Number of Stacking Planning Groups** N_g: The number of stacking planning groups refers to the number of groups into which the robot divides the objects. The fewer the stacking planning groups, the fewer the robot's movements during execution, resulting in higher efficiency.
- **Execution Time of the Plan** T_1: The execution time of the plan refers to the time required for the robot to complete all object rearrangement tasks. The shorter the execution time, the higher the efficiency of the algorithm. This time period only includes the movement time of the robotic arm and its base.
- **Decision Time** T_2: The decision time refers to the time required for the robot to complete the object grouping decision. The shorter the decision time, the higher the decision-making efficiency of the algorithm.

TABLE I

COMPARISON OF THE GROUPING METHODS FOR 30 OBJECTS.

Objects	Max_height	Method	N_g	$T_1(s)$	$T_2(s)$
30	——	Standard	——	443.5	——
	6	Random	12.1	246.7	——
		ACO	10.0	218.0	43.0
		CapAM	**8.4**	195.7	**0.9**
		Ours	**8.4**	**195.4**	**0.9**
	7	Random	11.0	232.1	——
		ACO	9.8	214.7	42.0
		CapAM	7.9	188.8	**0.9**
		Ours	**7.7**	**185.9**	**0.9**
	8	Random	9.8	215.2	——
		ACO	9.5	211.1	43.0
		CapAM	**6.9**	**174.7**	**0.9**
		Ours	7.0	175.1	**0.9**

TABLE II

COMPARISON OF THE GROUPING METHODS FOR 40 OBJECTS.

Objects	Max_height	Method	N_g	$T_1(s)$	$T_2(s)$
40	——	Standard	——	593.0	——
	6	Random	17.5	344.6	——
		ACO	15.1	310.8	53.0
		CapAM	**12.1**	**268.8**	**1.1**
		Ours	12.2	269.9	**1.1**
	7	Random	14.7	306.0	——
		ACO	13.2	284.6	53.0
		CapAM	10.9	252.0	**1.1**
		Ours	**10.7**	**249.0**	**1.1**
	8	Random	13.9	293.8	——
		ACO	12.3	272.4	52.0
		CapAM	10.0	239.4	**1.1**
		Ours	**9.5**	**232.8**	**1.1**

TABLE III

COMPARISON OF THE GROUPING METHODS FOR 50 OBJECTS.

Objects	Max_height	Method	N_g	$T_1(s)$	$T_2(s)$
50	——	Standard	——	743.0	——
	6	Random	20.1	401.7	——
		ACO	17.6	366.7	60.0
		CapAM	**13.8**	**313.8**	**1.3**
		Ours	**13.8**	313.9	**1.3**
	7	Random	18.6	381.3	——
		ACO	15.8	342.5	60.0
		CapAM	12.9	301.2	**1.3**
		Ours	**12.8**	**299.9**	**1.3**
	8	Random	17.1	359.6	——
		ACO	14.1	318.7	60.0
		CapAM	12.1	290.7	**1.3**
		Ours	**11.4**	**280.5**	**1.3**

TABLE IV

COMPARISON OF EXECUTION TIME IN REAL-WORLD SCENARIOS.

Objects	Max_height	Method	T_1/s
9	——	Standard	145.2
	4	Ours	**59.3**
	5	Ours	**54.9**
12	——	Standard	194.8
	4	Ours	**81.2**
	5	Ours	**73.4**
15	——	Standard	261.8
	4	Ours	**101.3**
	5	Ours	**95.8**

Fig. 4. The hardware setup of the robot (a) and the real robot experimental environment (b), where objects are placed on the right table, and the target area is on the left.

C. Experimental Results

Since we adopt a two-stage training approach, we first experiment with the object supportive capability grouping module. The number of objects is set to 30, 40, and 50, respectively. The supportive relationship matrix is converted into a graph, and different methods are applied for grouping. Tables I, II, and III present the comparison results for different object counts. As shown, our method performs well across all object numbers. For 30 objects, the number of stacking planning groups N_g is 8.4, the execution time T_1 is 195.4s, and the decision time T_2 is 0.9s. In scenarios with varying maximum stackable heights, our method significantly outperforms others. Compared to the ACO algorithm, N_g is reduced by 1.73 on average, and T_1 is reduced by 29.1s. For 40 and 50 objects, our method continues to outperform alternatives. Furthermore, while our method and CapAM show similar performance, it excels when the environment allows for a larger maximum stackable height. Compared to the standard method, which ignores supportive relationships, our approach reduces the time by 58.2%, 57.8%, and 59.9% when rearranging 30, 40, and 50 objects, respectively.

Since the stacking planning module requires semantic information, we compared the graph capsule convolutional neural network method used in this paper with the network framework from AM-RL. We also evaluated the impact of different orders of statistical moments p on the model. The Figure 5 shows the training performance of the stacking planning module after using the successfully trained object supportive capability grouping module. As shown, our method performs well for different object quantities. In most cases, the model with the 3rd-order statistical moment yields the best results. Additionally, a horizontal comparison reveals that for the same number of objects, the greater the maximum stackable height, the shorter the task duration.

Furthermore, since an LLM was used when determining the support relationship, if a poor-performing language model is employed, it will lead to incorrect input information for the entire model and thereby affect the entire decision-

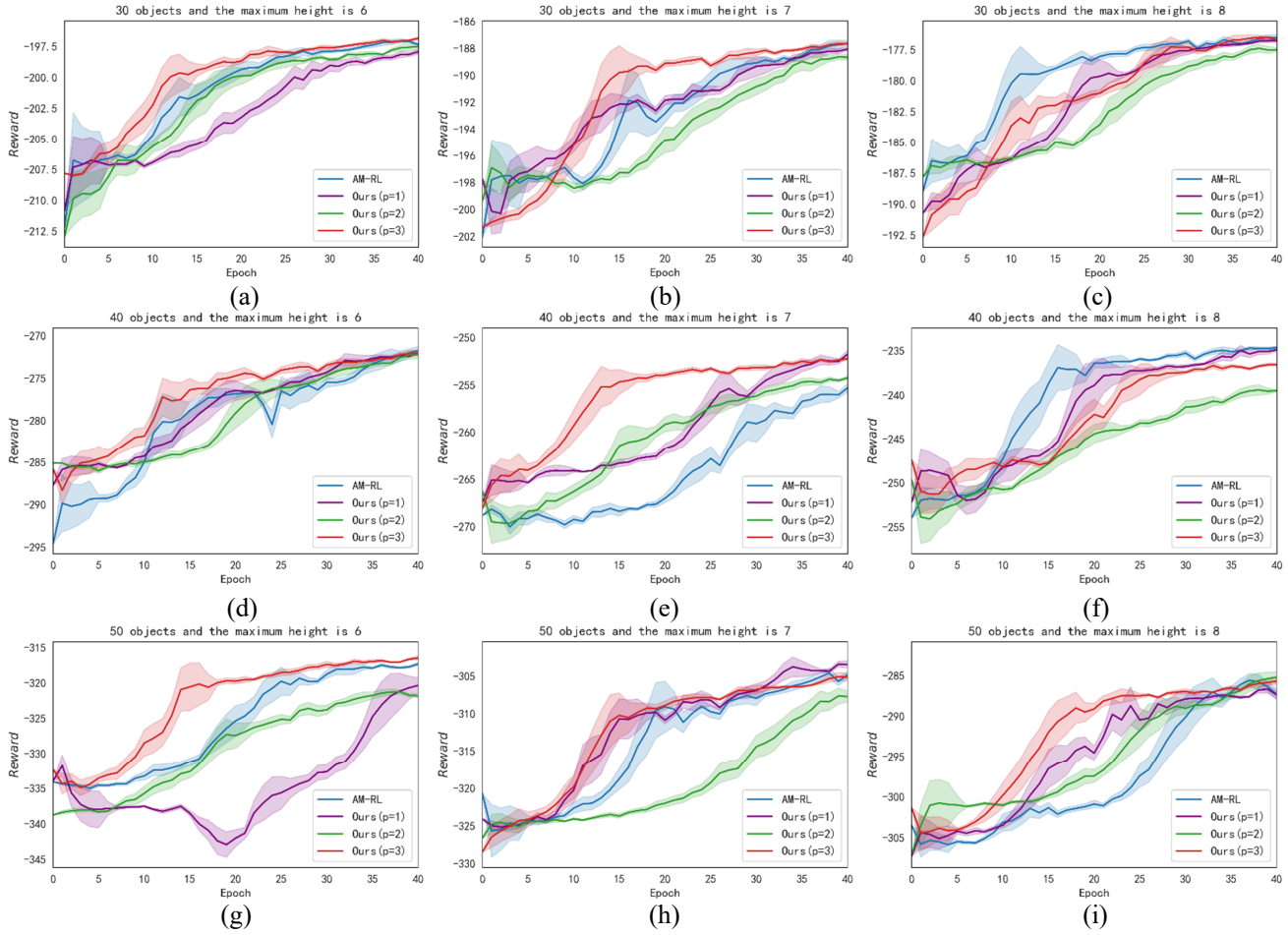

Fig. 5. The training results of the stacking planning module. (a)-(c) show the results when the number of objects is 30, (d)-(f) show the results when the number of objects is 40, and (g)-(i) show the results when the number of objects is 50. It demonstrates the influence of the statistical moments of different orders p on the model. The performance of the model keeps improving as the training process progresses.

making process. Moreover, during the experiment, due to the influence of the material properties of the objects themselves on their supporting capacity, we explicitly stated that flexible objects such as towels and sponges do not have supporting properties.

D. Experiments on Physical Platform

To evaluate the effectiveness of the proposed method in real-world environments, we conducted experiments on a physical platform equipped with a Franka Emika Panda robotic arm and a Realsense D435 camera, as illustrated in Figure 4. Table IV reports the comparison of execution times for rearranging 9, 12, and 15 objects. When rearranging 9 objects, our method achieved execution times of 59.3 s and 54.9 s with maximum heights of 4 and 5, corresponding to reductions of 59.2% and 62.2% compared to the standard method. When rearranging 12 objects, the execution times of our method were 81.2 s and 73.4 s, representing improvements of 58.3% and 62.3% over the standard method. When rearranging 15 objects, our method achieved 101.3 s and 95.8 s, yielding reductions of 61.3% and 63.4% relative to the standard method. Figure 6 demonstrates the robot's actions during this process. The experimental results confirm that our method performs effectively in a real-world physical environment.

IV. Conclusions and Future Work

In this paper, we propose an object rearrangement method based on the supportive relationships between objects. First, we utilize a LLM to extract the supportive relationships and semantic information of the objects. This extracted information is then fed into a hierarchical reinforcement learning framework. The experimental results demonstrate that our method achieves high efficiency when handling varying numbers of objects. Compared to baseline methods, our approach significantly reduces execution time. Furthermore, our method performs well in real-world physical environments, showing robustness to variations in object distribution and initial configurations. However, there are still some challenges, such as spatial constraints in real experimental settings and handling cases where the objects are more dispersed. Moreover, with the continuous advancement of multimodal large models, integrating multimodal object features could further enhance the algorithm's performance.

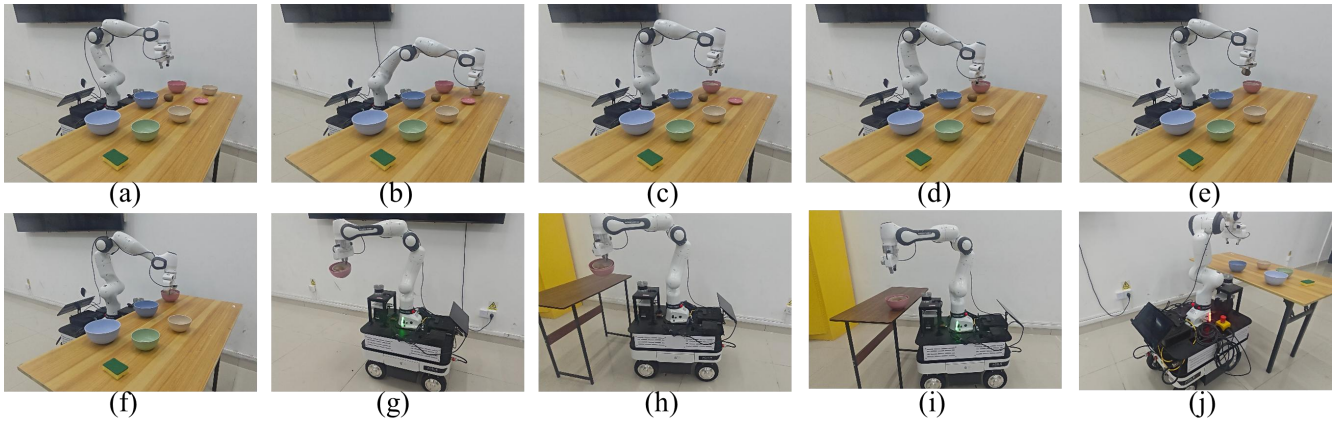

Fig. 6. Execution process of the robot in real-world scenes. Objects are first organized into hierarchical layers based on supportive capability, then grouped and stacked, and finally transported to the designated target area.

Therefore, in future work, we plan to incorporate additional multimodal information into the algorithm to improve its effectiveness. At the same time, expanding the algorithm's applicability to a wider range of real-world scenarios remains a key research direction.

REFERENCES

[1] A. Murali, A. Mousavian, C. Eppner, A. Fishman, and D. Fox, "Cabinet: Scaling neural collision detection for object rearrangement with procedural scene generation," Apr. 2023.

[2] A. Shukla, S. Tao, and H. Su. Maniskill-hab: A benchmark for low-level manipulation in home rearrangement tasks.

[3] K. Ramachandruni, M. Zuo, and S. Chernova, "Consor: A context-aware semantic object rearrangement framework for partially arranged scenes," Sep. 2023.

[4] A. L. Rosenberg, "Finite-state robots in a warehouse: achieving linear parallel speedup while rearranging objects," in *2013 42nd International Conference on Parallel Processing*. IEEE, 2013, pp. 379–388.

[5] M. Danielczuk, A. Mousavian, C. Eppner, and D. Fox, "Object rearrangement using learned implicit collision functions," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 6010–6017.

[6] R. Wang, K. Gao, D. Nakhimovich, J. Yu, and K. E. Bekris, "Uniform object rearrangement: From complete monotone primitives to efficient non-monotone informed search," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 6621–6627.

[7] J. Xu, "Active obstacle avoidance method for robot based on deep visual perception," in *Mechatronics and Automation Technology: Proceedings of the 2nd International Conference (ICMAT 2023), Wuhan, China, 28-29 October 2023*, vol. 46. IOS Press, 2024, p. 365.

[8] S. Ainetter and F. Fraundorfer, "End-to-end trainable deep neural network for robotic grasp detection and semantic segmentation from rgb," Feb. 2022.

[9] J. Lee, T. Lim, L. Bao, and W. Kim, "Collision-free target grasping in complex and cluttered environment for efficient task and motion planning," *IEEE Access*, 2024.

[10] E. Aljalbout, F. Frank, M. Karl, and P. van der Smagt, "On the role of the action space in robot manipulation learning and sim-to-real transfer," *IEEE Robotics and Automation Letters*, 2024.

[11] 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.

[12] D. M. Saxena, M. S. Saleem, and M. Likhachev, "Manipulation planning among movable obstacles using physics-based adaptive motion primitives," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 6570–6576.

[13] D. Huang, C. Tang, and H. Zhang, "Efficient object rearrangement via multi-view fusion," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 18 193–18 199.

[14] I. Kapelyukh, V. Vosylius, and E. Johns, "Dall-e-bot: Introducing web-scale diffusion models to robotics," *IEEE Robotics and Automation Letters*, vol. 8, no. 7, pp. 3956–3963, 2023.

[15] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn *et al.*, "Rt-2: Vision-language-action models transfer web knowledge to robotic control," *arXiv preprint arXiv:2307.15818*, 2023.

[16] Y. Ding, H. Geng, C. Xu, X. Fang, J. Zhang, S. Wei, Q. Dai, Z. Zhang, and H. Wang, "Open6dor: Benchmarking open-instruction 6-dof object rearrangement and a vlm-based approach," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024, pp. 7359–7366.

[17] A. Goyal and J. Deng, "Packit: A virtual environment for geometric planning," in *International Conference on Machine Learning*. PMLR, 2020, pp. 3700–3710.

[18] T. D. Kulkarni, K. Narasimhan, A. Saeedi, and J. Tenenbaum, "Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation," *Advances in neural information processing systems*, vol. 29, 2016.

[19] A. S. Vezhnevets, S. Osindero, T. Schaul, N. Heess, M. Jaderberg, D. Silver, and K. Kavukcuoglu, "Feudal networks for hierarchical reinforcement learning," in *International conference on machine learning*. PMLR, 2017, pp. 3540–3549.

[20] X. Yang, Z. Ji, J. Wu, Y.-K. Lai, C. Wei, G. Liu, and R. Setchi, "Hierarchical reinforcement learning with universal policies for multistep robotic manipulation," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 9, pp. 4727–4741, 2021.

[21] Y. Ji, Z. Li, Y. Sun, X. B. Peng, S. Levine, G. Berseth, and K. Sreenath, "Hierarchical reinforcement learning for precise soccer shooting skills using a quadrupedal robot," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 1479–1486.

[22] Y. Zhou and H. W. Ho, "Online robot guidance and navigation in non-stationary environment with hybrid hierarchical reinforcement learning," *Engineering Applications of Artificial Intelligence*, vol. 114, p. 105152, 2022.

[23] R. Gieselmann and F. T. Pokorny, "Planning-augmented hierarchical reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5097–5104, 2021.

[24] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[25] W. Kool, H. Van Hoof, and M. Welling, "Attention, learn to solve routing problems!" *arXiv preprint arXiv:1803.08475*, 2018.

[26] S. Paul, P. Ghassemi, and S. Chowdhury, "Learning scalable policies over graphs for multi-robot task allocation using capsule attention networks," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 8815–8822.