Effective Baselines for Multiple Object Rearrangement Planning in Partially Observable Mapped Environments

Engin Tekin[1], Elaheh Barati[2*], Nitin Kamra[1], and Ruta Desai[1]

[1] Meta Reality Labs Research, [2] TikTok

{tekinengin21, nitinkamra, rutadesai}@meta.com, elaheh.barati@gmail.com

Abstract

Many real-world tasks, from house-cleaning to cooking, can be formulated as multi-object rearrangement problems – where an agent needs to get specific objects into appropriate goal states. For such problems, we focus on the setting that assumes a pre-specified goal state, availability of perfect manipulation and object recognition capabilities, and a static map of the environment but unknown initial location of objects to be rearranged. Our goal is to enable home-assistive intelligent agents to efficiently plan for rearrangement under such partial observability. This requires efficient trade-offs between exploration of the environment and planning for rearrangement, which is challenging because of long-horizon nature of the problem. To make progress on this problem, we first analyze the effects of various factors such as number of objects and receptacles, agent carrying capacity, environment layouts etc. on exploration and planning for rearrangement using classical methods. We then investigate both monolithic and modular deep reinforcement learning (DRL) methods for planning in our setting. We find that monolithic DRL methods do not succeed at long-horizon planning needed for multi-object rearrangement. Instead, modular greedy approaches surprisingly perform reasonably well and emerge as competitive baselines for planning with partial observability in multi-object rearrangement problems. We also show that our greedy modular agents are empirically optimal when the objects that need to be rearranged are uniformly distributed in the environment – thereby contributing baselines with strong performance for future work on multi-object rearrangement planning in partially observable settings.

1 Introduction

Rearrangement problems, where the goal is to get a physical environment in a specific state has been proposed as the next frontier for embodied AI research (Batra et al. 2020). Many tasks in everyday life from house cleaning (Newman, Carlberg, and Desai 2020; Szot et al. 2021) to preparing groceries (Szot et al. 2021) can be formulated as rearrangement problems. Therefore, developing embodied agents to solve these problems would allow to us to make progress towards the next generation of home assistant agents.

In an embodied rearrangement task, an agent must rearrange an unknown environment using a combination of sensor observations and prior knowledge to reach a goal state, which is specified either geometrically, or through image, language, or predicates (Batra et al. 2020). Solving such generic rearrangement tasks requires an agent to solve a plethora of sub-problems, such as, reasoning about the goal state through semantic and commonsense understanding; building a map of the environment in order to navigate, search, and explore; effectively planning to figure out which objects to pick-drop and in what order; and finally manipulating objects. These problems span the spectrum of perception, planning, navigation, and manipulation making rearrangement an extremely challenging problem (Batra et al. 2020).

Because of the complexity of rearrangement problems, previous research has focused on different slices of the problem. Some researchers focus on understanding the goal state by leveraging human preferences and commonsense reasoning (Kant et al. 2022; Kapelyukh and Johns 2022); or through reasoning about changes in the environment configuration (Weihs et al. 2021; Hall et al. 2020). Another body of work focuses on perception, planning, and manipulation for rearrangement, albeit for small number (upto 5) of objects (Szot et al. 2021; Karkus et al. 2020). We focus on a specific slice of the rearrangement problem – *planning under partial observability*. In particular, to decouple our investigation from the manipulation, navigation and perception challenges, we assume perfect object recognition and interaction capabilities in the agent. We further assume the availability of a static map to focus on high-level task planning for rearrangement instead of integrated task and motion planning (TAMP) (Garrett et al. 2021). Lastly, instead of dealing with uncertain information ranging from incomplete maps, mis-classified objects etc., which makes it challenging to study the planning problem, we directly consider uncertainty over the object locations – decoupled from perception – which allows us to study the implications of uncertainty on the rearrangement planning problem in a more systematic and controlled manner. Specifically, the problem requires efficient exploration of the environment in combination with well-balanced planning for rearrangement and this forms the core of our work.

Overall, given a pre-specified goal state, perfect manipulation and object recognition capabilities, and a static map of the environment; our goal is to enable efficient planning

[*]Work done at Meta

Copyright © 2023, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

for rearrangement. Such a laser focus consequently enables us to analyze and present the effects of various factors such as number of objects, agent carrying capacity, environment layout complexity etc. on the complexity of planning. We find that higher agent capacity and larger environment layouts make the rearrangement planning problem challenging. On the other hand, counter-intuitively, higher number of objects reduces the problem complexity, since in this case rearrangement for seen objects implicitly shoulders the burden of exploration for unseen objects, thereby reducing the need for additional explicit exploration, and ultimately reducing the problem complexity.

Classical planners often fail to optimally solve task planning problems in real-time (Agia et al. 2022) and need a world model to be known. Motivated by these limitations of the classical approaches, we next present end-to-end monolithic deep reinforcement learning (DRL) based approaches. We find that akin to prior works, monolithic DRL methods do not succeed at long-horizon planning problems. We then propose modular methods as competitive baselines for planning under uncertainty in rearrangement problems. Specifically, our methods investigate ways to achieve better trade-off between exploration and planning. We empirically demonstrate that approaches which plan greedily and only explore conservatively achieve this trade-off optimally when objects that need to be rearranged have uniform distribution in the environments. We hope that our analysis and baselines will provide a good starting point and benchmark for future work on rearrangement planning.

2 Related Work

2.1 Object rearrangement

Object rearrangement has been studied in robotics (Ben-Shahar and Rivlin 1998), for a variety of problems ranging from table-top rearrangement (Ben-Shahar and Rivlin 1998; King, Cognetti, and Srinivasa 2016; Huang, Jia, and Mason 2019) to house-scale rearrangement (Agia et al. 2022). Majority of planning research for rearrangement in robotics further focuses on integrated task and motion planning (King, Ranganeni, and Srinivasa 2017; King, Cognetti, and Srinivasa 2016; Ben-Shahar and Rivlin 1998). Instead, our focus is on high-level task planning for house-scale rearrangement. Recent work from Newman et al. (2020) and Agia et al. (2022) have also focused on task planning for house-scale object rearrangement problems, however they assume full observability while we are interested in task planning for rearrangement under partial observability.

Embodied AI community, which is interested in virtual agents in addition to physical agents, has also started pushing on house-scale rearrangement problems (Batra et al. 2020; Szot et al. 2021; Weihs et al. 2021; Kant et al. 2022; Kim et al. 2021). Because of the use of perceptual sensors, they also have to deal with partial observability while doing rearrangement planning. These works push on solving rearrangement planning in combination with perception, manipulation, and navigation for rearrangement. Consequently, they only show rearrangement for scenarios with small number of objects (upto 5) and single agent capacity. Instead, we are interested in understanding various factors that affect rearrangement planning problem under partial observability. In particular, our focus is on the trade-off between two main components of rearrangement planning: object discovery via exploration and planning for rearrangement. We next review related work for exploration and planning.

2.2 Exploration in 3D environments

Exploring unknown environments is a well-studied problem in robotics and state-of-the-art results are generally achieved with frontier-based exploration (FBE) (Yamauchi 1998; Chen, Gupta, and Gupta 2019). The overarching idea in frontier-based approaches is to define a set of frontier locations and decide the next frontier location to explore (Yamauchi 1998). Recent studies have employed utility functions to choose between frontier locations in combination with DRL techniques for exploration (Chen, Gupta, and Gupta 2019; Niroui et al. 2019). Ramakrishnan et al. (2020) further benchmarked various DRL approaches with several reward functions as well as FBE method for embodied exploration. As an alternative approach to enable exploration, Fukazawa et al. (2005) uses the reaction-diffusion equation on graphs and optimize a custom potential function in order to create observation points; and then visit them in an optimal order. However, most of these studies focus solely on exploration of a 3D environment with the goal of maximizing area coverage or object/landmark detection. Instead, we are interested in exploration for object discovery to enable downstream rearrangement. We take inspiration from these studies and investigate both learning-based and frontier-based exploration approaches to enable rearrangement.

2.3 Planning for rearrangement

End-to-end DRL approaches as well as classical approaches based on Traveling Salesman Problem (TSP) formulation have both been used for rearrangement planning (Newman, Carlberg, and Desai 2020; Kant et al. 2022; Karkus et al. 2020; Andreas, Klein, and Levine 2016; Szot et al. 2021). Since multiple object rearrangement is a long-horizon problem, monolithic DRL approaches for the same have only been shown to be successful on small grid maps (Karkus et al. 2020; Andreas, Klein, and Levine 2016). Researchers have therefore used modular approaches (Szot et al. 2021; Kant et al. 2022) or classical approaches based on TSP (Newman, Carlberg, and Desai 2020) for house-scale rearrangement planning. Based on these findings, we also model planning for rearrangement as Capacitated Vehicle Routing Problem (CVRP), a variant of TSP, and use OR-Tools (Perron and Furnon 2022) to solve the same.

3 Problem Characterization

In this section, we concretely define the Multiple Object Rearrangement Planning (MORP) problem in mapped environments with partial observability over object locations. We also detail our procedure for generating the required datasets and evaluation metrics.

3.1 Task Definition

Notation: We will use a Partially Observable Markov Decision Processes (POMDP) framework to formally define MORP. Let $s \in \mathcal{S}$ denote a state in state space $\mathcal{S}$, $o \in \mathcal{O}$ denote an observation in observation space $\mathcal{O}$, $a \in \mathcal{A}$ denote action in action space $\mathcal{A}$, $g \in \mathcal{G}$ denote a goal specification in goal space $\mathcal{G}$, and finally $\pi(a_t|o_{t-1}...o_0, g) = Pr(a_t|o_{t-1}...o_0, g)$ agent's goal-conditioned policy. Note that we will omit time t subscript unless indicated otherwise.

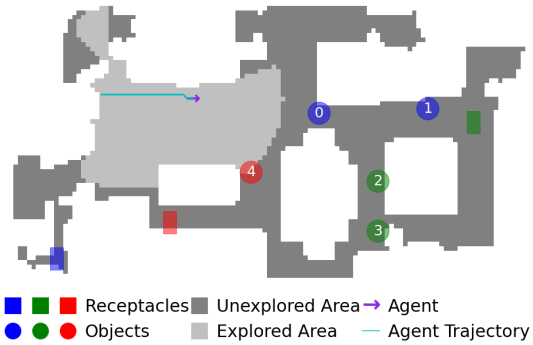

Figure 1: Multiple Object Rearrangement Planning (MORP) problem. Top down map of the house with the objects to be rearranged shown as circles and receptacles as squares. Same color signifies object-receptacle pairing and multiple objects of the same type can be placed into a receptacle (e.g., books into bookshelf). With perfect object recognition capabilities, unique instance IDs track the objects.

Goal Specification: We consider type-level geometric goal for rearrangement as described in (Batra et al. 2020) i.e., we describe the goal state for rearrangement geometrically with target location for all objects. Note that in this paper we will only use integers for location variables. Let $\mathbb{Z}$ denote the set of integers and $\mathbb{N}$ denote the set of natural numbers. We define $p_i^0 \in \mathcal{P} : \mathbb{Z}^{n_o \times 2}$ to be initial 2D object locations and $r_i \in \mathcal{R} : \mathbb{Z}^{n_r \times 2}$ to be 2D receptacle locations where $n_o \in \mathbb{N}$ is number of the objects and $n_r \in \mathbb{N}$ number of object types/receptacles. Each object type has a unique receptacle but multiple objects can be rearranged to the same receptacle. Let f be the correspondence function to map object index to an index corresponding to its receptacle type, so that we can elicit the object-receptacle pairing. A goal is then specified as $g = \bigwedge_{i=1}^{n_o}(p_i == r_{f(i)})$.

Rearrangement Task: Thus, we can formally define a rearrangement task as, given a goal specification g, the agent must transform the initial state s_0 to a goal state s^*, by solely acting based on observations $o \in \mathcal{O}$, where $s^* \in \mathcal{S}$ is a state that satisfies goal specifications. Hence, an agent spawned randomly in a home environment must find objects scattered randomly around the house and rearrange them to their desired receptacles (see Figure 1).

3.2 Dataset

Scenes: We choose 215 scenes from the Gibson dataset (Xia et al. 2018), with low clutter (Ramakrishnan et al. 2021) and no or limited furniture. Gibson scenes are loaded with AI Habitat simulator (Szot et al. 2021) and converted to top-down occupancy maps. The top-down occupancy maps are discrete 2D grids $\mathcal{M} \in \mathbb{Z}^{200 \times 150}$ with elements $m_{ij} \in \{-1, 0, 1\} \rightarrow$ {Innavigable, Unexplored Navigable, Explored Navigable} with a discretization of 0.1 meter per cell. The map size of 200×150 was empirically chosen. Scene metrics are presented in Table 1.

	nav-area (m^2)	nav-complexity	#scenes
large	66.09	19.36	88
medium	30.53	12.07	120
small	1.38	1.39	7

Table 1: Scene statistics: Navigable area[1](nav-area) and navigation complexity (nav-complexity), which is defined as the maximum ratio of geodesic and euclidean distance between any two navigable points in a scene, are shown for scenes in our large/medium/small dataset splits.

Objects and Receptacles: We randomly place objects and receptacles on the occupancy map. Object states contain only type and location information, therefore only distinction between objects is type. Agent interaction can only modify object locations p_i and there is no other parameter in the environment that changes during the episode. Objects and receptacles are not considered to occupy space on $\mathcal{M}$ to simplify collision computation. Multiple objects can be located in the same grid cell m_{ij}.

Episodes: Let $x^t : \mathbb{Z}^2$ and $\phi^t \in \{0, .., 7\} \rightarrow \{N, NE, E, SE, S, SW, W, NW\}$ be location and orientation of agent at time step t where {N:North, E:East, S:South, W:West}. At start of an episode, the agent is spawned at a random location x^0 and a random heading ϕ^0 (encoded as a 8-dimensional one-hot vector) as shown in Figure 1. The episode is terminated: (a) successfully, if all objects are in their receptacles before the time-step max_t, or (b) unsuccessfully, when the maximum episode time-step max_t is reached. Some objects may already be in their receptacles when the episode starts, but the agent must still verify this by at least seeing them in their receptacles for successful completion of goal g. Figure 2 shows a successful episode completion. We split the dataset into small/medium/large categories based on the area enclosed by the map. Each category contains train/validation/test splits. Episode statistics are presented in Table 4 in the Appendix.

3.3 Evaluation

For MORP, we are interested in measuring the agent's success rate in completing the task and the agent's efficiency, i.e., distance traveled/time taken for the task. In addition, to evaluate exploration methods, we use object discovery and

[1]We calculate navigable area using mesh calculated for agent with 1.5m height and 0.1m radius. Gibson dataset used to create top-down maps consists 1447 floors with 56 m^2 average navigable area. Comparing to Gibson floors our dataset consists less navigable area, since slicing a 3D scene into top-down map from a certain height does not necessarily capture whole floor area.

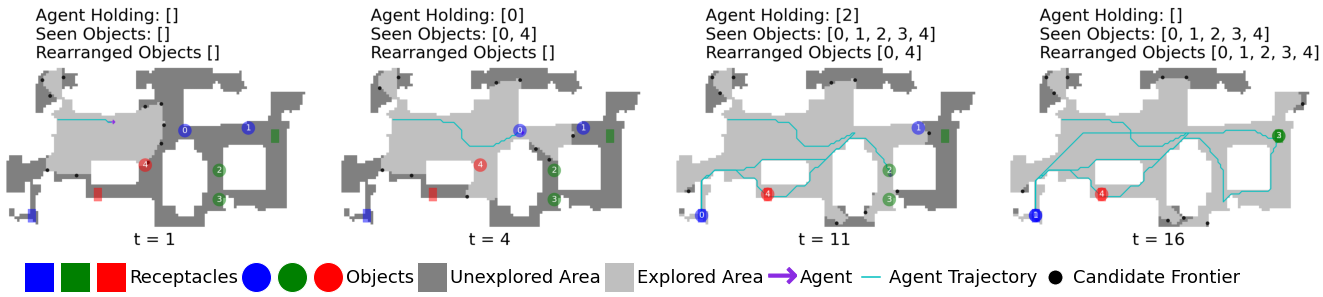

Figure 2: A sample episode for MORP – Agent finds misplaced objects, carries them to known receptacles locations and places them. Object ids for seen and rearranged objects as well as object held by the agent are shown.

map coverage metrics. We describe these metrics below:
Rearrangement:

- **Episode Success** (ES): If all objects are seen and rearranged, ES is 1 otherwise 0 (Kant et al. 2022).
- **Rearranged Object Ratio** (ROR): Ratio of the number of rearranged objects to the total number of objects that need to be rearranged.
- **Episodic Success Weighted by Path Length** ($ESPL$): In order to measure agent's efficiency, we compare path length and ES with oracle agent's path length. $ESPL$ is defined as

$$ESPL = ES\frac{z}{max(z,l)} \tag{1}$$

  where $z \in \mathbb{R}$ is the oracle agent's path length, and $l \in \mathbb{R}$ is the path length (Anderson et al. 2018). An oracle agent is essentially an agent that has full observability of the environment and uses CVRP planner to optimally perform rearrangement, similar to (Newman, Carlberg, and Desai 2020). We will describe this agent in more detail in Sec. 4.

Exploration:

- **Seen Object Ratio** (SOR): Ratio of the number of seen objects to the total number of objects.
- **Map Coverage** (MC): Ratio of the explored area to the total navigable area.

3.4 Agent Definition

Next, we define an agent, which is capable of sensing the environment and objects, navigating around, and rearranging objects.

Sensor Suite:

- Top-down Occupancy Map $\mathcal{M}$: We assume that a static top-down occupancy map of the environment $\mathcal{M}$ is available to the agent. Such a map could be obtained by a one-time scan of a house and thus may be a reasonable assumption for future home-assistant agents of any form.
- Receptacle Locations $\mathcal{R}$: Receptacle locations on the map $\mathcal{M}$ are available to agent.
- Agent Location x and Orientation ϕ: Agent can detect its location, x =(x,y) coordinates on $\mathcal{M}$ and orientation ϕ in 8 discrete directions.
- Field of View FOV: Agent can explore the map area and detect objects within its field-of-view determined by the conical sector (θ, r_s) where θ is angle around the orientation direction ϕ and r_s is the cut-off distance from x.
- Gripper $\mathcal{H} \in \mathbb{R}^{c \times n_o}$: This proprioceptive sensor contains flags indicating whether object j is being held by the agent's gripper slot i, where $c \in \mathbb{N}$ is number of the slots equivalent to agent's carrying capacity. All objects can be held by any slot and a slot can only hold one object at a time.

Action Space: The agent has three discrete actions: *forward*, *left* turn, and *right* turn for navigation and a single *grab/drop* action for object manipulation. The forward action moves the agent to one of 8 immediate grid cells in the map depending on its orientation ϕ. The left/right turn action changes ϕ by $45°$ in counter-clockwise/clockwise direction respectively. The grab/drop action is similar to "discrete object grasping" (Batra et al. 2020). Such action space $\mathcal{A}$ enables abstraction from downstream embodiment including parameters for continuous control of motors and allows us to focus on high-level task planning and exploration.

4 Complexity Analysis and Benchmarking

In this section, we present an analysis of factors that affect the complexity of rearrangement planning. Inspired by related work, we investigate various factors for our analysis, such as agent carrying capacity c (Batra et al. 2020), total number of objects n_o (Agia et al. 2022), number of receptacles n_r, and navigable area of the scene. To that end, we introduce heuristic and oracle agents and benchmark their performance using metrics introduced in Sec. 3 on the MORP as a way to analyze the complexity of MORP.

4.1 Oracle Agent

In order to find an upper bound on performance for MORP, we consider the *oracle* agent with access to privileged information. Specifically, the oracle agent has full state information and knows all object locations. Such full observability enables the agent to use a Capacitated Vehicle Routing Problem (CVRP) based approach to calculate shortest path length for rearranging objects, similar to (Newman, Carlberg, and Desai 2020). We describe the CVRP formulation in more detail in Appendix A.2. We solve the formulated planning problem using OR-Tools (Perron and Furnon 2022).

The oracle agent also leverages the full state information to ignore objects that are already arranged at the start of an episode. Such CVRP-based optimal path computation for only misplaced objects truly makes the oracle agent's performance an upper bound for multiple object rearrangement planning *without* any uncertainty. We next extend this oracle agent with exploration capability in a heuristic manner to deal with the uncertainty of object location in our setup.

4.2 Heuristic Agents

To benchmark the performance for multiple object rearrangement planning in presence of uncertainty (MORP), we consider *heuristic* agents. We leverage the intuition that MORP requires solving two sub-problems – (a) efficient exploration and search of the indoor environment to deal with location uncertainty of misplaced objects, (b) optimal path planning balanced with such object search to rearrange the misplaced objects. Based on this intuition, our heuristic agents use a modular approach that greedily combines classical optimal approaches for both of these sub-problems.

In particular, we use CVRP-based approach similar to the oracle agent for rearrangement planning of discovered objects and frontier-based exploration (FBE) approaches and its variants for efficient exploration and object discovery, based on the evidence of FBE's performance in classical robotics applications (Niroui et al. 2019). The agent greedily chooses *planning* if there are any seen objects that need to be rearranged, and *exploration* otherwise. Our approach of such greedy combination of these exploration and planning methods is also similar to recent work on house-scale rearrangement in EAI (Kant et al. 2022; Fukazawa et al. 2005).

One can also think of *planning* and *exploration* as high-level actions. Given one of these high-level actions, the agent executes a sequence of low-level navigation actions described in Sec. 3 until certain conditions are satisfied: (1) *planning* reaches the planned location, or (2) *exploration* reaches the target location, or (3) maximum high-level action distance max_{dist} is reached. These navigation actions are computed to follow a shortest path [2] between the agent's current location and the target *planned* or *exploration* location. The agent executes *grab/drop* action when it reaches a planned location and replans when new object(s) are discovered. Likewise, the agent chooses to explore if no objects have been discovered or if all the discovered objects are in their goal state and Rearranged Object Ratio (ROR) as described in Sec. 3 is less than 1. Such a modular approach of executing a low-level policy toward a goal given by a high-level policy is also inspired by recent work in embodied AI (Chaplot et al. 2020).

We now describe the variants of the FBE method and a random exploration approach, which we investigated for efficient exploration in our heuristic agents. Each time the agent engages the high-level action of *exploration*, a target location to be explored is obtained based on one of these approaches, described below.

[2]The shortest path is computed using the A* algorithm (Hart, Nilsson, and Raphael 1968).

Weighted Frontier Based Exploration FBE computes unexplored frontiers, which are the borderline grid-cells between explored and unexplored navigable area on a map. It then chooses the closest frontier location to the agent's location as the next location to visit for exploration (Yamauchi 1997). Recent variants of FBE referred to as weighted FBE (WFBE) use a utility function to determine next the frontier location to visit (Niroui et al. 2019). The utility function balances the potential information gain for exploration achieved by visiting a frontier location with the distance that needs to be traveled to reach the frontier location from the agent's current location. The information gain for a given frontier is defined as the sum of newly seen grid-cells along the shortest path between the agent's current location and the frontier[3]. Inspired by the performance of WFBE method, we investigate two variants of WFBE, which leverage two different utility functions:

- **WFBE$_\mathbf{r}$**: uses the ratio between frontier information gain and frontier distance as the utility function. It chooses a frontier that maximizes this function.
- **WFBE$_\mathbf{w}$**: uses weighted sum of normalized distance and normalized gain as the utility function, where the weight w is weight on distance and $(1-w)$ is the weight on gain. It then chooses the frontier that minimizes this function. We experiment with different values of the w: $0, 0.5, 1$ respectively.

Please refer to the Appendix A.3 for more details on the utility functions.

Random Exploration (RND) This approach picks a random navigable location from the unexplored area as the target location for exploration.

4.3 MORP Complexity Analysis

In order to analyze the effect of various factors such as agent carrying capacity c, total number of objects n_o, number of receptacles n_r, and navigable area of the scene etc., we consider different configurations of MORP and show the performance of the heuristic and oracle agents on these configurations. Specifically, we vary these factors over a range to create different configurations of MORP: $c \in \{1, 3\}$, $n_o \in \{1, 3, 5, 10\}$, $n_r \in \{1, 2, 3\}$. We present the empirical results in Table 2 with these different configurations. The "medium" and "large" dataset splits as described in Table 1 were used for these experiments. Note that we investigate both WFBE$_\text{r}$ and WFBE$_\text{w}$ with $w = \{0, 0.5, 1\}$, but we found WFBE$_\text{r}$ to work the best. We therefore use WFBE$_\text{r}$ for most of our conclusions. Next, we summarize our main findings on the effect of the various factors on MORP.

Higher agent capacity c makes MORP more challenging. In Table 2, we observe that all agents' performance decreases with the increase in agent capacity c. For instance, when the agent capacity increases from 1 to 3, on average ESPL drops by 10%. Such increase in MORP's complexity can be attributed to planning. Specifically, fig. 3 shows increase in planning time taken by the CVRP solver for oracle

[3]We find no difference in WFBE performance when the information gain is computed only at the frontier location instead.

			ESPL				
Explorer	c	n_o / n_r	1	3	5	10	*avg*
RND	1	3	.62	.69	.74	.82	.72
	3	3	.62	.63	.60	.62	.62
WFBE_1	1	3	.56	.67	.75	.83	.71
	3	3	.56	.62	.61	.64	.61
$\text{WFBE}_{0.5}$	1	3	**.68**	.75	**.81**	**.87**	**.78**
	3	3	**.68**	.69	**.65**	.67	.67
WFBE_0	1	3	.65	.71	.76	.84	.74
	3	3	.65	.65	.61	.65	.64
WFBE_r	1	1	.67	.79	.83	.91	.80
	3	1	.67	.60	.63	.65	.64
WFBE_r	1	3	.67	**.76**	**.81**	**.87**	**.78**
	3	3	.67	**.70**	**.65**	**.68**	**.68**
WFBE_r	1	5	.67	.76	.81	.88	.78
	3	5	.67	.70	.70	.71	.70

Table 2: MORP's complexity analysis as measured through the ESPL performance metric of heuristic agents: We measure the effect of number of objects n_o, number of receptacles n_r and agent capacity c on MORP. For all agents, other metrics were $MC = 0.83$, $ES = 1$ and $SOR = 1$ on average (detailed metrics are shown in Appendix A.4). Bold numbers indicate best performance in a particular MORP configuration.

agents with higher carrying capacity. Although we perform experiments with static agent capacity, practical instantiations of MORP such as house-cleaning might require dynamic capacity through use of containers etc., which may further increase the complexity of MORP.

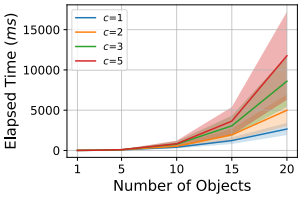

Figure 3: Scalability of oracle agent: time spent by CVRP solver for different values of c and n_o.

Higher number of objects reduces exploration complexity. When there are more objects in the environment, higher percentage of the object discovery happens while planning. For instance, average percentage of objects discovered during the planning $\{26\%, 43\%, 59\%\}$ increases with the total number of objects $\{3, 5, 10\}$ respectively. This reduces the burden on exploration for object search. Inspite of increase in planning complexity with higher number of objects (see fig. 3), such reduction in exploration complexity also reduces the overall problem complexity thereby leading to improved ESPL (Tab. 2). Specifically, when the number of objects increases from 1 to 10, on average ESPL for agent with $c = 1$ increases by 20%. Such performance improvement in ESPL however disappears with increase in agent capacity e.g., no significant correlation between n_o and ESPL is observed for $c = 3$. This highlights complex interplay between exploration and planning complexity for MORP.

Higher number of receptacles $n_r > 1$ reduces MORP's complexity only when the agent has higher capacity $c > 1$. In Table 2, we see that increase in n_r has different effects on performance of agents with different c. When n_r is increased from 1 to 5, we observe that on average ESPL for $c = 1$ decreases by 2%, whereas ESPL for $c = 3$ increases by 6%. It suggests that for episodes with multiple receptacles ($n_r > 1$), there are scenarios where an object's receptacle is within the close proximity of another object. The agents can exploit such scenarios to reduce navigated distance and thereby ESPL for MORP.

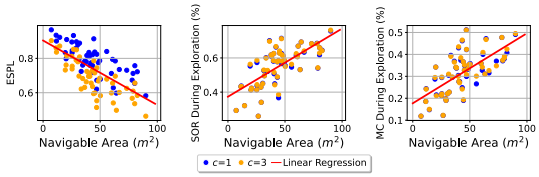

Figure 4: Effect of navigable area on performance as measured by ESPL for WFBE_r exploration policy.

Greater navigable area worsens the performance. Figure 4 shows how WFBE_r exploration policy performance is negatively correlated with navigable area. Similar trends were observed with other exploration strategies. Overall, exploration strategy matters more for larger areas and thus affects performance. We didn't find any correlation between navigation complexity of the scenes (as described in Sec. 3) and performance.

4.4 Agent Benchmarking

We next elaborate on the analysis and failure modes of the classical approaches that we used for exploration and planning in our heuristic and oracle agents. Specifically, we investigate the scaling of the CVRP solver, its applicability to real-time applications, and the performance of different exploration policies.

Planning In Figure 3, we see that the time spent by oracle agent to solve the CVRP problem increases exponentially w.r.t. n_o. Figure 3 also shows that increasing c makes the planning more complex. Since our focus is not on solvers, we choose a CVRP solver configuration[4] from OR-Tools (Perron and Furnon 2022) that can find optimal solution under real-time constraints for our experiments. Based on the exponential increase in compute time for CVRP problems, we recommend using satisficing or learning-based planners e.g., (Agia et al. 2022) to tackle larger-scale planning problems in the future.

Exploration We find that agents with WFBE_r and WFBE_{50} exploration policy outperform all other agents, which suggests that one could potentially learn to find an optimal balance between gain and distance. All other exploration approaches exhibit varied failure modes. For instance,

[4]CVRP solver: RoutingModel, FirstSolutionStrategy: PARALLEL-CHEAPEST-INSERTION, solution-limit:50.

the agents with RND and WFBE_0 exploration policies start picking locations at the opposite sides of the map consecutively and keep going back and forth after exploring for a while. This prevents these agents from exploring unexplored areas. Consequently, the agents fail to discover objects and complete the task within maximum episode time max_t. Although intuitively choosing the closest frontier location aka WFBE_{100} or frontier with maximum gain i.e., WFBE_0 perform better than agents with RND exploration policy, they perform worse than the agents that consider both gain and distance aka agents with WFBE_r and WFBE_{50}. Based on these observations, we also propose WFBE_r and its performance on MORP as a baseline for benchmarking future research on MORP. Please see the Appendix for detailed comparison on various exploration approaches using all the metrics defined in Sec. 3.

5 Learning-based Agents

Inspired by the limitations of our heuristic agents and their classical exploration and planning approaches, we explore learning-based agents to obtain policies better than that of the heuristic agents for MORP. In particular, we investigate end-to-end approaches aka monolithic RL for MORP. In addition, we also experiment with ways to improve our heuristic agents. We describe these learning-based agents in detail below.

Monolithic agents: These agents leverage monolithic deep reinforcement learning approach such as (Karkus et al. 2020) to learn direct mapping from observations to actions.

- **End-to-End Planner (E2E-P)** learns direct mapping from observations to low-level actions – *forward*, *left*, *right*, *grab/drop*. E2E-P agent thus learns to navigate, explore, and then to accomplish rearrangement from scratch.
- **Where-to-Go Planner (W2G-P)** learns where to go on the occupancy map $\mathcal{M}$. Instead of low-level navigation actions of *forward*, *left*, *right*, W2G-P agent uses navigable cells from $\mathcal{M}$ as actions in combination with the *grab/drop* action. Once target cell from $\mathcal{M}$ is chosen as the action, we obtain the low-level navigation actions to follow the shortest path between the agent's current location and the chosen cell as described in Sec. 4.2. Unlike E2E-P that has to learn low-level navigation in combination with exploration and planning for MORP, the W2G-P focuses on only learning exploration and planning.

Modular agents: To improve our modular heuristic agents that greedily combine WFBE-based exploration and CVRP-based planning (Sec. 4), we investigate ways to a) improve exploration performance, b) improve the trade-off between exploration and planning beyond greedy, and c) jointly improve both.

- **Learnt Explorer (LE)** is focused on improving exploration performance of WFBE methods in our heuristic agents, inspired by the performance difference between these agents in Table 2. Specifically, we learn a utility function approximation for frontier selection in WFBE, which trades-off information gain and frontier distance. We train two types of LE agents: (1) LE_{disc} that chooses directly among candidate frontiers, in particular, these candidates correspond to the agent's discrete actions and (2) LE_w (Niroui et al. 2019) that outputs a continues variable $w \in \mathbb{R} : [0, 1]$ to be used as a weight on normalized distance and normalized information gain in WFBE_w exploration policy. See Appendix A.3 for more details on WFBE_w and candidate frontier computation. Contrary to WFBE_w, LE_w dynamically changes the w at each step rather than using a fixed w. We train LE agents explicitly on the exploration task of finding all objects but not rearranging them for the MORP dataset. We then evaluate the LE agents on MORP by combining them with CVRP planner, akin to the heuristic agents.
- **Optimal Balancer (OB)** agent learns to combine WFBE-based exploration and CVRP-based planning optimally, instead of greedily choosing between them as in the heuristic agents. OB agent thus has two corresponding high-level actions: *explore* and *plan* that it learns to choose from.
- **Balanced Explorer (BE)** learns exploration policy and optimal balance between *exploration* and *planning* jointly. BE agent's discrete actions thus consists of *plan* action, which calls the CVRP planner; and actions that map to candidate frontier locations.

For all of the above agents, the input consists of occupancy map, receptacle locations, agent location, total number of objects to be rearranged, and agent's current gripper state as described in agent's sensor suite (see Sec. 3). For modular agents, we additionally use frontier locations as input. More details on the representations of these inputs for individual agents are described in Appendix A.5. For the policy architecture, we use ConvLSTM-like architecture for E2E-P and W2G-P agents and ConvMLP-like architecture[5] for OB, LE, and BE agents. The reward function for all agents is the weighted sum of navigated distance, the number of newly seen objects, grab/drop reward, the newly discovered map area at time step t, and the episode success reward. Please see the Appendix A.5 for more details.

5.1 Training Process

We use Rllib (Liang et al. 2018) implementation of DD-PPO (Wijmans et al. 2020) in order to train our agents with train splits of the dataset. E2E-P and W2G-P agents are trained with a naive curriculum, where we first train with single object rearrangement episodes before training with MORP episodes. The "small" dataset split was used to train E2E-P and W2G-P. All other agents were trained using the "medium" and "large" dataset splits (see Table 1). Details on task configuration including hyperparameters for training and benchmarking can be found in Appendix A.5.

[5]Since our inputs for OB, LE, and BE contain all the information pertaining to the sufficient state for MORP, recurrent policies are not needed. We verified this empirically by swapping the MLP layer with the recurrent LSTM layer and found no difference in ESPL of these agents.

5.2 Benchmarking Learning-based Agents

We evaluate the performance of learning-based agents and compare them against the best performing heuristic agent WFBE_r in Table 3. We also present our conclusions on leveraging learning to obtain policies for MORP.

		$ESPL$				$\Delta ESPL$	
Methods	c \ n_o	1	3	5	10	μ	σ
WFBE_r	1	.67	**.76**	**.81**	**.87**	-	-
	3	.67	**.70**	**.65**	**.68**	-	-
LE_w	1	**.68**	.72	.75	.85	.029	.15
	3	**.68**	**.70**	**.65**	**.68**	.002	.15
LE_disc	1	.64	.69	.72	.82	.062	.19
	3	.64	.67	.62	.65	.033	.20
OB	1	**.68**	**.76**	**.81**	**.87**	.007	.05
	3	**.68**	**.70**	**.65**	**.68**	.006	.05
BE	1	.58	.65	.72	.77	.097	.19
	3	.58	.60	.58	.58	.090	.18

Table 3: Comparison of heuristic and learning-based agents. $\Delta ESPL$ shows mean and variance for $ESPL$ difference between best performing heuristic agent (WFBE_r) and other agents. For all agents, other metrics on average are: $MC = 0.83$, $ES = 1$ and $SOR = 1$. Bold numbers indicate best performance in a particular MORP configuration.

Learning rearrangement planning from scratch is hard. We find that both E2E-P and W2G-P fail in completing even single object rearrangement episodes. They learn to discover the object and can move towards the object, yet they fail to grab the object or to rearrange it into the receptacle. DRL approaches are known to struggle in long-horizon and sparse reward settings (Matheron, Perrin, and Sigaud 2020) as is the case with E2E-P. Likewise, DRL approaches struggle with large discrete actions spaces (Dulac-Arnold et al. 2015) such as that of W2G-P. Prior work on object rearrangement therefore uses modular approaches over monolithic ones (Karkus et al. 2020; Szot et al. 2021). We do not include E2E-P and W2G-P metrics in Table 3 since these agents do not succeed in any MORP episodes.

Learning exploration explicitly or jointly with rearrangement planning does not help MORP. We first compare agents that explicitly learn to explore (LE_w and LE_disc) with WFBE-based exploration on purely exploration task of finding all objects in MORP scenes (see Table 8 in Appendix A.5 for detailed comparison of the same). We find that learnt exploration agents do not perform better than the WFBE policies when evaluated on this exploration task. Agents with short-sighted frontier-based exploration policies e.g., WFBE_r and WFBE_w agents, which try to find objects by exploring most area while navigating minimum distance, perform better than the learnt exploration policies that consider the long-term and minimize total path length in order to discover all the objects in LE_w and LE_disc agents. Consequently, combining them with optimal planning using the CVRP solver for rearrangement (LE_w, LE_disc in Table 3) does not lead to improvement in MORP. This suggests that on average, finding the first object fast and counting on the object discovery during rearrangement planning is a good strategy for MORP. Since LE agents explicitly learn to explore without accounting for rearrangement planning, we also train the BE agent which learns to combines exploration with CVRP planning, while jointly learning to explore. However, we find that this doesn't improve performance on MORP either.

The *greedy combination* of frontier-based exploration and CVRP-based planning is *empirically* optimal. During the training of OB agent, we intermittently evaluate the agent on the test dataset in order to understand the evolution of agent behavior over the training process. Our evaluations indicate that in the early stages of the training, OB acts non-greedily as opposed to the heuristic WFBE_r agent. Yet, despite hyper-parameter tuning such as that of the entropy loss coefficient in DD-PPO and reward shaping, OB eventually converges to a greedy behaviour. This is evident in $\Delta ESPL$ between OB's performance and WFBE_r agent's performance in Table 3, which is similar to a zero-centered normal distribution with 5% standard error. Post convergence, OB's high entropy action policies have occasional attempts (10% of the test data) to explore non-greedily i.e., explore even when there are discovered objects that need to be rearranged. Such non-greedy behavior enables OB to discover objects earlier than the WFBE_r agent in certain episodes. However, on average OB's behavior is greedy. This empirically demonstrates that the greedy combination of frontier-based exploration and CVRP-based planning (as in WFBE agents) is optimal for MORP.

In summary, E2E-P and W2G-P demonstrate that conventional, monolithic deep RL does not succeed at MORP since the agent needs to learn navigation, exploration, and planning from scratch. Modular agents that leverage the inductive biases of conventional frontier-based exploration and CVRP-based planning in combination with learning to improve exploration and/or the trade-off between exploration and planning still do not outperform their greedy, heuristic counterparts.

6 Conclusion

We propose Multiple Object Rearrangement Planning (MORP) in mapped environments with partial observability over object locations as a benchmark task. We conduct a comprehensive complexity analysis of MORP, where we investigate various factors such as number of objects and receptacles, agent carrying capacity, environment layouts etc. that make MORP challenging. We further introduce competitive heuristic baselines for MORP that greedily combine classical frontier-based exploration and optimization-based planning. We also train reinforcement learning policies for MORP in order to outperform the heuristic baselines. However, we find that monolithic RL policies struggle at MORP while modular RL policies converge to behaviors similar to heuristic policies. This empirically demonstrates that greedy combination of exploration and planning for MORP is optimal when objects to be rearranged are uniformly distributed in the 3D environments. However, developing agents that outperform heuristic agents at MORP remains an open problem for future research.

Acknowledgements

We'd like to thank Kevin Carlberg and Roberto Calandra for many helpful discussions on this problem.

References

Agia, C.; Jatavallabhula, K. M.; Khodeir, M.; Miksik, O.; Vineet, V.; Mukadam, M.; Paull, L.; and Shkurti, F. 2022. Taskography: Evaluating robot task planning over large 3D scene graphs. In Faust, A.; Hsu, D.; and Neumann, G., eds., *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, 46–58. PMLR.

Anderson, P.; Chang, A. X.; Chaplot, D. S.; Dosovitskiy, A.; Gupta, S.; Koltun, V.; Kosecka, J.; Malik, J.; Mottaghi, R.; Savva, M.; and Zamir, A. R. 2018. On Evaluation of Embodied Navigation Agents. *CoRR*, abs/1807.06757.

Andreas, J.; Klein, D.; and Levine, S. 2016. Modular Multitask Reinforcement Learning with Policy Sketches.

Batra, D.; Chang, A. X.; Chernova, S.; Davison, A. J.; Deng, J.; Koltun, V.; Levine, S.; Malik, J.; Mordatch, I.; Mottaghi, R.; Savva, M.; and Su, H. 2020. Rearrangement: A Challenge for Embodied AI. *CoRR*, abs/2011.01975.

Ben-Shahar, O.; and Rivlin, E. 1998. Practical pushing planning for rearrangement tasks. *IEEE Transactions on Robotics and Automation*, 14(4): 549–565.

Chaplot, D. S.; Gandhi, D.; Gupta, S.; Gupta, A.; and Salakhutdinov, R. 2020. Learning to Explore using Active Neural SLAM. *CoRR*, abs/2004.05155.

Chen, T.; Gupta, S.; and Gupta, A. 2019. Learning Exploration Policies for Navigation. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Dulac-Arnold, G.; Evans, R.; Sunehag, P.; and Coppin, B. 2015. Reinforcement Learning in Large Discrete Action Spaces. *CoRR*, abs/1512.07679.

Fukazawa, Y.; Chomchana, T.; Ota, J.; Yuasa, H.; Arai, T.; Asama, H.; and Kawabata, K. 2005. Realizing the exploration and rearrangement of multiple unknown objects by an actual mobile robot. *Advanced Robotics*, 19(1): 1–20.

Garrett, C. R.; Chitnis, R.; Holladay, R.; Kim, B.; Silver, T.; Kaelbling, L. P.; and Lozano-Pérez, T. 2021. Integrated task and motion planning. *Annual review of control, robotics, and autonomous systems*, 4: 265–293.

Hall, D.; Talbot, B.; Bista, S. R.; Zhang, H.; Smith, R.; Dayoub, F.; and Sünderhauf, N. 2020. The Robotic Vision Scene Understanding Challenge. *CoRR*, abs/2009.05246.

Hart, P.; Nilsson, N.; and Raphael, B. 1968. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2): 100–107.

Huang, E.; Jia, Z.; and Mason, M. T. 2019. Large-Scale Multi-Object Rearrangement. In *2019 International Conference on Robotics and Automation (ICRA)*, 211–218.

Kant, Y.; Ramachandran, A.; Yenamandra, S.; Gilitschenski, I.; Batra, D.; Szot, A.; and Agrawal, H. 2022. Housekeep: Tidying Virtual Households using Commonsense Reasoning. arXiv:2205.10712.

Kapelyukh, I.; and Johns, E. 2022. My house, my rules: Learning tidying preferences with graph neural networks. In *Conference on Robot Learning*, 740–749. PMLR.

Karkus, P.; Mirza, M.; Guez, A.; Jaegle, A.; Lillicrap, T.; Buesing, L.; Heess, N.; and Weber, T. 2020. Beyond Tabula-Rasa: a Modular Reinforcement Learning Approach for Physically Embedded 3D Sokoban.

Kim, U.-H.; Kim, Y.-H.; Park, J.-M.; Choi, H.-S.; and Kim, J.-H. 2021. Learning to Explore, Navigate and Interact for Visual Room Rearrangement.

King, J. E.; Cognetti, M.; and Srinivasa, S. S. 2016. Rearrangement planning using object-centric and robot-centric action spaces. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 3940–3947.

King, J. E.; Ranganeni, V.; and Srinivasa, S. S. 2017. Unobservable monte carlo planning for nonprehensile rearrangement tasks. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 4681–4688. IEEE.

Liang, E.; Liaw, R.; Nishihara, R.; Moritz, P.; Fox, R.; Goldberg, K.; Gonzalez, J. E.; Jordan, M. I.; and Stoica, I. 2018. RLlib: Abstractions for Distributed Reinforcement Learning. In *International Conference on Machine Learning (ICML)*.

Liu, R.; Lehman, J.; Molino, P.; Such, F. P.; Frank, E.; Sergeev, A.; and Yosinski, J. 2018. An Intriguing Failing of Convolutional Neural Networks and the CoordConv Solution. *CoRR*, abs/1807.03247.

Lloyd, S. 1982. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2): 129–137.

Matheron, G.; Perrin, N.; and Sigaud, O. 2020. Understanding Failures of Deterministic Actor-Critic with Continuous Action Spaces and Sparse Rewards. In Farkaš, I.; Masulli, P.; and Wermter, S., eds., *Artificial Neural Networks and Machine Learning – ICANN 2020*, 308–320. Cham: Springer International Publishing. ISBN 978-3-030-61616-8.

Newman, B.; Carlberg, K.; and Desai, R. 2020. Optimal Assistance for Object-Rearrangement Tasks in Augmented Reality. *CoRR*, abs/2010.07358.

Niroui, F.; Zhang, K.; Kashino, Z.; and Nejat, G. 2019. Deep Reinforcement Learning Robot for Search and Rescue Applications: Exploration in Unknown Cluttered Environments. *IEEE Robotics and Automation Letters*, 4(2): 610–617.

Perron, L.; and Furnon, V. 2022. OR-Tools. Google.

Ramakrishnan, S. K.; Gokaslan, A.; Wijmans, E.; Maksymets, O.; Clegg, A.; Turner, J. M.; Undersander, E.; Galuba, W.; Westbury, A.; Chang, A. X.; Savva, M.; Zhao, Y.; and Batra, D. 2021. Habitat-Matterport 3D Dataset (HM3D): 1000 Large-scale 3D Environments for Embodied AI. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Ramakrishnan, S. K.; Jayaraman, D.; and Grauman, K. 2020. An Exploration of Embodied Visual Exploration. *CoRR*, abs/2001.02192.

Ronneberger, O.; Fischer, P.; and Brox, T. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. *CoRR*, abs/1505.04597.

Szot, A.; Clegg, A.; Undersander, E.; Wijmans, E.; Zhao, Y.; Turner, J.; Maestre, N.; Mukadam, M.; Chaplot, D.; Maksymets, O.; Gokaslan, A.; Vondrus, V.; Dharur, S.; Meier, F.; Galuba, W.; Chang, A.; Kira, Z.; Koltun, V.; Malik, J.; Savva, M.; and Batra, D. 2021. Habitat 2.0: Training Home Assistants to Rearrange their Habitat. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Weihs, L.; Deitke, M.; Kembhavi, A.; and Mottaghi, R. 2021. Visual Room Rearrangement. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, 5922–5931. Computer Vision Foundation / IEEE.

Wijmans, E.; Kadian, A.; Morcos, A.; Lee, S.; Essa, I.; Parikh, D.; Savva, M.; and Batra, D. 2020. DD-PPO: Learning Near-Perfect PointGoal Navigators from 2.5 Billion Frames. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Xia, F.; R. Zamir, A.; He, Z.-Y.; Sax, A.; Malik, J.; and Savarese, S. 2018. Gibson env: real-world perception for embodied agents. In *Computer Vision and Pattern Recognition (CVPR), 2018 IEEE Conference on*. IEEE.

Yamauchi, B. 1997. A frontier-based approach for autonomous exploration. In *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97. 'Towards New Computational Principles for Robotics and Automation'*, 146–151.

Yamauchi, B. 1998. Frontier-based exploration using multiple robots. In *Proceedings of the second international conference on Autonomous agents*, 47–53.

A Appendix

A.1 Dataset

Table 4 shows the number of scenes and episodes for the small, medium and large variants of our dataset. Figure 5 visualizes multiple episodes in our dataset variants with varying number of objects and receptacles.

		#scenes	#episodes			
	n_o	-	1	3	5	10
large	train	44	4.4k	4.4k	4.4k	4.4k
	test	22	220	220	220	220
medium	train	60	6k	6k	6k	4843
	test	30	300	300	299	231
small	train	5	500	-	-	-
	test	1	10	-	-	-

Table 4: Episode and scene counts in the dataset.

A.2 Capacitated Vehicle Routing Problem

In this section, we describe the CVRP formulation employed by the oracle agent. Let $\mathbb{O} = \{0, ..., n_s - 1\}$ denote the set of n_s objects seen and yet to be rearranged. Then the set of object pickup locations and their corresponding receptacle locations yields $2n_s + 1$ locations in total including the agent's initial location x. We order these locations with indices $\{0, ..., 2n_s\}$ such that the zeroth index corresponds to x, every odd index $2i+1$ corresponds to the pickup location in $\mathcal{P}$ and every even index $2i+2$ corresponds to the dropoff location in $\mathcal{B}$ for object $i \in \mathbb{O}$. We denote the set of pickup locations as $\Gamma = \{1, 3, ..., 2n_s - 1\}$ and the set of dropoff locations as $\Psi = \{2, 4, ..., 2n_s\}$.

Let $\tau : \{0, ..., 2n_s\} \times \{0, ..., 2n_s\} \rightarrow \mathbb{R}$ be the transportation cost function which calculates the geodesic distance between two navigable locations given the location indices and satisfies $\tau(i, i) = 0$. We define $\varphi : \{0...2n_s\} \rightarrow \{0...2n_s\}$ as an invertible operator that maps a step number to a location index to visit. The operator φ is the solution variable for our oracle planner, so the optimal sequence of location visits is given by $(\varphi^*(0), ..., \varphi^*(2n_s))$ and is computed as the solution to a combinatorial optimization problem. Invertibility is assumed for φ since every location is visited exactly once and each visit (except the zeroth location x) is implicitly tied to either an object pickup or dropoff. Let $\mathbf{1_I}(\cdot)$ be the indicator function to check if its argument is in the set $\mathbf{I}$. The CVRP optimization problem is then framed as:

$$\min_{(\varphi(0),...\varphi(2n_s))} \sum_{i=1}^{2n_s} \tau(\varphi(i), \varphi(i-1)) \tag{2a}$$

$$\text{s.t. } \varphi(0) = 0 \tag{2b}$$

$$\sum_{i=1}^{j} \mathbf{1}_\Gamma(\varphi(i)) - \mathbf{1}_\Psi(\varphi(i)) \leq c, \forall j \in \{1, ..., 2n_s\} \tag{2c}$$

$$\varphi^{-1}(i+1) > \varphi^{-1}(i), \forall i \in \Gamma \tag{2d}$$

$$\varphi^{-1}(2i+1) \leq ||\mathcal{H}||_1, \forall i \in \{\mathbb{O} \mid \ ||\mathcal{H}_{:,i}||_1 = 1\} \tag{2e}$$

The objective function in eq (2a) corresponds to the total distance traveled. Eq (2b) states that the initial location of the agent should be the first location visited. Eq (2c) expresses the capacity constraint for the agent. Eq (2d) indicates that the agent should pickup the object before dropping it to its receptacle. Eq (2e) use the gripper slot representation $\mathcal{H} \in \mathbb{R}^{c \times n_o}$ defined in Section 3.4 to maintain consistency for held objects when re-planning occurs and is trivially satisfied if no object is being held. It states that the pickup for held objects must happen before others. But since they are already being held in the environment, no actual re-pickup needs to be performed during plan execution.

A.3 Weighted Frontier Based Exploration

We detail our implementation of WFBE along with the utility function definitions in this section.

In our implementation, we first compute the FBE frontiers and then cluster them into K clusters[6] using K-means clustering (Lloyd 1982). Denoting the set $\{1, \ldots, K\}$ by $[K]$, let $\mathcal{V}$ be the set of K frontier locations with element $v_k \in \mathbb{Z}^2$ being the frontier location closest to the centroid of cluster $k \in [K]$. Define d_k to be the shortest path length between v_k and x. Further, let $gain_k \in \mathbb{Z}$ be the gain for the candidate frontier v_k defined as the sum of newly seen cells along the shortest path between x and v_k.

We explore two choices to calculate the utility function for choosing the next frontier:

- **WFBE$_\mathbf{r}$**: Chooses the candidate frontier which maximizes the ratio between gain and distance of the frontier from x,

$$choice = \arg\max_{k \in [K]} \left\{ \frac{gain_k}{d_k} \right\} \tag{3}$$

- **WFBE$_\mathbf{w}$**: Chooses the candidate frontier which minimizes a weighted sum of normalized gain and normalized distance from x,

$$choice = \arg\min_{k \in [K]} \{w * \bar{d}_k + (1 - w) * \overline{gain}_k\}, \tag{4}$$

$$\text{where } \bar{d}_k = \frac{d_k}{\sum_{j \in [K]} d_j}, \overline{gain}_k = \frac{gain_k}{\sum_{j \in [K]} gain_j},$$

and w is a balancing hyperparameter (Niroui et al. 2019).

A.4 Complexity analysis of MORP

Table 5 shows the performance metrics for heuristic agents when the number of objects n_o, number of receptacles n_r and the agent capacity c vary.

A.5 Learning-Based Agents Implementation Details

In this section, we describe the policy neural network architectures, reward functions and training hyperparameters for our learning-based agents.

[6] We empirically chose the number of clusters K to be 10.

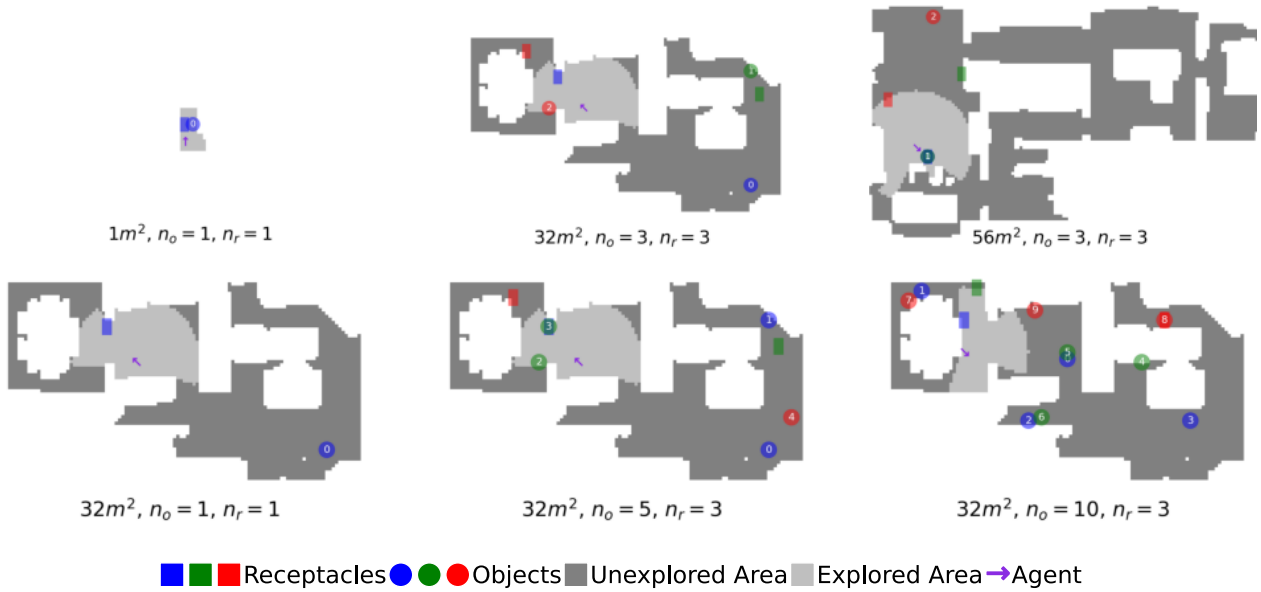

Figure 5: Instantiation of multiple episodes. The top row shows different scenes that belong to small/medium/large splits of our dataset. The bottom row shows the instantiation of a single scene with varying number of objects n_o and receptacles n_r.

Inputs We first define a common set of inputs for learning-based agents. The inputs to individual agents are shown in Table 6.

- Agent Location Map: A matrix $\mathcal{M}_a \in \mathbb{Z}^{200\times150}$ with all entries 0s, except the agent location which is a 1.
- Expected Number of Objects Map: $\mathcal{M}_o : \mathbb{R}^{n_r\times200\times150}$ which contains the expected number of objects of type i in the i-th slice $\mathcal{M}_{o,i,:,:}$. The matrix $\mathcal{M}_o$ is initialized with zeros. Every seen object of type i contributes a further additive value of 1 and unseen type i objects contribute the additive value $\frac{1}{\text{\# of unseen navigable cells}}$ to the i-th slice.
- Final Object Locations Map: $\mathcal{M}_b : \mathbb{R}^{n_r\times200\times150}$ with the i-th slice $\mathcal{M}_{b,i,:,:}$ containing the number of type i objects at the i-th receptacle location and zero otherwise.
- Candidate Frontier Map: $\mathcal{M}_v \in \mathbb{Z}^{200\times150}$ with candidate frontier locations represented as ones and all other locations as zeros.
- Candidate Frontier Distances and Gains: $\mathcal{D} = \{d_{v_i}, \forall v_i \in \mathcal{V}\}$ distances and $\mathcal{K} = \{gain_{v_i}, \forall v_i \in \mathcal{V}\}$ gains associated with candidate frontiers.
- Exploration Location Map: $M_e : \mathbb{Z}^{200\times150}$ which represents the selected exploration location with 1 and all other locations with 0.
- Planning Location Map: $M_p : \mathbb{Z}^{200\times150}$ which represents the next planned object/receptacle location as 1 and all other locations as 0.
- Gripper Load: $g_l = \frac{||\mathcal{H}||_1}{c}$, continuous representation of gripper load in $[0, 1]$.

Policy Architectures In order to present policy architectures we define individual and combined layers as follows:

- *conv1:* 8 (4×4) kernels, padding:1, stride:2 + ReLU.
- *conv2:* 16 (4×4) kernels, padding:1, stride:2 + ReLU.
- *conv3:* 16 (3×3) kernels, padding:1, stride:2 + ReLU.
- *conv4:* 8 (3×3) kernels, padding:1, stride:2 + ReLU.
- *conv5:* 4 (3×3) kernels, padding:1, stride:1 + ReLU.
- *conv6:* 2 (2×2) kernels, padding:1, stride:1 + ReLU.
- *conv7:* 32 (3×3) kernels, padding:1, stride:1 + ReLU.
- *flat1:* FC layer from 1900-dim to 512-dim + ReLU.
- *flat2:* FC layer from 3800-dim to 512-dim + ReLU.
- *concat1:* FC layer from 1545-dim to 256-dim + ReLU.
- *concat2:* FC layer from 1609-dim to 256-dim + ReLU.
- *concat3:* LSTM layer from 1549-dim to 256-dim.
- *concat4:* LSTM layer from 1613-dim to 256-dim.
- *concat5:* FC layer from 585-dim to 256-dim + ReLU.
- *concat6:* LSTM layer from 521-dim to 256-dim.
- *act-emb:* Embeds action into 4-dim cont. space.
- *dir-emb:* Embeds direction into 8-dim cont. space.
- *info:* FC layer from 20-dim to 64-dim + ReLU.
- *c1:* $flat1(conv5(conv4(conv4(conv1(\cdot)))))$
- *c2:* $flat1(conv5(conv4(conv3(conv2(\cdot)))))$
- *c3:* $flat2(conv4(conv3(conv7(conv2(\cdot)))))$
- *act:* Action head FC layer from 256-dim to actions.
- *cri:* Critic head FC layer from 256-dim to 1-dim.

Learning-based agents' policy architectures are as follows:

			MC				SOR				ROR				$ESPL$				
Explorer	c	n_r \ n_o	1	3	5	10	1	3	5	10	1	3	5	10	1	3	5	10	*avg*
RND	1	3	.66	.84	.89	.94	1	1	1	1	1	1	1	1	.62	.69	.74	.82	.72
	3	3	.66	.84	.89	.94	1	1	1	1	1	1	1	1	.62	.63	.60	.62	.62
WFBE_1	1	3	.66	.85	. 89	.94	1	1	1	1	1	1	1	1	.56	.67	.75	.83	.71
	3	3	.66	.85	.87	.94	1	1	1	1	1	1	1	1	.56	.62	.61	.64	.61
$\text{WFBE}_{0.5}$	1	3	.65	.84	.88	.94	1	1	1	1	1	1	1	1	**.68**	.75	**.81**	**.87**	**.78**
	3	3	.65	.84	.89	.94	1	1	1	1	1	1	1	1	**.68**	.69	**.65**	.67	.67
WFBE_0	1	3	.65	.84	.89	.94	1	1	1	1	1	1	1	1	.65	.71	.76	.84	.74
	3	3	.65	.84	.89	.93	1	1	1	1	1	1	1	1	.65	.65	.61	.65	.64
WFBE_r	1	3	.65	.84	.89	.94	1	1	1	1	1	1	1	1	.67	**.76**	**.81**	**.87**	**.78**
	3	3	.65	.84	.89	.93	1	1	1	1	1	1	1	1	.67	**.70**	**.65**	**.68**	**.68**
LE_w	1	3	.65	.80	.87	.94	1	1	1	1	1	1	1	1	**.68**	.72	.75	.85	.75
	3	3	.65	.80	.87	.94	1	1	1	1	1	1	1	1	**.68**	**.70**	**.65**	**.68**	**.68**
LE_disc	1	3	.65	.79	.87	.94	1	1	1	1	1	1	1	1	.67	.64	.69	.72	.68
	3	3	.65	.79	.87	.94	1	1	1	1	1	1	1	1	.67	.64	.67	.62	.65
OB	1	3	.65	.84	.89	.94	1	1	1	1	1	1	1	1	**.68**	**.76**	**.81**	**.87**	**.78**
	3	3	.65	.84	.89	.94	1	1	1	1	1	1	1	1	**.68**	**.70**	**.65**	**.68**	**.68**
BE	1	3	.66	.85	.90	.94	1	1	1	1	1	1	1	1	.58	.65	.82	.77	.68
	3	3	.66	.85	.89	.94	1	1	1	1	1	1	1	1	.58	.60	.58	.58	.59

Table 5: MORP's complexity analysis as measured through the ESPL performance metric of heuristic agents: We measure the effect of number of objects n_o, number of receptacles n_r and agent capacity c on MORP.

- **LE_w, LE_{disc}:** *concat5*([*c3*($\mathcal{M}, \mathcal{M}_a, \mathcal{M}_o, \mathcal{M}_v$), *dir-emb*($\phi$), *info*([$\mathcal{K}, \mathcal{D}$])]).
- **E2E-P:** *concat6*([*c3*($\mathcal{M}, \mathcal{M}_a, \mathcal{M}_o, \mathcal{M}_b$), *dir-emb*($\phi$), g_l)]).
- **W2G-P:** We use E2E-P network to obtain state representation and 200×150 top-down map, action logits, is reconstructed by using *up-sampling* similar to U-Net (Ronneberger, Fischer, and Brox 2015) combined with *coordinate convolution* (Liu et al. 2018).
- **OB(MLP):** *concat1*([*c1*($\mathcal{M}$), *c2*($\mathcal{M}_b, \mathcal{M}_e, \mathcal{M}_p$), *c1*($\mathcal{M}_o, \mathcal{M}_b$), *dir-emb*($\phi$), g_l]).
- **OB(LSTM):** *concat3*([*c1*($\mathcal{M}$), *c2*($\mathcal{M}_b, \mathcal{M}_e, \mathcal{M}_p$), *c1*($\mathcal{M}_o, \mathcal{M}_b$), *dir-emb*($\phi$), *act-emb*(*prev-action*), g_l]).
- **BE(MLP):** *concat2*([*c1*($\mathcal{M}$), *c2*($\mathcal{M}_b, \mathcal{M}_v, \mathcal{M}_p$), *c1*($\mathcal{M}_o, \mathcal{M}_b$), *dir-emb*($\phi$), g_l, *info*([$\mathcal{K}, \mathcal{D}$])]).
- **BE(LSTM):** *concat4*([*c1*($\mathcal{M}$), *c2*($\mathcal{M}_b, \mathcal{M}_v, \mathcal{M}_p$), *c1*($\mathcal{M}_o, \mathcal{M}_b$), *dir-emb*($\phi$), *act-emb*(*prev-action*), g_l, *info*([$\mathcal{K}, \mathcal{D}$])]).

We feed output of *concatX* layers into *act* and *cri* layers in order to produce the actor and the critic in the DD-PPO algorithm.

Reward Functions The reward function for all agents is the weighted sum of navigated distance, the number of newly seen objects, grab/drop reward, the newly discovered map area at time step t, and the episode success reward. Letting $\Delta x_t = d(x_t, x_{t-1})$ be the distance navigated by agent at t, $\Delta SO_t = (SOR_t - SOR_{t-1}) * n_o$ be the change in the number of seen objects at time t, $\Delta RO_t = (ROR_t - ROR_{t-1}) * n_o$ be the change in the number of rearranged objects at time t, $\Delta SC_t = (MC_t - MC_{t-1}) \times$ {total number of navigable cells in the map} be the number of newly seen cells at time t, $\Delta h_t = c * (g_{l_t} - g_{l_{t-1}})$ be the change in the number of held objects at time t, and ES be the episode success from Section 3.3, we can define the reward functions as follows:

$$R_t^1 = -3\Delta x_t + 20(\Delta RO_t + \Delta SO_t) + 10^{-2}\Delta SC_t + 500ES \quad (5a)$$

$$R_t^2 = -3\Delta x_t + 20(\Delta RO_t + \Delta SO_t + \Delta h_t) + 10^{-2}\Delta SC_t \quad (5b)$$

$$R_t^3 = \left\{ \begin{array}{ll} 0, & t \neq T \\ e^{ESPL}, & t = T \end{array} \right\} \quad (5c)$$

These rewards are then used to train the various learning-based agents as shown in Table 6.

Agent	Inputs	Reward
LE_{disc} , LE_w	$\mathcal{M}_v, \mathcal{K}, \mathcal{D}$	Eq (5a)
E2E-P, W2G-P	$\mathcal{M}_b, g_l$	Eq (5b)
OB(MLP)	$\mathcal{M}_b, \mathcal{M}_e, \mathcal{M}_p, g_l$	Eq (5a)
OB(LSTM)	$\mathcal{M}_b, \mathcal{M}_e, \mathcal{M}_p, g_l$	Eq (5c)
BE(MLP)	$\mathcal{M}_b, \mathcal{M}_v, \mathcal{M}_p, \mathcal{K}, \mathcal{D}, g_l$	Eq (5a)
BE(LSTM)	$\mathcal{M}_b, \mathcal{M}_v, \mathcal{M}_p, \mathcal{K}, \mathcal{D}, g_l$	Eq (5c)

Table 6: Inputs, policies and rewards for learning-based agents. Additionally, all agents' input consists of $\mathcal{M}, \mathcal{M}_o, \mathcal{M}_a, \phi$ (where ϕ is the orientation as defined in Section 3.2).

Hyperparameters We train agents with Rllib implementation of DD-PPO algorithm. Each agent is trained for 100M steps with 8x Tesla V100 16GB GPUs and on a 64-core Intel Xeon E5-2686 CPU. Table 7 shows the training and evaluation hyperparameters.

Symbol	Value
θ, r_s, max_t, n_o, n_r, c	360°, 2.0m, 100, 5, 3, 3
batch size	128
discount factor γ	0.9
lambda λ	1.0
vf-loss-coeff	0.8
entropy-coeff	10^{-1} to 10^{-3} in 5M steps
vf-clip-param	100
clip-param ε	0.1

Table 7: Training and evaluation hyperparameters for benchmarking unless otherwise specified.

Comparison of learnt exploration methods with frontier-based exploration methods Table 8 shows a detailed comparison of learnt exploration agents with WFBE agents for purely exploration task on MORP test scenes. LE agents were also trained for the exploration task. The goal of the exploration task is to find all objects in a given scene.

	Path Length $l(m)$					
Metds. \ n_o	1	3	5	10	avg	FO
RND	368	870	1250	2053	1104	81
$WFBE_1$	198	336	393	474	346	84
$WFBE_{0.5}$	146	268	339	440	293	58
$WFBE_0$	183	349	440	601	386	71
$WFBE_r$	146	260	320	411	280	59
LE_w	146	267	338	440	293	92
LE_{disc}	176	341	425	548	366	70

Table 8: Comparison of exploration methods: We evaluate exploration agents using exploration only task on MORP test scenes. We compare agent's path length required to find all objects in the scene and path length until the first object is found (FO).