2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

BASENET: A Learning-based Mobile Manipulator Base Pose Sequence Planning for Pickup Tasks

Lakshadeep Naik[1], Sinan Kalkan[2], Sune L. Sørensen[3], Mikkel B. Kjærgaard[3], and Norbert Krüger[1,4]

***Abstract*— In many applications, a mobile manipulator robot is required to grasp a set of objects distributed in space. This may not be feasible from a single base pose and the robot must plan the sequence of base poses for grasping all objects, minimizing the total navigation and grasping time. This is a Combinatorial Optimization problem that can be solved using exact methods, which provide optimal solutions but are computationally expensive, or approximate methods, which offer computationally efficient but sub-optimal solutions. Recent studies have shown that learning-based methods can solve Combinatorial Optimization problems, providing near-optimal and computationally efficient solutions.**

In this work, we present BASENET - a learning-based approach to plan the sequence of base poses for the robot to grasp all the objects in the scene. We propose a Reinforcement Learning based solution that learns the base poses for grasping individual objects and the sequence in which the objects should be grasped to minimize the total navigation and grasping costs using Layered Learning. As the problem has a varying number of states and actions, we represent states and actions as a graph and use Graph Neural Networks for learning. We show that the proposed method can produce comparable solutions to exact and approximate methods with significantly less computation time. The code and Reinforcement Learning environments will be made available on the project webpage[*].

I. INTRODUCTION

Mobile Manipulators (MMs) are widely used for pick-up tasks across different domains including logistics, manufacturing, service, home automation, elderly care, and hospitality applications [1]. Pickup tasks involve determining suitable robot base poses for object pick-up, followed by navigating to the base pose and picking up the objects using the manipulator [2], [3].

In challenging environments, multiple base poses are generally required for grasping all the objects (see Fig. 1). In such situations, the robot must plan the optimal sequence of base poses for picking up the objects such that the total navigation and grasping time is minimized. This is a Combinatorial Optimization (CO) problem [4], which can be addressed using exact methods such as dynamic programming, which ensure optimal solutions [5], [6]. However, these methods are computationally expensive [7], [8], prohibiting their use in practice on robots as they often require re-planning due to changes in the object configuration, such as those caused by human interference or collisions with other objects during the picking process. Consequently, various approximate solutions [8], [9], [10], [11], [12], [13] have been proposed, balancing optimality and computational efficiency [14].

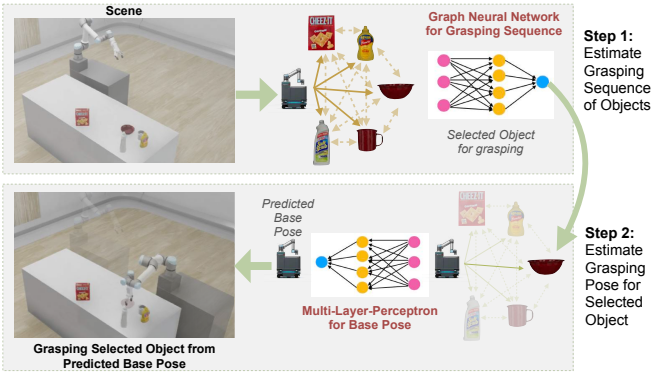


Fig. 1: BASENET **Top row:** The robot represents the scene as a graph and determines the next object to grasp using the *grasp sequence* policy. **Bottom row:** The robot predicts the base pose for grasping the selected object using the *base pose* policy and performs the action.

Several recent works have explored using learning-based methods for solving CO problems, such as routing problems [15], [16]. These methods offer near-optimal and computationally efficient solutions. Drawing inspiration from these works, we propose a learning-based approach for determining the optimal sequence of base poses for grasping all the objects in the scene.

However, a significant distinction exists between the routing problem and determining the sequence of base poses for grasping. In the routing problem, costs depend on the node features (for example, city coordinates in the Travelling Salesman Problem). In the base pose sequence planning problem, node features consist of object poses and the cost depends on the base pose selected for grasping the object. Furthermore, each object can be grasped from several different base poses. Thus, in addition to learning the optimal sequence in which the objects should be grasped, the base pose for grasping each object also needs to be learned. This makes it more challenging to learn compared to routing problems. Furthermore, due to the limited view of the robot's onboard camera and uncertainty in the robot's self-localization, often only uncertain object poses are available

[1]SDU Robotics, Mærsk Mc-Kinney Møller Institute (MMMI), Faculty of Engineering, University of Southern Denmark, Odense M, Denmark {lana,norbert}@mmmi.sdu.dk
[2]Department of Computer Engineering and ROMER Robotics Center, Middle East Technical University (METU), Ankara, Turkey skalkan@metu.edu.tr
[3]SDU Software Engineering, Mærsk Mc-Kinney Møller Institute (MMMI), Faculty of Engineering, University of Southern Denmark, Odense M, Denmark {slso,mbkj}@mmmi.sdu.dk
[4]Danish Institute for Advanced Studies (DIAS), Odense M, Denmark
[*]https://lakshadeep.github.io/basenet/

for base pose sequence planning.

Existing works that learn base poses for grasping have focused on grasping a single object [17], [18]. For planning the sequence of base poses for grasping multiple objects, the current robot pose as well as the poses of all the objects in the scene also must be considered. This poses two main challenges:

1) *Varying number of states and actions.* As the number of objects in the scene can vary, the state and actions cannot be represented using a fixed-dimensional vector.
2) *Sample inefficiency.* Learning in such a high-dimensional state and action space requires a large amount of training data.

We address the first challenge by representing states and actions as a graph and using Graph Neural Network (GNN) to encode a state into a fixed dimensional vector. To address sample inefficiency, we use Layered Learning (LL) [19] in combination with Reinforcement Learning (RL) similar to our previous work [18]. In Layer 1, we learn the *grasp sequence* policy which selects the next object to grasp among the remaining objects (see Fig. 1 first row). In Layer 2, we learn the *base pose* policy which predicts the base pose for grasping the object selected by *grasp sequence* policy (see Fig. 1 second row). We choose to ignore object pose uncertainties to simplify learning. Moreover, by augmenting robot onboard camera views with external cameras in the environment and temporal fusion, accurate pose estimates can be obtained for pre-grasp planning [20] such as the base pose sequence planning.

To summarize, we make the following contributions:

1) We formulate the problem of base pose sequence planning to optimize total navigation and grasping costs as an RL problem.
2) We sequentially learn the base poses for grasping individual objects and the object grasp sequence using LL.
3) We address the variable state and action space challenge in grasp sequence planning by formulating the problem as a graph node regression problem.
4) Through experimental evaluation, we show that BASENET can reduce the total planning and execution time by more than 50% compared to the best-performing baselines with almost the same success rate.

II. RELATED WORK

A. Explicit base pose planning

The selection of a base pose for grasping an object relies on the availability of valid Inverse Kinematics (IK) solutions to achieve the desired grasp pose. Searching for base poses with valid IK solutions in SE(2) can be computationally intensive. Therefore, existing works have suggested the utilization of Inverse Reachability Maps (IRM) [21], [22]. IRM discretizes the base pose space using a grid-based approximation and stores the base poses from which IK solutions are available for the selected grasp pose in the offline phase. During online execution, a specific heuristic is employed to select a particular base pose. The availability of an IK solution does not guarantee that a valid trajectory can be planned to the desired end-effector pose, as trajectory planning depends on several factors such as self-collision, collision with other objects in the scene, joint limits, manipulability ellipsoid of the manipulator, etc. As a result, additional online validations are required to ensure that the trajectory can be planned from the selected base pose.

Recent works have also proposed learning-based methods to predict the base pose for grasping single objects [17], [23]. These methods have shown to be much more computationally efficient and do not suffer from grid-based approximation like IRM.

Difference. In this work, we learn to plan the optimal base pose for grasping an object while also minimizing the combined cost of navigating to the selected base pose from the robot's current base pose and grasping the object.

B. Grasp sequence planning

Being a Combinatorial Optimization (CO) problem, grasp sequence planning can be solved using exact methods that provide optimal solutions at high computational cost [6], or evolutionary [24] or heuristic methods [8], [9], [10], [11], [12] that offer sub-optimal solutions at low computational cost.

Sørensen et al. [6] have employed dynamic programming with memoization to find optimal base pose sequences; however, the quality of obtained solutions directly depends on the action space resolution used for computing the costs. High action space resolution for cost computation produces better solutions but at a high computational cost. Most works that find sub-optimal but quick solutions using non-exact methods utilize IRM and make certain assumptions, such as all objects can be grasped from a single base pose [24], [9], or base pose orientation is fixed [10], [12], or the order in which the objects should be grasped is already known [3], to simplify the complexity of the problem.

Difference. In this work, instead of making any such assumptions, we let the robot itself explore the base pose space for grasping objects in the scene and learn the optimal base pose sequence.

C. Combinatorial optimization and learning

Exact methods, such as dynamic programming, can be applied to any generic CO problems to obtain optimal solutions, albeit at a very high computational cost. Conversely, approximate methods provide quick but sub-optimal solutions by making certain assumptions designed by domain experts to simplify the problem. Moreover, for similar problem instances, such as base pose sequence planning for the same workspace, the optimal solutions would be similar. Hence, learning techniques such as RL can be employed to search for heuristics using data instead of hand-crafted heuristics [15].

Initial works with learning-based solutions for CO problems, such as Pointer networks [25], used supervised data to find the solutions. Later works, such as [14], [26], trained policies in an unsupervised manner using RL, attention

mechanisms [27], etc. In recent years, Graph Neural Networks (GNN) [28] have emerged as efficient state representations for CO problems. GNNs can learn the vector representation that encodes crucial graph structures required to solve CO problems efficiently [16].

Difference. In this work, we employ the Graph Attention Layers [29] to learn a vector representation that encodes relevant grasp scene information for learning the grasp sequence in an unsupervised manner using REINFORCE with greedy rollout baseline, similar to [14].

III. PROBLEM FORMULATION

We address the problem of picking up a set of N rigid objects $\mathcal{O} = \{o_n\}_n$ from a table using a mobile manipulator robot. We assume that the objects can be grasped using an overhead (top-down) grasp and that the robot has a navigation stack [30] to navigate to the planned base pose and a manipulation stack [31] for grasping.

It may not be possible for the robot to pick up all objects from one base pose a^n_{bp} ($\in \text{SE}(2)$) and hence may have to move to a sequence of base poses

$$\mathcal{A}_{\text{seq}} = \{a^1_{\text{bp}}, a^2_{\text{bp}}, ...a^N_{\text{bp}}\}, \tag{1}$$

to pick up N different objects. Our objective is to determine the optimal sequence for grasping objects and the corresponding base pose for each object, ensuring time-efficient completion of the pickup task.

We formulate this as an RL task. During the training stage, N objects $\mathcal{O} = \{o_n\}_n$ are randomly placed on the table. Each RL episode consists of a maximum of N steps. At each step, the robot predicts the next base pose a^n_{bp} ($n \in [1, N]$) and the object to grasp o_m. The objective of the RL agent is to complete the task efficiently, minimizing the total execution time for navigation t_{nav} and grasping t_{grasp}.

Learning such a policy requires information about the current robot base pose and the object poses. Thus, the state space consists of:

$$\mathcal{S} = \{s_r, \{s_{\text{o}_1}, s_{\text{o}_2}, ...s_{\text{o}_M}\}\}, \tag{2}$$

where $s_r \in \text{SE}(2)$ represents the robot base pose in the table frame $\mathbf{W}$, $s_{\text{o}_m} \in \text{SE}(2)$ denotes the m-th object pose in the table frame $\mathbf{W}$, and $M \leq N$ is the number of objects yet to be grasped. The action space consists of actions:

$$\mathcal{A} = \{a^n_{\text{bp}}, \{a_{\text{o}_1}, a_{\text{o}_2}, ...a_{\text{o}_M}\}\}, \tag{3}$$

where $a^n_{\text{bp}} \in \text{SE}(2)$ represents the predicted next robot base pose in the frame of the object selected for grasping, and a_{o_m} signifies the probability of grasping object o_m.

An episode ends when the agent exceeds N steps, collides with the table, or when all the objects are grasped. Further, all states and actions are internally represented with a time variable t, which, however, is omitted in our notations for convenience.

IV. BASENET

We decompose the task of learning to plan a base pose sequence into two sub-tasks:

1) Selecting the next object o_m to be grasped from among the M remaining objects using the *grasp sequence* policy π_{seq} (Layer 1 in Fig. 2).
2) Determining base pose a^n_{bp} for grasping the selected object o_m using the *base pose* policy π_{bp} (Layer 2 in Fig. 2).

Both sub-tasks are learned within the LL framework as shown in Fig. 2. The *base pose* policy π_{bp} is learned before the *grasp sequence* policy π_{seq} as a^n_{bp} is required to perform the action predicted by π_{seq}. In the following sections, we describe learning to estimate the base pose for grasping (Section IV-A) and grasping sequence (Section IV-B).

A. Learning to estimate base pose for grasping

The *base pose* policy π_{bp} is learned using the Soft Actor-Critic (SAC) algorithm [32] as a single-step policy in Layer 2 (see Fig. 2(b)). Each episode consists of a single step wherein the object o_m is randomly placed on the table and the robot is randomly placed in the room within the 3m radius of the table. Given the object pose s_{o_m} and the robot base pose s_r the agent learns to predict the base pose a^n_{bp} for grasping the object o_m:

$$a^n_{\text{bp}} \sim \pi_{\text{bp}}(\ \cdot\ | s_r, s_{\text{o}_m};\ \phi_{\text{base}}), \tag{4}$$

where ϕ_{base} are learnable parameters. The base pose a^n_{bp} is predicted in the object o_m frame; i.e., it is a transformation from the object frame o_m to the robot base frame b; $\mathbf{T}^{o_m}_b$.

The reward is defined as:

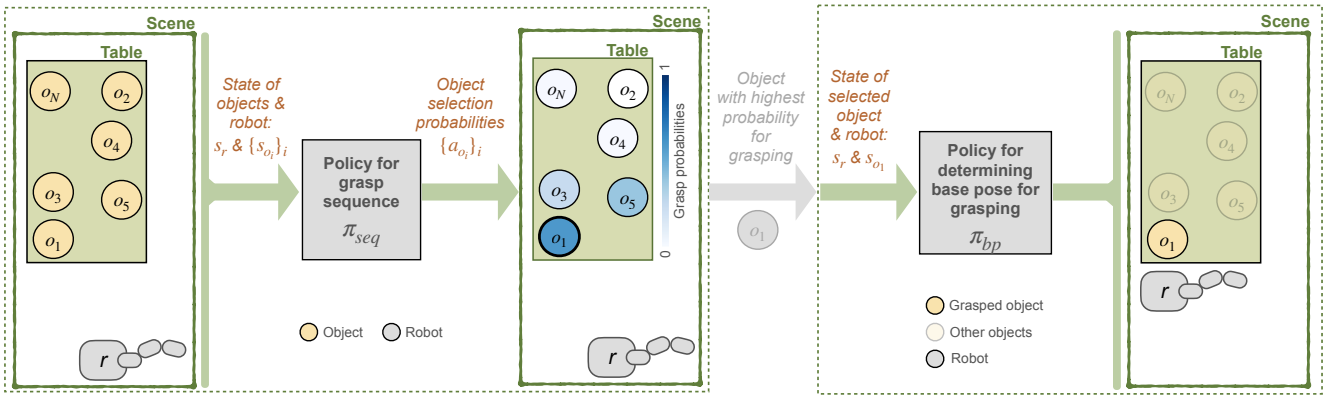
$$\begin{aligned} R_{\text{bp}}(s_{\text{o}_m}, a^n_{\text{bp}}) = \quad & \gamma_1 \cdot \mathbb{1}(\text{collision}(s_{\text{o}_m}, a^n_{\text{bp}})) + \\ & \mathbb{1}(\text{IK}(s_{\text{o}_m}, a^n_{\text{bp}})) \cdot \left[\gamma_2 + \frac{\gamma_3}{1 + t_{\text{nav}}} + \frac{\gamma_4}{1 + t_{\text{grasp}}}\right], \end{aligned} \tag{5}$$

where $\mathbb{1}(\text{collision}(s_{\text{o}_m}, a^n_{\text{bp}}))$ is 1 if there is a collision with the table after moving to a^n_{bp} and 0 otherwise; $\mathbb{1}(\text{IK}(s_{\text{o}_i}, a^n_{\text{bp}}))$ is 1 if IK solutions are available to grasp the object o_m after moving to the base pose a^n_{bp} and 0 otherwise; t_{nav} is the time required to navigate from current robot base pose to the next base pose a^n_{bp}; t_{grasp} is the time required to grasp the object m from the base pose a^n_{bp} and γ_1, γ_2, γ_3, and γ_4 are hyper-parameters.

B. Learning to estimate grasping sequence

In Layer 1, a probability for grasping each object o_m (among the M remaining objects on the table) is learned while using the policy π_{bp} already learned in Layer 2 for taking action a^n_{bp}. Thus, the agent uses a composite policy for exploration, and only the parameters ϕ_{seq} of π_{seq} are learned here (see Fig. 2 (a)):

$$\begin{aligned} & a_{\text{o}_i} \sim \pi_{\text{seq}}(\ \cdot\ | s_{\text{o}_i}, \{s_{\text{o}_j}\}_j, s_r; \phi_{\text{seq}}), \quad i, j \in \{1...M\} \wedge j \neq i, \\ & k = \underset{i}{\text{argmax}} \{a_{\text{o}_i}\}_i, \\ & a^n_{\text{bp}} \sim \pi_{\text{bp}}(\ \cdot\ | s_r, s_{\text{o}_k}; \phi_{\text{base}}), \end{aligned} \tag{6}$$



(a) **Layer 1:** Learning object grasping sequence (b) **Layer 2:** Learning base pose

Fig. 2: Proposed Layered Approach - BASENET: **(b) Layer 2:** Learning base poses for grasping individual objects (*base pose* policy π_{bp}). **(a) Layer 1:** Learning object grasping sequences (*grasp sequence* policy π_{seq}) using already learned π_{bp} for determining the base pose for grasping for the selected object. The example demonstrates that in Layer 1, the object o_1 is selected for grasping as it receives the highest probability. The *base pose* policy is then used in Layer 2 to determine the base pose for grasping object o_1.

where o_i is the object for which the grasp probability is being calculated, $\{s_{o_j}\}_j$ are other objects in the scene that need to be grasped and ϕ_{seq} are learnable parameters. Use of the already learned *base pose* policy reduces the exploration space as only the grasp sequence $\mathcal{A}_{\text{seq}}$ order needs to be explored such that the reward over the entire episode is maximized. The reward for learning grasping sequence, R_{seq}, is calculated as:

$$R_{\text{seq}}(\mathcal{S}, \mathcal{A}) = -\gamma_5 \cdot t_{\text{nav}}, \tag{7}$$

where t_{nav} is the time required to navigate from the current robot base pose to the predicted base pose and γ_5 is a hyper-parameter.

As the number of objects in the scene is not fixed, the state for learning the *grasp sequence* policy π_{seq} cannot be represented using a fixed-dimensional vector similar to the state for the *base pose* policy π_{bp}. Since Graph Neural Networks (GNN) are invariant to node permutations, we use GNN for encoding state into fixed dimensional vector.

Encoder. We use Graph Attention Layers [29] to encode relevant information into a context embedding and formulate the grasp sequence policy π_{seq} as a graph node regression problem. We use a heterogeneous graph with three different types of nodes: the robot r, the object under consideration for grasping o_i, and other objects to be grasped o_j. In the first layer, context embeddings for each node are generated as shown in Fig. 3. All three types of nodes r, o_i, o_j are initially projected to a higher dimensional space using weights w_r, w_g, and w_o respectively. Attention coefficients α are then calculated to determine the level of attention to be given to other objects in the scene o_j while encoding the context embedding for the object under consideration for grasping s_{o_i}. Thus, the context embedding for the object under consideration for grasping o_i is encoded as:

$$h_{o_i} = w_g \cdot s_{o_i} + w_r \cdot s_r + \sum_{j \in \Omega(o_i)} \alpha_{o_i, o_j} \cdot w_o \cdot s_{o_j}, \tag{8}$$

where $\Omega(o_i)$ are the object neighbors of the object o_i (other objects to be grasped in Fig. 3).

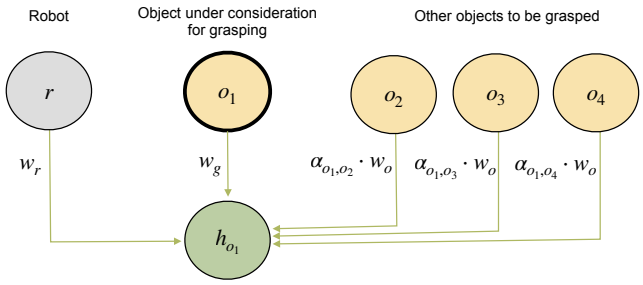


Fig. 3: Attention-Based Graph Encoder.

Decoder. Each episode consists of N steps (number of objects to grasp). At each time step, the grasp probability a_{o_i} is calculated for all objects in the scene that have not yet been grasped. First, the encoder processes relevant information into a fixed-dimensional context embedding for each object. Subsequently, the decoder, which is a Multi-Layer Perceptron (MLP), predicts the grasp probability for each object using the encoded embedding (Graph Node Regression). The object to be grasped is selected by sampling from a categorical distribution parameterized by the grasp probabilities a_{o_i} over objects. The base pose policy for the selected object class is used to determine the base pose and complete the pickup. Fig. 4 presents a decoding example for a 4 objects scene. The grasp sequence policy π_{seq} is learned using REINFORCE [33] with a Greedy Rollout Baseline similar to [14]. The gradient of loss $\mathcal{L}(\phi_{\text{seq}}|S)$ for optimizing learnable parameters ϕ_{seq} in state S is calculated as:

$$\nabla \mathcal{L}(\phi_{\text{seq}}|\mathcal{S}) = -(R_{\text{seq}}(\mathcal{S}, \mathcal{A}) - R_{\text{baseline}}(\mathcal{S})) \cdot \nabla \log a_{o_k}, \tag{9}$$

where a_{o_k} is the grasp probability for the selected object o_k and $R_{\text{baseline}}(S)$ is the baseline reward [14]. The baseline reduces variance and accelerates learning. Attempting to learn

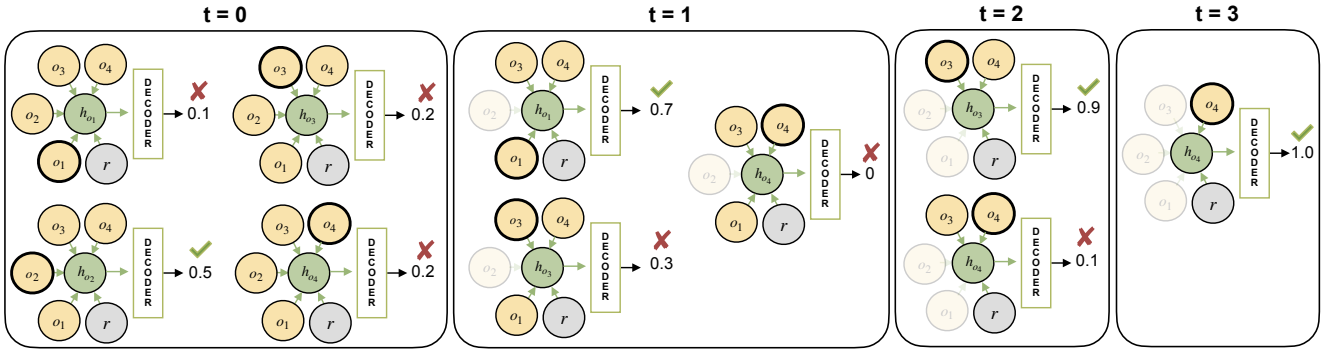


Fig. 4: **Decoder:** The decoder takes the object context embeddings provided by the encoder as input and outputs the grasp probability for each object. Already grasped objects are masked during subsequent time steps. The example demonstrates the construction of a grasp sequence for a scene with four objects.

π_{seq} using actor-critic proved unsuccessful due to difficulties in effectively representing the state and actions for learning the value function (critic).

V. EXPERIMENTAL SETUP

A. Experiment and implementation Details

For evaluating our work, we created an environment in NVIDIA Isaac Sim, using the mobile manipulator platform and a rectangular table (2.0m×0.8m) with up to 10 YCB benchmark [34] objects (belonging to 5 different classes) on it, as shown in Fig. 1. Overhead (top-down) grasp poses for grasping objects of different classes were pre-defined.

For the *base pose* policy π_{bp} learning, we used a Multi-Layer Perceptron (MLP) with three hidden layers in both the Actor and Critic networks in SAC [32]. Each hidden layer comprised 256 neurons with ReLU activation. The learning rate was set to 3e-4. The reward hyper-parameters γ_1, γ_2, γ_3 and γ_4, were empirically set to -2e5, 1e6, 5e5, and 5e5, respectively.

For the *grasp sequence* policy π_{seq} learning, we used a 64-bit vector representation learned through five 64-bit Graph Attention Layers [29] followed by ReLU activation. This vector representation served as input to an MLP with two hidden layers, each comprising 64 neurons with ReLU activation. Both networks were optimized simultaneously using an Adam optimizer [35] with gradients computed via REINFORCE with a greedy rollout baseline [14] and a learning rate of 1e-3. The reward hyper-parameter γ_5 was empirically set to 1e3.

In each episode, the robot's starting pose was randomly sampled within a 2.5-3m radius around the table. For an episode with N objects on the table, the robot can select up to N base poses for grasping all the objects. The episode terminates when the robot has visited N base poses or has grasped all the objects or the predicted base pose cannot be reached as it will result in a collision with the table. The state $\mathcal{S}$ was calculated based on the states provided by the simulator.

To expedite training, instead of using a navigation stack to move to the predicted base poses, the robot was teleported. The navigation cost was computed using the approximation based on the linear and angular travel distance [8]. For grasping, the grasp execution time was determined using approximate trajectory execution time provided by the Lula Trajectory generator available in NVIDIA Isaac Sim, while IRMs were computed using the Lula Kinematics Solver with a discretization of 10cm and 45°. For the algorithmic implementation of our approach, we used the Mushroom RL library [36].

All the experiments were carried out on the workstation equipped with Intel Core i9-13900KF 24-Core processor, 64 GB RAM, and NVIDIA GeForce RTX 4090 24GB GPU.

B. Experiment objectives and baselines

The experiments aim to verify whether the proposed learning-based method can produce solutions comparable to those obtained using exact and approximate methods in a shorter computation time (Section VI-A). The following baselines are considered:

Proximity-Based Greedy selection (PBG): IRMs are used to obtain a set of base poses from which each object can be grasped. The robot selects the object closest to it for grasping next and uses the greedy selection strategy based on the navigation cost to select the base pose for grasping.

Minimum Base Poses (MBP): IRMs are used to obtain a set of base poses from which each object can be grasped. The base poses are then selected to minimize the number of base poses required for grasping all objects. Subsequently, the robot employs a greedy selection strategy based on navigation cost to determine the next base pose. This is similar to [10], [12] without making any assumptions regarding the action space such as fixed base orientation.

Dynamic Programming (DP): IRMs are used to acquire a set of base poses from which each object can be grasped. Then navigation and grasp execution costs are computed for all the base poses. Dynamic Programming with memoization is then used to plan the optimal sequence of base poses similar to [6].

PBG and MBP are approximate methods, whereas DP is

Method	% of objects grasped		Planning time (in s) P		Navigation time (in s) N		Grasping time (in s) G		Total execution time (in s) N + G		Total planning and execution time (in s) P+N+G	
	5-objs	10-objs	5-objs	10-objs	5-objs	10-objs	5-objs	10-objs	5-objs	10-objs	5-objs	10-objs
Approximate Methods												
PBG	10.0	8.9	**0.0±0.0**	**0.0±0.0**	*12.8±4.7*	*15.7±6.7*	*7.8±12.2*	*15.3±16.2*	*20.7*	*31.0*	*20.7*	*31.0*
PBG-GC	94.4	89.5	136.0±2.5	273.2±4.0	21.3±6.2	28.8±7.1	72.9±12.0	137.2±18.6	94.2	166.0	230.3	439.3
MBP	39.6	53.4	**0.0±0.0**	**0.0±0.0**	*12.0±5.9*	*17.5±5.2*	*29.9±14.9*	*80.7±22.7*	*41.9*	*98.2*	*41.9*	*98.2*
MBP-GC	96.0	**97.9**	129.1±1.9	250.5±2.7	**19.3±5.4**	**18.3±2.8**	74.0± 6.8	151.2± 9.3	93.3	169.6	222.5	420.1
Exact Methods												
DP	**98.4**	97.1	131.4±1.7	259.9±3.5	22.3±7.0	19.4±3.4	**65.9± 4.8**	**127.1± 6.8**	**88.3**	**146.4**	219.7	406.3
BASENET (**Ours**)	97.5	94.8	**0.0±0.0**	**0.0±0.0**	27.8±6.0	36.2±4.6	76.0± 8.0	148.6± 9.9	103.8	184.8	**103.8**	**184.8**

TABLE I: Success rate & time comparison. Red highlights unacceptable success rates (less than 80%) & hence *blue italic values* are irrelevant.

an exact method. As IRM does not guarantee that a grasp trajectory can be planned from the base pose, we also present results for PBG and MBP by first validating the set of selected base poses and only considering the base poses from which trajectory can be planned for grasping the object. These baselines are referred to as **PBG-GC** and **MBP-GC**.

In addition, we present ablation studies in Section VI-C to validate our design choices. First, we compare the learning performance of *base pose* policy π_{bp} when the base pose a^n_{bp} is predicted in the frame of the object o_m selected for grasping and in the table frame $\mathbf{W}$. Second, we compare the learning performance of *grasp sequence* policy π_{seq} (i) with and without using a greedy rollout baseline with REINFORCE and (ii) with and without using attention coefficients α for encoding context embedding.

VI. RESULTS

A. Experiment 1: Planning and execution time analysis

In Table I, we present the mean and standard deviation values for *planning time*, *execution time* (*navigation* t_{nav} and *grasping* t_{grasp}), *total time*, and the *percentage of objects grasped* for the five selected baselines and BASENET. We considered two tasks: '5-objs' and '10-objs', each with 5 and 10 objects to grasp, respectively. We evaluated each task over 50 random scenes. During navigation, the maximum base linear and angular velocities were set to 0.5m/s and 0.5rad/s. During manipulation with the UR5e manipulator, the maximum velocities for shoulder and elbow joints were set to 1.0rad/s, and for wrist joints, it was set to 2.0rad/s.

In real-world setups, several challenges related to the robot's perception can contribute to additional execution time and failures. These include planning robot camera views to accurately estimate object poses before grasping, performing 6D pose estimation, etc. Furthermore, inaccurate pose estimates can lead to grasp failures. Since these challenges are not addressed in this work, to avoid their influence, we used a simulated environment for evaluation.

Table I shows that, as expected, DP, being an exact method, produces the most optimal solutions in terms of total *execution time*, with more than 97% of objects successfully grasped. PBG-GC and MBP-GC, which are approximate methods, also produce near-optimal solutions. However, all these baselines have very high *planning* time. The majority of the *planning* time is attributed to the computation of navigation and grasping costs for the action space indicated by the IRM. PBP and MBP have very low *planning* time because they do not involve any cost computation, assuming that trajectories can be planned to grasp the object from all the base poses in the IRM. However, as this assumption doesn't always hold true, especially for 6 DOF manipulators like UR5e, they tend to perform poorly.

BASENET produces solutions comparable to those produced by PBG-GC and MBP-GC in terms of total *execution time* and the *percentage of objects grasped*, but with significantly lower *planning time*. While DP produces better solutions in terms of *execution time*, it also has high *planning time*. Therefore, when it comes to total *planning and execution time* BASENET outperforms all the baselines for both the tasks.

B. Experiment 2: Qualitative results

In Fig. 5, we present qualitative results for a random scene using all the baselines and our method BASENET. It can be observed that the base poses a^n_{bp} planned by BASENET are farther away from the table compared to the DP solutions. This occurs because the *base pose* policy π_{bp} learns to maintain a safe distance from the table, given the high penalty for collision with the table. Consequently, BASENET has longer *grasping time* as the manipulator requires longer trajectories to grasp the objects.

All baselines have a discrete base pose space with the discretization of 10cm and 45°. Consequently, the base poses planned by the baselines are perfectly aligned with each other, requiring only linear robot motions to move between them. In contrast, BASENET predicts base poses in the continuous space and hence they are not perfectly aligned. As a result, the robot requires both linear and angular motion to move between them. This incurs significant navigation costs and explains the higher navigation times for BASENET.

In this work, we have used generic rewards. Both of the above issues can be addressed through task and robot-specific reward engineering.

C. Experiment 3: Ablation analysis

Learning performance of π_{bp}. Fig. 6a compares the performance of *base pose* policy π_{bp} using the object frame

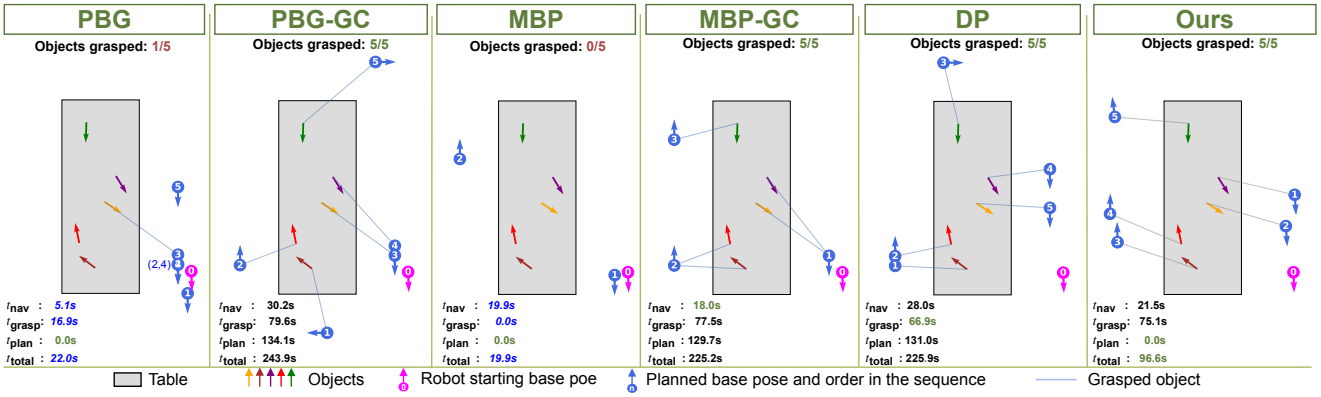
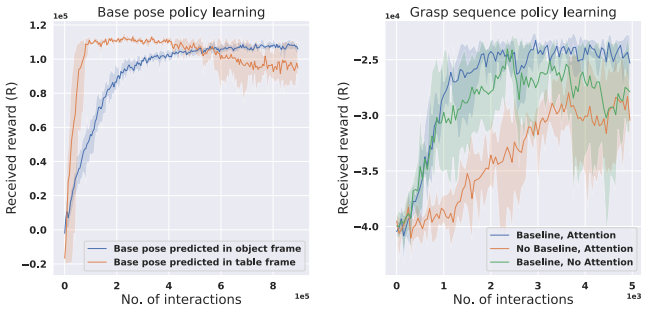


Fig. 5: Sequence of base poses planned by baselines and our method for a random scene. Red highlights unacceptable success rates (objects grasped) & hence *blue italic values* are irrelevant. Predicted base poses are centers of the mobile base. The manipulator is positioned on the right back corner of the mobile base as can be seen in Fig. 1.

o_m and the table frame $\mathbf{W}$ for predicting base poses a^n_{bp} during training. When using the table frame for predicting base poses, π_{bp} quickly learns highly optimal base poses for grasping objects in specific regions of the table. However, it fails to effectively explore the base pose space for objects placed anywhere on the table. Conversely, the use of object frame leads to very stable learning. This observation is further supported by Fig. 7, which shows the learned base poses (colorbar colors) in both cases for 1000 random object poses (orange) on the table (red rectangle). It can be seen that base poses learned in object frame are more generic and have a grasp success rate of over 96%, compared to base poses learned in the table frame, which achieve only around a 91% grasp success rate. These results can be attributed to the fact that in the object frame, the agent only needs to explore the region around the object to predict base poses, simplifying the learning process.



(a) *Base pose* policy π_{bp} (b) *Grasp sequence* policy π_{seq}

Fig. 6: Comparing the learning performance of *base pose* policy π_{bp} and *grasp sequence* policy π_{seq} under different scenarios. Figures show mean over 5 seed runs and min and max variations.

Learning performance of π_{seq}. Fig. 6b compares the performance of the *grasp sequence* policy π_{seq} with and without the Greedy Rollout Baseline and the use of attention coefficients α during training. The use of the baseline accelerates learning and results in better policies with higher rewards. This improvement occurs because the baseline reduces variance during learning [14]. Additionally, we observe that learning attention coefficients (blue) leads to stable learning compared to learning without attention coefficients (green). The attention coefficients learn to determine how much attention should be given to other objects in the scene, thus generating more informative context embeddings.

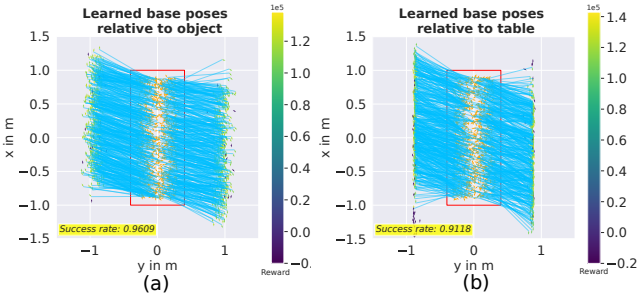


Fig. 7: Learned base poses (colored with colorbar) in **(a)** object frame and **(b)** table frame. The red rectangle denotes the table, objects are orange, and blue lines connect each base pose to the grasped object.

VII. CONCLUSION AND FUTURE WORK

In this work, we have presented BASENET, a learning-based approach for planning mobile manipulator base pose sequences for pick-up tasks while optimizing total navigation and grasping time. We compared our work with three baselines (+2 variations) that use exact and approximate methods for solving the problem. Our experiments show that BASENET produces comparable solutions in significantly less computation time. In this way, BASENET allows the robots to quickly re-plan when the object configuration in the scene changes, either due to human intervention or collision with other objects in the scene.

BASENET has several limitations. For example, it doesn't consider dynamic obstacles in the environment, and new policies must be learned for different workspaces. This can be addressed by including workspace and obstacle shapes

[17] or costmaps [37] in the state space. Another limitation is it doesn't consider uncertainty in the object poses and the robot's self-localization. However, execution failures can be prevented by estimating the uncertainties [20] and assessing whether the estimated errors are acceptable for the successful execution of the planned action [38]. If uncertainties exceed acceptable thresholds, the robot can defer the action execution until uncertainties reduce to acceptable levels. Future works should investigate including pose uncertainties in the state space during learning so that robots can plan the base poses considering uncertainties.

ACKNOWLEDGMENT

This work was supported by the Innovation Fund Denmark's FacilityCobot project and the European Union's Fluently project.

REFERENCES

[1] M. A. Roa, M. R. Dogar, J. Pages, C. Vivas, A. Morales, N. Correll, M. Gorner, J. Rosell, S. Foix, R. Memmesheimer, *et al.*, "Mobile manipulation hackathon: Moving into real world applications," *IEEE Robotics & Automation Magazine*, vol. 28, no. 2, pp. 112–124, 2021.

[2] T. Sandakalum and M. H. Ang Jr, "Motion planning for mobile manipulators a systematic review," *Machines*, vol. 10, no. 2, p. 97, 2022.

[3] F. Reister, M. Grotz, and T. Asfour, "Combining navigation and manipulation costs for time-efficient robot placement in mobile manipulation tasks," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 9913–9920, 2022.

[4] C. H. Papadimitriou and K. Steiglitz, *Combinatorial optimization: algorithms and complexity*. Courier Corporation, 1998.

[5] M. Held and R. M. Karp, "A dynamic programming approach to sequencing problems," *Journal of the Society for Industrial and Applied mathematics*, vol. 10, no. 1, pp. 196–210, 1962.

[6] S. L. Sørensen, L. Naik, P. K. D. T. Nguyen, A. Kramberger, L. Bodenhagen, M. B. Kjærgaard, and N. Krüger, "Planning base poses and object grasp choices for table-clearing tasks using dynamic programming," in *16th International Conference on Agents and Artificial Intelligence*, 2024.

[7] S. D. Han, N. M. Stiffler, A. Krontiris, K. E. Bekris, and J. Yu, "Complexity results and fast methods for optimal tabletop rearrangement with overhand grasps," *The International Journal of Robotics Research*, vol. 37, no. 13-14, pp. 1775–1795, 2018.

[8] F. Wang, J. R. G. Olvera, and G. Cheng, "Optimal order pick-and-place of objects in cluttered scene by a mobile manipulator," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 6402–6409, 2021.

[9] B. Du, J. Zhao, and C. Song, "Optimal base placement and motion planning for mobile manipulators," in *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, vol. 45035. American Society of Mechanical Engineers, 2012, pp. 1227–1234.

[10] K. Harada, T. Tsuji, K. Kikuchi, K. Nagata, H. Onda, and Y. Kawai, "Base position planning for dual-arm mobile manipulators performing a sequence of pick-and-place tasks," in *2015 IEEE International Conference on Humanoid Robots (Humanoids)*. IEEE, 2015, pp. 194–201.

[11] S. Vafadar, A. Olabi, and M. S. Panahi, "Optimal motion planning of mobile manipulators with minimum number of platform movements," in *2018 IEEE International Conference on Industrial Technology (ICIT)*. IEEE, 2018, pp. 262–267.

[12] J. Xu, K. Harada, W. Wan, T. Ueshiba, and Y. Domae, "Planning an efficient and robust base sequence for a mobile manipulator performing multiple pick-and-place tasks," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 11 018–11 024.

[13] H. Zhang, K. Mi, and Z. Zhang, "Base placement optimization for coverage mobile manipulation tasks," *arXiv preprint arXiv:2304.08246*, 2023.

[14] W. Kool, H. van Hoof, and M. Welling, "Attention, learn to solve routing problems!" in *International Conference on Learning Representations*, 2018.

[15] N. Mazyavkina, S. Sviridov, S. Ivanov, and E. Burnaev, "Reinforcement learning for combinatorial optimization: A survey," *Computers & Operations Research*, vol. 134, p. 105400, 2021.

[16] Q. Cappart, D. Chételat, E. B. Khalil, A. Lodi, C. Morris, and P. Veličković, "Combinatorial optimization and reasoning with graph neural networks," *Journal of Machine Learning Research*, vol. 24, no. 130, pp. 1–61, 2023.

[17] S. Jauhri, J. Peters, and G. Chalvatzaki, "Robot learning of mobile manipulation with reachability behavior priors," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 8399–8406, 2022.

[18] L. Naik, S. Kalkan, and N. Krüger, "Pre-grasp approaching on mobile robots: A pre-active layered approach," *IEEE Robotics and Automation Letters*, vol. 9, no. 3, pp. 2606–2613, 2024.

[19] P. Stone and M. Veloso, "Layered learning," in *European conference on machine learning*. Springer, 2000, pp. 369–381.

[20] L. Naik, T. M. Iversen, A. Kramberger, J. Wilm, and N. Krüger, "Multi-view object pose distribution tracking for pre-grasp planning on mobile robots," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 1554–1561.

[21] A. Makhal and A. K. Goins, "Reuleaux: Robot base placement by reachability analysis," in *2018 IEEE International Conference on Robotic Computing (IRC)*. IEEE, 2018, pp. 137–142.

[22] N. Vahrenkamp, T. Asfour, and R. Dillmann, "Robot placement based on reachability inversion," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 1970–1975.

[23] S. Kim and J. Perez, "Learning reachable manifold and inverse mapping for a redundant robot manipulator," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 4731–4737.

[24] D. Berenson, J. Kuffner, and H. Choset, "An optimization approach to planning for mobile manipulation," in *2008 IEEE International Conference on Robotics and Automation*. IEEE, 2008, pp. 1187–1192.

[25] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer networks," *Advances in neural information processing systems*, vol. 28, 2015.

[26] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio, "Neural combinatorial optimization with reinforcement learning," *arXiv preprint arXiv:1611.09940*, 2016.

[27] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[28] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 1, pp. 4–24, 2020.

[29] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *International Conference on Learning Representations*, 2018.

[30] R. L. Guimarães, A. S. de Oliveira, J. A. Fabro, T. Becker, and V. A. Brenner, "Ros navigation: Concepts and tutorial," *Robot Operating System (ROS) The Complete Reference (Volume 1)*, pp. 121–160, 2016.

[31] S. Chitta, I. Sucan, and S. Cousins, "Moveit![ros topics]," *IEEE Robotics & Automation Magazine*, vol. 19, no. 1, pp. 18–19, 2012.

[32] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*. PMLR, 2018, pp. 1861–1870.

[33] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[34] B. Calli, A. Walsman, A. Singh, S. Srinivasa, P. Abbeel, and A. M. Dollar, "Benchmarking in manipulation research: Using the yale-cmu-berkeley object and model set," *IEEE Robotics & Automation Magazine*, vol. 22, no. 3, pp. 36–52, 2015.

[35] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[36] C. D'Eramo, D. Tateo, A. Bonarini, M. Restelli, and J. Peters, "Mushroomrl: Simplifying reinforcement learning research," *JMLR*, vol. 22, no. 131, pp. 1–5, 2021.

[37] D. Honerkamp, T. Welschehold, and A. Valada, "N2m2: Learning navigation for arbitrary mobile manipulation motions in unseen and dynamic environments," *IEEE Transactions on Robotics*, 2023.

[38] L. Naik, T. M. Iversen, A. Kramberger, and N. Krüger, "Robotic task success evaluation under multi-modal non-parametric object pose uncertainty," *arXiv preprint arXiv:2403.10874*, 2024.