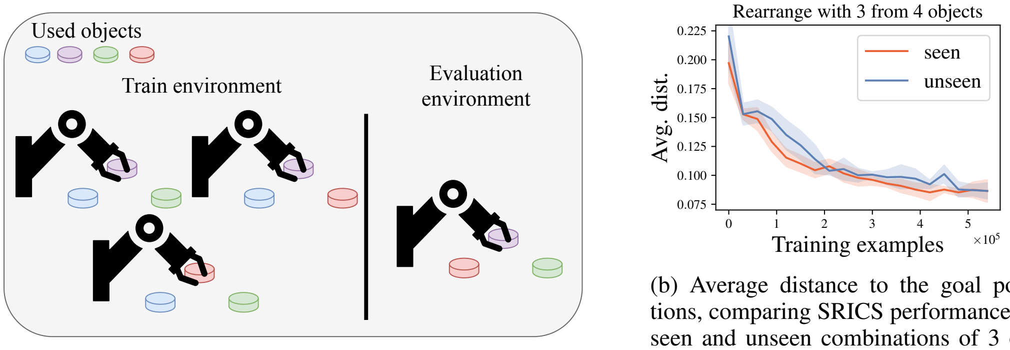
Appendix

A SRICS pseudocode

Algorithm 1 SRICS: Self-Supervised Relational RL with Independently Controllable Subgoals

```
Require: GNN Dynamical model $D$, goal-conditional attention policy $\pi_\theta$, goal-conditional SAC trainer, number of training episodes $K$.
Train GNN Dynamical model $D$ on sequences uniformly sampled from $\mathcal{D}$ using the loss described in Eq. 5 and estimate the interaction graph $G$.
for $n = 1, ..., K$ episodes do
    Sample goal $\mathbf{s}^{\text{goal}}$ and construct subgoal $\mathbf{g}^i$ using $G$.    ▷ See Eq. 6
    Collect episode data with policy $\pi_\theta(\mathbf{a}_t \mid \mathbf{s}_t, \mathbf{g}^i)$.
    Store transitions $\{(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}, \mathbf{g}^i), \ldots\}$ into replay buffer $\mathcal{R}$.
    Sample transitions from replay buffer $(\mathbf{s}, \mathbf{a}, \mathbf{s}', \mathbf{g}^i) \sim \mathcal{R}$.
    Relabel $\mathbf{g}^i$ goal components to a combination of future states and goal sampling distribution.
    Compute selectivity reward signal $R = r_{\text{sel},i}(\mathbf{s}', \mathbf{s}, \mathbf{g}^i)$.    ▷ See Eq. 7
    Update policy $\pi_\theta(\mathbf{a}_t \mid \mathbf{s}_t, \mathbf{g}^i)$ using $R$ with SAC trainer.
end for
```

B Generalization to Unseen Combination of Objects



(a) Train and evaluation environments.

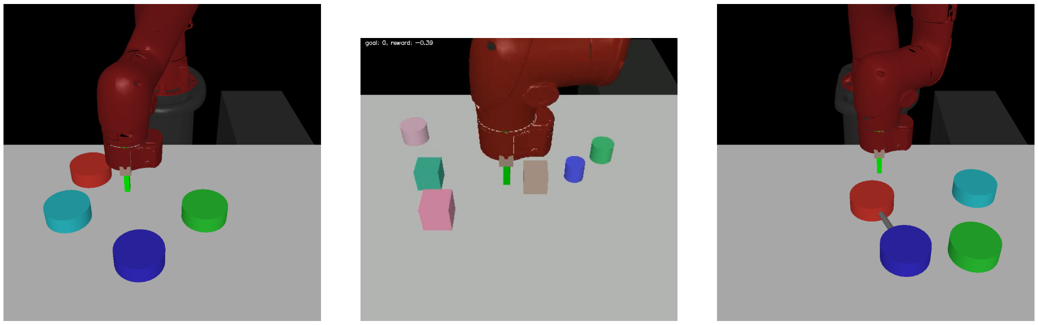
(b) Average distance to the goal positions, comparing SRICS performance on seen and unseen combinations of 3 objects.

Figure 6: Generalization to unseen combination of objects.

To test our agent on a novel combination of objects, we modify the *Multi-Object Rearrange* environment with 4 objects by deleting one of the objects from the table. We split possible combinations of three objects on training and evaluation combinations as shown in Fig. 6a. We train the GNN dynamical model on the training combinations and then average all interaction weights to estimate the global interaction graph. Next, we train our SRICS agent on the training combinations and evaluate it on an unseen combination. The performance of SRICS on the unseen combination is close to its performance on *Multi-Object Rearrange* with 3 objects (see Fig. 6b). This demonstrates that agents equipped with object-centric representations and a compatible policy are not restricted to the particular combination of objects they were trained on. Such agents should be able to learn how to control many objects and reuse learned skills for manipulation over different scenes containing only a random subset of objects.

C Multi-object Rearrange Environments

We implement several modifications of the original *Multi-object Rearrange* environment to study how our agent performs in more challenging settings. First, we implement the *Multi-object Relational Rearrange* environment by incorporating additional constraints to the *Multi-object Rearrange* environment. In particular, for the *Multi-object Rearrange* environment with 4 objects, we add one spring connection and make one object static. The goals are sampled from a random arrangement of the objects, where the constraints above are fulfilled. Next, we implement *Multi-object Rearrange* with different objects by varying objects' shapes (cube and cylinder), size, and mass. The manipulation of such different objects is more challenging thus an agent has to learn more complex policy (see Fig. 7 for the visualization of the environments)



(a) (b) (c)

Figure 7: Visualization of the *Multi-object Rearrange* environment with a) 4 objects, b) 6 different objects and c) *Multi-object Relational Rearrange environment*.

D Additional Experimental Results

D.1 Larger Number of Different Objects

We have conducted several additional experiments to study how the SRICS method performs in a more challenging environment with a larger number of different objects. In particular, we trained the SRICS agent in the *Multi-object Rearrange* environment with 6 different objects (see Fig. 7b). We compared SRICS performance to the SMORL and SAC baselines that are shown to work in more simple *Multi-object Rearrange* with 4 objects. For these experiments, we do no additional hyperparameter optimization using optimal parameters from *Multi-object Rearrange* with 4 objects.

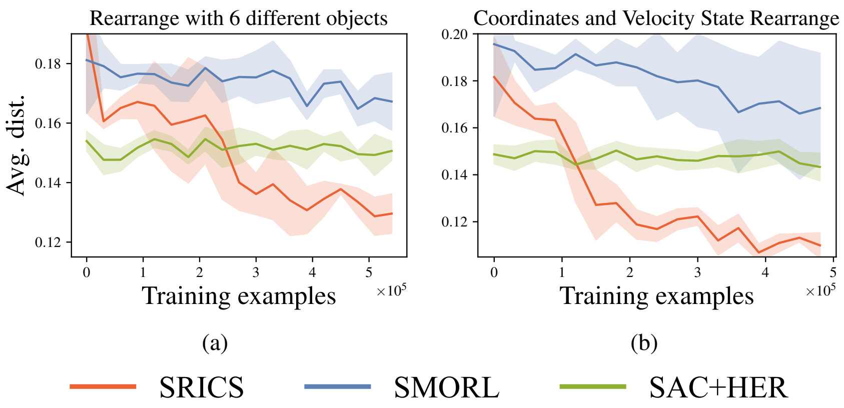
We show the results in Fig. 8a. The SRICS agent makes progress in this environment while the SMORL agent performs close to a random agent and SAC consistently solves only the easiest "arm" subgoal. This shows that the SRICS agent can learn and efficiently combine many subtasks when the subtasks are different (e.g. manipulation of objects with different shapes).

D.2 State Representation Extended with Object's Velocity

In addition to more challenging environments, it is also important to show that the SRICS method is not restricted to coordinate-based object-centric representations. For this, we studied the performance of the SRICS agent when the state representation also includes the object's velocity. In the modified environment the representation of each object is encoded by the position vector $\mathbf{s}^{j,\text{where}} \in \mathbb{R}^2$, the velocity vector $\mathbf{s}^{j,\text{vel}} \in \mathbb{R}^2$ and the identifier $s^{j,\text{what}} \in \mathbb{R}^K$. For all the methods, we use positions $\mathbf{s}^{j,\text{where}}$ to calculate the distance to the goal in the reward signal. The results are presented in Fig. 8b. The SRICS agent outperforms both baselines and reaches the performance that is comparable to its performance with coordinates-based state representation.

E Goal-Conditioned Attention Policy

We train one policy that incorporates all learned skills. For this purpose, we use a goal-conditioned attention policy [10]. This policy needs to vary its behavior based on the goal at hand (e.g. one goal can be reaching a particular position with the robotic arm, whereas another goal can be moving an object to a particular position). To allow this flexibility, we use the attention module with a goal-dependent query $Q(\mathbf{s}^{\text{goal},i}) = \mathbf{s}^{\text{goal},i} W^q$. Each object is allowed to match with the query via an object-dependent key $K(\mathbf{s}_t) = \mathbf{s}_t W^k$ and contribute to the attention's output through the value



(a) (b)

Figure 8: Average distance to the goal positions, comparing our method to the SAC and SMORL baselines on a) *Rearrange* environment with 6 different objects and b) *Rearrange* environment with 4 objects with coordinates and velocity state representation.

$V(\mathbf{s}_t) = \mathbf{s}_t W^v$, which is weighted by the similarity between $Q(\mathbf{s}^{\text{goal},i})$ and $K(\mathbf{s}_t)$. The attention head A_k is computed as

$$A_k = \text{softmax}\left(\frac{\mathbf{s}^{\text{goal},i} W^q (S_t W^k)^T}{\sqrt{d_e}}\right) S_t W^v,$$

where S_t is a packed matrix of all $\mathbf{s}_t^i$'s; the parameters W^q, W^k, W^v constitute learned linear transformations and d_e is the common dimensionality of the key, value and query. The attention output A is a concatenation of all attention heads $A = [A_1; \ldots; A_K]$.

Finally, the attention output A is combined with the subgoal $\mathbf{s}^{\text{goal},i}$ and processed by a fully connected neural network f:

$$\pi_\theta\left(\{\mathbf{s}_t^i\}_{i \in \mathcal{O}_t}, \mathbf{s}^{\text{goal},i}\right) = f(A, \mathbf{s}^{\text{goal},i}).$$

The parameters used for training of the goal-conditioned attention policy are presented in App. J.1.

F Subgoals Selectivity as an Evaluation Metric

The selectivity (as defined in Eq. 7) is a measure of the agent's selective influence on the components of the environment. In Sec. 3.2, we show that it can be used as an additional reward signal to motivate the agent to selectively control different components of the environment. Additionally, the selectivity can be used as an evaluation metric. This metric features how selective is the influence of an agent on the components of the environment. Here, we compute the selectivity measure for SRICS and SMORL agents that learn to control components of the representation separately from each other. As seen in Fig. 9, the selectivity measure increases for both agents during the goal-conditioned training. Concurrent with the objective the SRICS agent is trained on, the selectivity measure for the SRICS agent is increasing much faster and with smaller variance compared to the SMORL agent. Therefore, the selectivity is an important objective for autonomous control that can make training more stable and efficient.

G Estimation of the Global Interaction Graph

To learn sparse interaction weights w_t^{ij}, we use the sparsity prior p_{prior} (see Eq. 5). Specifically, the sparsity prior p_{prior} is the Bernoulli distribution

$$f(k;p) = \begin{cases} p & \text{if } k = 1, \\ 1 - p & \text{if } k = 0. \end{cases}$$

For all experiments, we use the same prior probability for the relation presence $p = 0.05$. The interaction weights w_t^{ij} deviate from this prior only when the relation is required for the improvement of

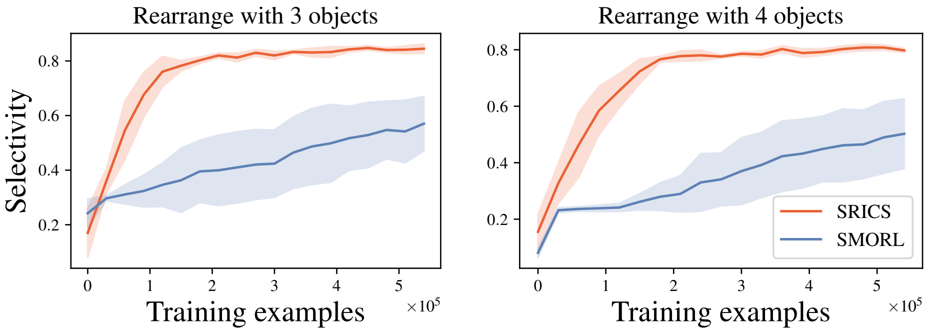


Figure 9: The selectivity part of the reward signal for both SRICS and SMORL agents averaged over all entities. While the SMORL agent is not optimized for being selective, the selectivity increases over SMORL training because the agent is gaining control over objects. However, for the SRICS agent, the increase in selectivity is much faster as the agent is incentified to be selective.

the dynamical model predictions. As can be seen in Fig. 10, such approach successfully reconstructs most of the relations for both *Multi-Object Rearrange* and *Multi-Object Relational Rearrange* environments.

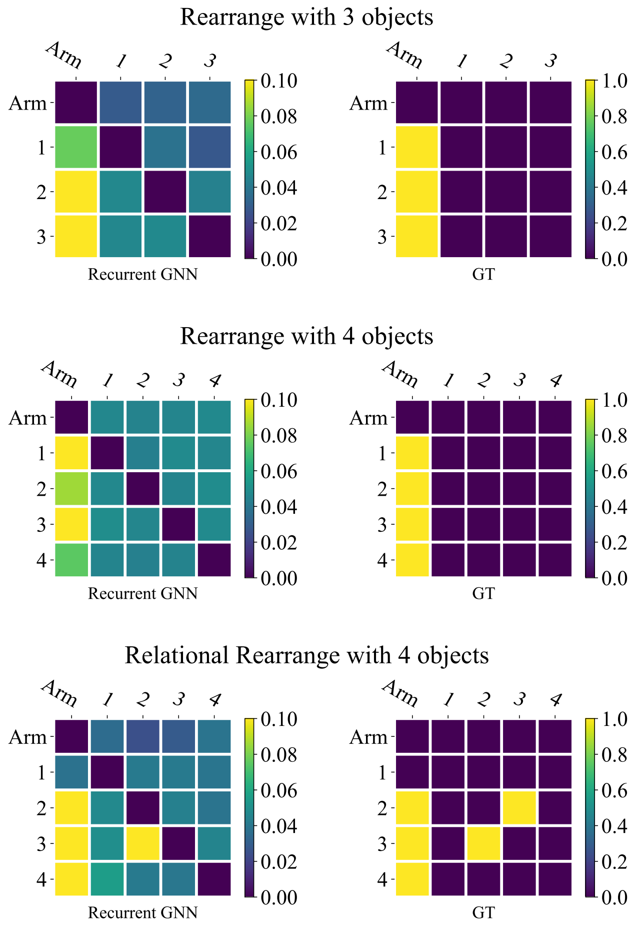


Figure 10: Average interaction weights obtained from the GNN dynamical model.

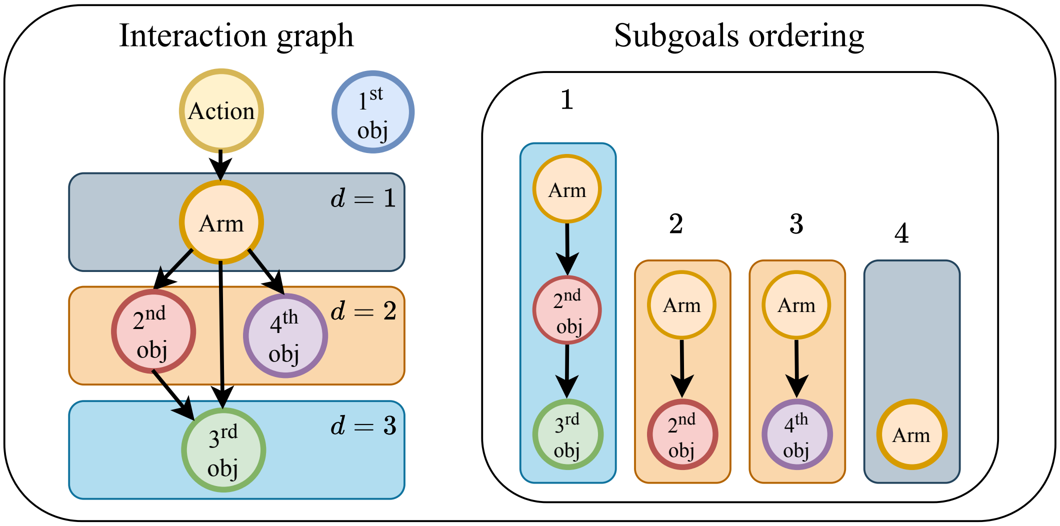


Figure 11: Ordering of the independently controllable subgoals according to the depth of the corresponding nodes in the interaction graph. When the interaction graph is a DAG, such ordering corresponds to the reversed topological ordering.

H Evaluation on the Average Objects Distance

We additionally evaluate our method on the average object distance metric, similarly to the SMORL paper [10]. This metric is calculated as the average of all distances from objects on a table (without arm) to their goal position. Thus, it is biased towards controlling the external objects (which are mostly independent of each other). As can be seen in Fig. 12, SRICS outperforms SMORL on this metric, whereas SAC performs similar to a passive policy. This result shows the benefit of using the goal-directed selectivity reward signal for the control of external objects. In contrast to the average object distance metric, the average distance metric presented in this paper also reveals the importance of the goal decomposition into a sequence of compatible subgoals.

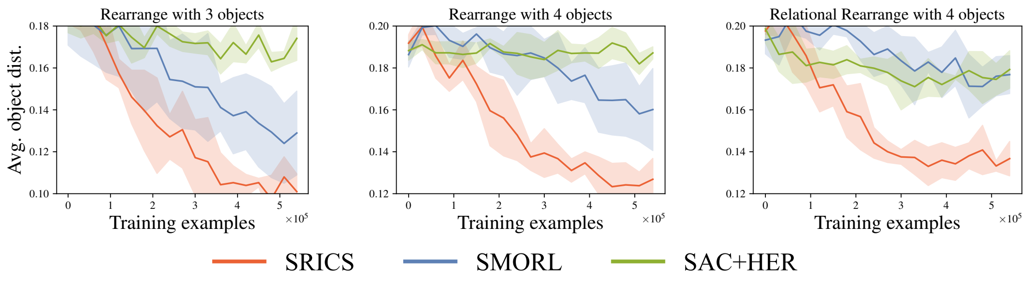
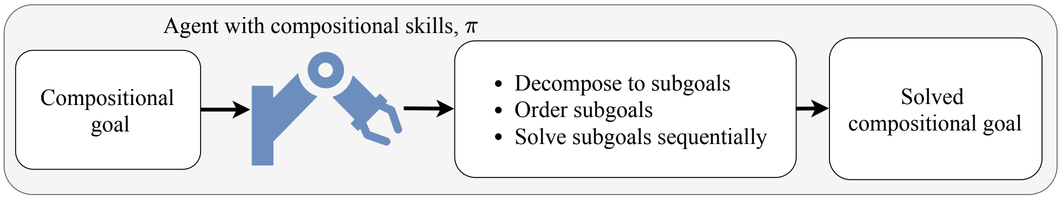


Figure 12: Average object distance to the goal positions, comparing SRICS to SMORL and SAC+HER.

I Ordering of the Subgoals

As reaching one subgoal can affect the results of reaching other subgoals, it is necessary to order the subgoals such that the resulting sequence of skills is compatible. Intuitively, for each compositional goal, we want to first manipulate such objects that require movement of other objects for their manipulation. For example, in case of a robotic arm and objects on the table, we first want to control objects using the robotic arm and then control the arm itself. As the robotic arm has no dependencies in the interaction graph, the corresponding selectivity reward signal should incentify the agent to control the arm while not affecting all other objects, thus making the arm subgoal compatible with objects' subgoals (if solved perfectly).

Generally, we order all subgoals by their depth in the interaction graph, executing subgoals with larger depth first (as illustrated in Fig. 11). The depth of a node is defined as the length of the longest path without loops from the action variable A to the node. The order of the subgoals with the same depth is random. When the learned interaction graph is a directed acyclic graph (DAG), such ordering corresponds to the reversed topological ordering. The nodes in a DAG are *topologically ordered* if for any edge $v \to u$ in graph G, node v comes after node u. Due to such ordering, only the subgoals that correspond to the nodes $j \notin \mathrm{Anc}(i)$ are executed before the subgoal i. These subgoals should not be affected when the selectivity part (Eq. 7) of the reward signal is maximized. Thus, such ordering of the subgoals guarantees the compatibility of the subgoals sequence when each of the subgoals is solved with a maximal reward signal.



Agent with compositional skills, π
Compositional goal
• Decompose to subgoals
• Order subgoals
• Solve subgoals sequentially
Solved compositional goal

Figure 13: SRICS pipeline during evaluation.

J Implementation Details

J.1 SRICS

We refer to Table 1 and Table 2 for the hyperparameters of SRICS for all environments. We use the same number of subgoal solving attempts as in SMORL. During the evaluation, the number of attempts is equal to 7 for environments with 3 objects and 9 for environments with 4 objects. As the number of attempts k is larger than the number of entities n, we order only the last n subgoals.

J.2 Prior Work

For both SMORL and SAC, we use previously optimized settings for *Multi-Object Rearrange* with 3 and 4 objects from Zadaianchuk et al. [10]. In addition, we make a hyperparameter search over more than 100 runs for finding the best HAC hyperparameters. Specifically, we evaluate the performance of HAC while varying the number of steps for each subgoal, number of levels, and action noise. For the *Multi-Object Relational Rearrange* environment with 4 objects, we use the same parameters as in the *Multi-Object Rearrange* environment with 4 objects for all algorithms.

Hyperparameter	**Value**
Selectivity parameter α	0.25
Optimizer	Adam with default settings
RL Batch Size	2048
Reward Scaling	1
Automatic SAC entropy tuning	yes
SAC Soft Update Rate	0.05
# Training Batches per Time Step	1
Hidden Activation	ReLU
Network Initialization	Xavier uniform
Separate Attention for Policy & Q-Function	yes
Replay Buffer Size	250000
Relabeling Fractions Rollout/Future/Sampled Goals	0.1 / 0.4 / 0.5
Number of Initial Random Samples	10000
Number of Heads	5
Discount Factor	0.95
Learning Rate	0.001
Policy Hidden Sizes	$[128, 128]$
Q-Function Hidden Sizes	$[128, 128, 128]$
Training Path Length	20

Table 1: General hyperparameters used by SRICS for all environments.

Hyperparameter	**Value**
Sparsity prior p	0.05
Threshold θ	0.06
Number of episodes	1000
Episode length	50
Sequence size during RNN modeling T	20
Number of updates	100000
Learning Rate	0.0005
Batch Size	100

Table 2: Hyperparameters for the interaction graph estimation for all environments.