Diffusion-Guided Multi-Arm Motion Planning

Viraj Parimi*, **Brian Williams**
Massachusetts Institute of Technology

Abstract: Multi-arm motion planning is fundamental for enabling arms to complete complex long-horizon tasks in shared spaces efficiently but current methods struggle with scalability due to exponential state-space growth and reliance on large training datasets for learned models. Inspired by Multi-Agent Path Finding (MAPF), which decomposes planning into single-agent problems coupled with collision resolution, we propose a novel diffusion-guided multi-arm planner (DG-MAP) that enhances scalability of learning-based models while reducing their reliance on massive multi-arm datasets. Recognizing that collisions are primarily pairwise, we train two conditional diffusion models, one to generate feasible single-arm trajectories, and a second, to model the dual-arm dynamics required for effective pairwise collision resolution. By integrating these specialized generative models within a MAPF-inspired structured decomposition, our planner efficiently scales to larger number of arms. Evaluations against alternative learning-based methods across various team sizes demonstrate our method's effectiveness and practical applicability. Project website: https://diff-mapf-mers.csail.mit.edu

Keywords: Multi-Arm Motion Planning, Diffusion Models, Planning

1 Introduction

The ability of multiple arms to effectively coordinate in shared spaces is crucial for solving complex tasks beyond the capability of individual arms. Humans naturally perform many such sophisticated tasks by leveraging social interaction [1], shared information [2], and distributed responsibilities [3]. Enabling robotic systems, particularly those with multiple manipulators operating in close proximity, to achieve similar efficient, collision-free coordination remains a central challenge. This involves handling high-dimensional joint configuration spaces inherent to multi-arm systems. Traditional sampling-based motion planners (SMPs), such as RRT and PRM [4, 5, 6, 7, 8, 9], face significant challenges when applied to multi-arm systems due to the curse of dimensionality in the joint space. Alternatively, optimization-based planners can efficiently refine trajectories but are susceptible to local minima and often require good initial seeds. Recently, learning-based methods have emerged that can guide such planners [10, 11] with a primary focus on enhancing single-arm planning. Extending them to multi-arm setups would require expensive multi-arm training data and solving large, non-convex optimization problems. To address the multi-agent scalability more directly, other methods [12] have integrated SMPs with Multi-Agent Path Finding (MAPF) decomposition techniques [13], constructing individual roadmaps for each arm and coordinating them using MAPF-like solvers. Nonetheless, scalability remains constrained, as the computational cost of generating initial SMP roadmaps becomes a major bottleneck, particularly as the number of arms increases.

To address these scalability challenges, end-to-end learning-based approaches have gained increasing attention. In particular, Multi-Agent Reinforcement Learning (MARL) methods [14] have emerged, training decentralized policies using expert demonstrations from SMPs such as BiRRT. While they have been shown to be scalable, MARL policies often depend heavily on diverse train-

*Correspondence to vparimi@mit.edu.

9th Conference on Robot Learning (CoRL 2025), Seoul, Korea.

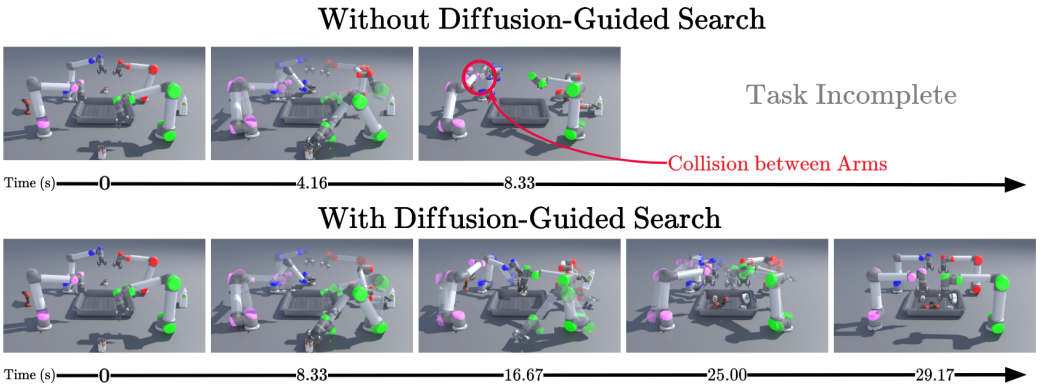

Figure 1: **Multi-Arm Motion Planning**: Timestamped snapshots compare an end-to-end learning method (top) with our DG-MAP approach (bottom) on a multi-arm pick-and-place task. Both were trained using only lower-order interaction data (single/dual-arm trajectories). The end-to-end method fails due to collision with another arm in the shared workspace, while DG-MAP successfully completes the task by leveraging specialized diffusion models combined with MAPF-inspired structured decomposition.

ing data, and those trained primarily on simpler interactions involving only two arms frequently fail to generalize to more complex, larger team scenarios.

An alternative line of work leverages diffusion models [15, 16, 17, 18, 19], which offer a promising combination of generative flexibility and explicit constraint handling. For instance, approaches such as Multi-robot Multi-model planning Diffusion (MMD) [20] integrate single-robot diffusion models within MAPF frameworks to enable collision-free navigation. Nonetheless, encoding complex multi-agent geometric constraints, particularly for articulated manipulators, within diffusion models remains a substantial challenge. Furthermore, current end-to-end multi-agent diffusion policies [19] require centralized training with full-team data, limiting their scalability to unseen team compositions and motivating the need for novel, multi-arm-specific approaches.

Achieving effective multi-arm coordination requires planners that are scalable to large teams, adaptable to dynamic layouts, cooperative in goal achievement, and capable of closed-loop operation for collision avoidance, as highlighted by Ha et al. [14] and illustrated in Figure 1. Motivated by these challenges, we propose a novel closed-loop multi-arm motion planning framework called DG-MAP that combines structured decomposition principles from MAPF with conditional diffusion models to achieve scalable and data-efficient planning. Instead of planning directly in the high-dimensional joint space, we plan in the *conflict-resolution space* wherein we plan motions for each arm independently and use search to iteratively deconflict these plans from any potential collisions among pairs of arms using conflict-resolution based methods. To this end, we introduce two specialized conditional denoising diffusion models. The first model learns to generate feasible single-arm trajectories respecting individual constraints. The second model is specifically designed to generate feasible trajectories necessary to effectively resolve pairwise conflicts, conditioned on the relevant states of the interacting pair. By integrating these generative models within a MAPF-inspired framework, our planner manages combinatorial complexity and reduces the dependency on large-scale higher-order multi-arm training data.

Our main contributions are:

- Conditional denoising diffusion models tailored for generating single-arm trajectories and, distinctively, for resolving pairwise arm collisions.
- A novel framework integrating specialized diffusion models with MAPF-inspired decomposition for multi-arm motion planning.
- Empirical validation showcasing significant improvements in scalability and effectiveness even when trained on simpler interaction scenarios compared to alternative approaches.

2 Preliminaries

2.1 Problem Definition

Consider a total of N manipulator arms, where each arm i has a fixed base pose $\boldsymbol{\xi}_i^{\text{base}}$, d_i degrees-of-freedom (DoF), configuration space $\mathcal{Q}_i \subseteq \mathbb{R}^{d_i}$, and collision-free subspace $\mathcal{Q}_i^{\text{free}}$ with respect to itself and any static obstacles in the environment. Multi-arm motion planning seeks a simultaneous continuous path $\boldsymbol{\tau} : [0, T] \to \mathcal{Q}_{\text{sys}}$, where $\mathcal{Q}_{\text{sys}} = \times_{i=1}^{N} \mathcal{Q}_i$ and $T > 0$ is the final time when all arms have reached their goals, mapping time t to the system configuration $\boldsymbol{q}(t) = (\boldsymbol{q}_1(t), \ldots, \boldsymbol{q}_N(t)) = \boldsymbol{\tau}(t) \in \mathcal{Q}_i$. The path $\boldsymbol{\tau}$ must satisfy:

1. **Boundary Conditions:** $\boldsymbol{\tau}(0) = \boldsymbol{q}_{\text{start}}$ and $\boldsymbol{\tau}(T) = \boldsymbol{q}_{\text{final}}$ such that for all i, the final end-effector pose $\boldsymbol{p}_i^{\text{ee}}(\boldsymbol{q}_i(T)) = \texttt{FK}(\boldsymbol{q}_i(T))$ is within tolerances $(\delta_{\text{pos}}, \delta_{\text{rot}})$ of the goal $\boldsymbol{p}_i^{\text{goal}}$, i.e., $d_{\text{pos}}(\boldsymbol{p}_i^{\text{ee}}(\boldsymbol{q}_i(T)), \boldsymbol{p}_i^{\text{goal}}) \leq \delta_{\text{pos}}$ and $d_{\text{rot}}(\boldsymbol{p}_i^{\text{ee}}(\boldsymbol{q}_i(T)), \boldsymbol{p}_i^{\text{goal}}) \leq \delta_{\text{rot}}$.
2. **Collision Avoidance:** For all $t \in [0, T]$, each arm is collision-free ($\boldsymbol{q}_i(t) \in \mathcal{Q}_i^{\text{free}}\ \forall\ i$) and there are no inter-arm collisions between any distinct pair (i, j), $i \neq j$.

2.2 Multi-Agent Path Finding (MAPF)

Multi-Agent Path Finding (MAPF) [13] is a discrete abstraction of the multi-arm coordination problem that seeks collision-free paths $\boldsymbol{\tau}$ for N agents on a graph representing feasible configurations and transitions. Scalable constraint-based algorithms like Prioritized Planning (PP) [21] and Conflict-Based Search (CBS) [22] excel in MAPF by decomposing the multi-agent planning problem into a series of single-agent problems. They rely on single-agent planners to propose individual trajectories for each agent independent of each other and resolve detected conflicts between two agents by imposing explicit spatio-temporal constraints on subsequent planning queries.

2.3 Diffusion Models

Diffusion models, particularly Denoising Diffusion Probabilistic Models (DDPMs) [16, 15], are probabilistic generative models that learn to reverse a fixed Markovian process that gradually adds Gaussian noise to data over K steps. To generate a sample $\boldsymbol{z}_0$, such as a trajectory segment, the model starts from pure noise $\boldsymbol{z}_K \sim \mathcal{N}(0, \mathbf{I})$ and iteratively denoises it via:

$$\boldsymbol{z}_{k-1} = f(\boldsymbol{z}_k, \boldsymbol{\epsilon}_\theta(\boldsymbol{z}_k, k)) \quad \text{for } k = K, \ldots, 1, \tag{1}$$

where $\boldsymbol{\epsilon}_\theta$ is a neural network predicting the noise at each step, and $f(\cdot)$ denotes the denoising update based on the noise schedule. Training minimizes the mean squared error between the true and predicted noise. For a data point $\boldsymbol{z}_0$ and a timestep $k \sim \text{Uniform}(1, K)$, noise $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$ is added according to, $\boldsymbol{z}_k = \sqrt{\bar{\alpha}_k}\boldsymbol{z}_0 + \sqrt{1 - \bar{\alpha}_k}\boldsymbol{\epsilon}$, where $\bar{\alpha}_k = \prod_{i=1}^{k} \alpha_i$ is the cumulative noise schedule. The loss function then minimizes the difference between the predicted and true noise:

$$\mathcal{L}(\theta) = \mathbb{E}_{\boldsymbol{z}_0, \boldsymbol{\epsilon}, k}\left[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\boldsymbol{z}_k, k)\|^2\right]. \tag{2}$$

Due to their ability to model complex, multi-modal distributions and their stable training dynamics, diffusion models have been successfully applied to robotics tasks such as trajectory planning [18, 17, 23] making them suitable for generating meaningful manipulator trajectories.

3 Approach

We propose DG-MAP, our scalable and closed-loop multi-arm motion planner that integrates specialized generative models like conditional denoising diffusion models within a MAPF-inspired structured decomposition. First, we discuss details about the offline training of these specialized diffusion models following which we will present a simple search-based strategy to integrate these models into a closed-loop planning framework.

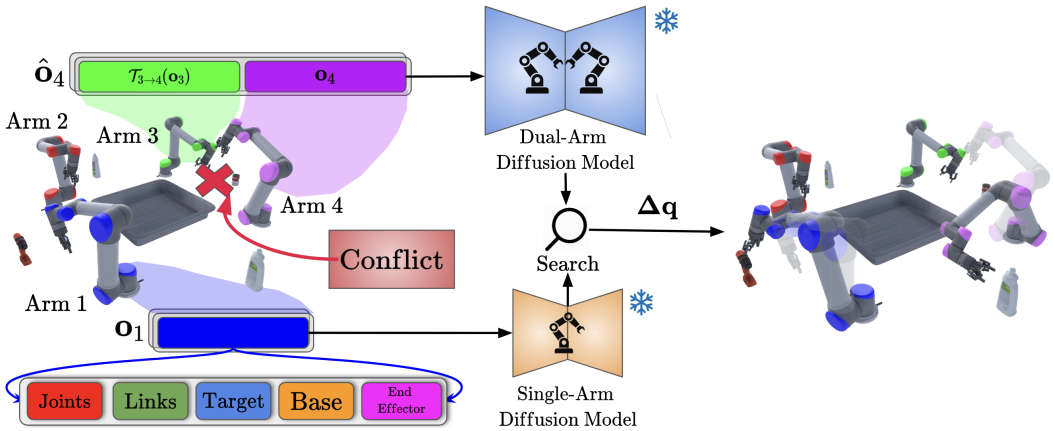

Figure 2: Overview of how DG-MAP operates in a closed-loop, receding horizon controller, generating delta-action sequences for multiple arms. Initially, each arm plans independently using its dedicated single-arm model ($\boldsymbol{\epsilon}_{\theta_1}$). Conflicts trigger Rebranch (Alg. 2) Repair (Alg. 3) strategies. The Repair strategy is highlighted (top) where a conflicting arm (pink) extracts paired observations with the other conflicting arm (green) and utilizes the trained dual-arm model ($\boldsymbol{\epsilon}_{\theta_2}$) to generate feasible delta-actions avoiding the collision.

3.1 Single-Arm Diffusion Model

Addressing the need for a closed-loop control that is crucial for multi-arm coordination with moving obstacles like other arms, we leverage the Diffusion Policy framework [23] to train a generative model for single-arm trajectory generation. A shared set of model parameters, denoted θ_1, is learned using expert demonstrations collected from single-arm BiRRT plans, following the setup in [14]. During planning and execution, each arm i utilizes an independent instantiation of this model to predict a trajectory tailored to its current observation history.

The policy, denoted $\boldsymbol{\epsilon}_{\theta_1}$, generates a sequence of future delta joint actions $\Delta\boldsymbol{q}_i = [\Delta\boldsymbol{q}_i(t), \ldots, \Delta\boldsymbol{q}_i(t+T_p-1)]$ over a prediction horizon T_p. Each predicted trajectory is conditioned on a recent sequence of T_o observations $\boldsymbol{o}_i = [\boldsymbol{o}_i(t-T_o-1), \ldots, \boldsymbol{o}_i(t)]$ unique to arm i. Each observation frame $\boldsymbol{o}_i(t)$ includes the joint configuration $\boldsymbol{q}_i(t)$, end-effector pose $\boldsymbol{p}_i^{\text{ee}}(t)$, target end-effector pose $\boldsymbol{p}_i^{\text{goal}}$, link positions $\boldsymbol{L}_i(t)$, and constant base pose $\boldsymbol{\xi}_i^{\text{base}}$. This stacked observation sequence provides temporal and goal-directed context to the policy.

$$\mathcal{L}(\theta_1) = \mathbb{E}_{k,\Delta\boldsymbol{q}_i^0,\boldsymbol{o}_i,\boldsymbol{\epsilon}}\left[\left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\theta_1}(\boldsymbol{o}_i, \Delta\boldsymbol{q}_i^k, k)\right\|^2\right] \tag{3}$$

The model $\boldsymbol{\epsilon}_{\theta_1}$ is trained via the standard diffusion objective $\mathcal{L}(\theta_1)$ (Eq. (3)) to predict the noise $\boldsymbol{\epsilon}$ added to ground-truth actions $\Delta\boldsymbol{q}_i^0$, conditioned on observations $\boldsymbol{o}_i$ and timestep k. After training, each arm independently samples its own action trajectory by running the denoising process using its current observation history $\boldsymbol{o}_i$. This trained model serves to generate initial, potentially conflicting, trajectory proposals for individual arms.

3.2 Dual-Arm Diffusion Model

To resolve inter-arm conflicts during multi-arm planning, we introduce a dual-arm diffusion model $\boldsymbol{\epsilon}_{\theta_2}$ that learns using expert demonstrations collected from dual-arm BiRRT plans. While the action space remains unchanged where we are predicting delta joint action sequences $\Delta\boldsymbol{q}_i$ for the ego-arm, the observation structure is reorganized to explicitly capture interactions between the ego-arm and the conflicting arm from this interacting pair inspired from [2, 14].

The dual-arm observation, denoted as $\hat{\boldsymbol{o}}_i$, is constructed by *pairing* the transformed observations of the conflicting arm with the ego-arm's own observations at each timestep across a his-

tory window of T_o steps. Specifically, for each timestep $t' \in [t - T_o - 1, \ldots, t]$, we define $\hat{\boldsymbol{o}}_i(t') = [\mathcal{T}_{j \to i}(\boldsymbol{o}_j(t')) \oplus \boldsymbol{o}_i(t')]$, where $\mathcal{T}j \to i(\cdot)$ denotes the transformation that maps arm j's observations into arm i's reference frame, and $\oplus$ represents concatenation. The full observation input to the diffusion model is then $\hat{\boldsymbol{o}}_i = [\hat{\boldsymbol{o}}_i(t - T_o - 1), \ldots, \hat{\boldsymbol{o}}_i(t)]$. The dual-arm diffusion model $\boldsymbol{\epsilon}_{\theta_2}$ is trained to predict the added noise at each diffusion timestep k, minimizing the denoising loss:

$$\mathcal{L}(\theta_2) = \mathbb{E}_{k, \Delta \boldsymbol{q}_i^0, \hat{\boldsymbol{o}}_i, \boldsymbol{\epsilon}} \left[\left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\theta_2}(\hat{\boldsymbol{o}}_i, \Delta \boldsymbol{q}_i^k, k) \right\|^2 \right], \tag{4}$$

where $\Delta \boldsymbol{q}_i^0$ denotes the ground truth sequence of delta joint actions. The model thus learns to generate collision-avoiding ego-arm trajectories while conditioning on the relevant information of the conflicting arm.

3.3 Diffusion-Guided Multi-Arm Planning (DG-MAP)

We integrate the trained single-arm ($\boldsymbol{\epsilon}_{\theta_1}$) and dual-arm ($\boldsymbol{\epsilon}_{\theta_2}$) diffusion models within a search framework inspired by MAPF principles. The planner searches over candidate delta-action sequences for each arm, leveraging the diffusion models for proposal generation and conflict resolution. The overall procedure is detailed in Algorithm 1, utilizing the subroutines defined in Algorithms 2 and 3.

Planner State and Initialization: The search state is represented by nodes $\mathcal{N}$ in a search tree, each corresponding to a tuple of selected plan indices $\boldsymbol{b} = (b_1, \ldots, b_N)$. An index $b_i \in \{1, \ldots, B\}$ points to a candidate delta-action sequence $\Delta \boldsymbol{q}_i^{b_i}$ for arm i, covering a prediction horizon T_p. The search maintains a frontier set $\mathcal{F}$, implemented as a min-priority queue storing nodes $\mathcal{N}$ ordered by a cost function $g(\mathcal{N})$ guiding the search towards promising nodes. This cost estimates solution quality by combining path smoothness, goal proximity and penalty for collisions. Initially, each arm i uses its single-arm model $\boldsymbol{\epsilon}_{\theta_1}$ conditioned on its observation history $\boldsymbol{o}_i$ of up to T_o steps in the past to sample B diverse candidate delta-action sequences $\mathcal{P}_i = \{\Delta \boldsymbol{q}_i^1, \ldots, \Delta \boldsymbol{q}_i^B\}$. The search begins with a root node $\mathcal{N}_0$ placed in $\mathcal{F}$ along with an empty set of conflicting plan indices ($\mathcal{K}$) that will be updated upon expansions. A collision cache $\mathcal{C}$ stores pairwise collision check results.

Algorithm 1 Diffusion-Guided Multi-Arm Planner (DG-MAP)

1: **Input:** Models $\boldsymbol{\epsilon}_{\theta_1}, \boldsymbol{\epsilon}_{\theta_2}$
2: **Output:** Collision-free plan $\{\Delta \boldsymbol{q}_i\}_{i=1}^N$ or best effort
3: **Initialize:** Frontier set $\mathcal{F} \leftarrow \emptyset$, collision cache $\mathcal{C} \leftarrow \emptyset$
4: **for** $i = 1 \ldots N$ **do**
5: $\quad \boldsymbol{o}_i \leftarrow$ `GetObs(i)`
6: $\quad \mathcal{P}_i \leftarrow \boldsymbol{\epsilon}_{\theta_1}(\boldsymbol{o}_i, \cdot, \cdot)$
7: **end for**
// Generate initial plans independent of each other
8: $\mathcal{N}_0 \leftarrow$ `Node`$(\boldsymbol{b} = \vec{0}, \mathcal{K} = \emptyset)$
// Node stores indices and conflicting plan indices
9: $\mathcal{F}$.`Insert`$(\mathcal{N}_0, g(\mathcal{N}_0))$
10: **while** $\mathcal{F}$ not empty and time not exceeded **do**
11: $\quad \mathcal{N} \leftarrow \mathcal{F}$.`ExtractMin`$()$
12: $\quad \boldsymbol{\tau} \leftarrow \{\boldsymbol{\Delta q}_i^{\mathcal{N}.b_i}\}_{i=1}^N$
13: $\quad \boldsymbol{c} \leftarrow$ `FindFirstCollision`$(\boldsymbol{\tau}, \mathcal{C})$
14: $\quad$ **if** $\boldsymbol{c}$ is **null then**
15: $\quad\quad$ **return** $\boldsymbol{\tau}, t^*$ // Solution found
16: $\quad$ **end if**
17: $\quad (i, j, \hat{t}) \leftarrow \boldsymbol{c}$ // Conflicting pair and time
18: $\quad t^* = \min(t^*, \hat{t})$ // Update earliest collision time
19: $\quad \kappa_i = \mathcal{N}.\mathcal{K} \cup \mathcal{N}.b_i$ // Attempt to fix for arm i
20: $\quad$ `Rebranch`(i, κ_i)
21: $\quad$ `Repair`(i, j, κ_i)
22: $\quad \kappa_j = \mathcal{N}.\mathcal{K} \cup \mathcal{N}.b_j$ // Attempt to fix for arm j
23: $\quad$ `Rebranch`(j, κ_j)
24: $\quad$ `Repair`(j, i, κ_j)
25: **end while**
26: **return** Best plan found in $\mathcal{F}$ based on cost
// Timeout or failure

Search Process and Conflict Resolution: The main loop (lines 10-25) iteratively extracts the lowest-cost node $\mathcal{N}$ from $\mathcal{F}$. It checks the corresponding plan combination for collisions using `FindFirstCollision` (line 13). If no conflict is found, a valid solution is returned (line 14).

As the node $\mathcal{N}$ is expanded we detect a conflict $c = (i, j, \hat{t})$, where $\hat{t}$ is the earliest time step within the prediction horizon T_p where the plans $\Delta \boldsymbol{q}_i^{b_i}$ and $\Delta \boldsymbol{q}_j^{b_j}$ collide. t^* is the earliest conflict time

Algorithm 2 Generate Successors by Rebranch

Input: ego arm, conflicts κ
for $m = 1 \dots |\mathcal{P}_{\texttt{ego}}|$ **do**
  if $m \notin \kappa$ **then**
    $b^{\text{new}}_{\texttt{ego}} \leftarrow m$
    $\boldsymbol{b}' \leftarrow (b_1, \dots, b^{\text{new}}_{\texttt{ego}}, \dots, b_N)$
    $\mathcal{N}' \leftarrow \texttt{Node}(\boldsymbol{b} = \boldsymbol{b}', \mathcal{K} = \kappa)$
    $\mathcal{F}.\texttt{Insert}(\mathcal{N}', g(\mathcal{N}'))$
  end if
end for

Algorithm 3 Generate Successors by Repair

Input: ego arm, other arm, conflicts κ, Model $\boldsymbol{\epsilon}_{\theta_2}$
$\hat{\boldsymbol{o}}_{\texttt{ego}} \leftarrow \texttt{GetPairedObs}(\texttt{ego}, \texttt{other})$
Sample $\{\Delta \boldsymbol{q}^{\text{new},m}_{\texttt{ego}}\}_{m=1}^{B}$ using $\boldsymbol{\epsilon}_{\theta_2}(\hat{\boldsymbol{o}}_{\texttt{ego}}, \cdot, \cdot)$
for $m = 1 \dots B$ **do**
  $b^{\text{new}}_{\texttt{ego}} \leftarrow \mathcal{P}_{\texttt{ego}}.\texttt{Update}(\Delta \boldsymbol{q}^{\text{new},m}_{\texttt{ego}})$
  $\boldsymbol{b}' \leftarrow (b_1, \dots, b^{\text{new}}_{\texttt{ego}}, \dots, b_N)$
  $\mathcal{N}' \leftarrow (\boldsymbol{b} = \boldsymbol{b}', \mathcal{K} = \kappa)$
  $\mathcal{F}.\texttt{Insert}(\mathcal{N}', g(\mathcal{N}'))$
end for

and is updated based on $\hat{t}$. It signifies that the current combination of plans is collision-free up to time $t^* - 1$, which could potentially allow for safe partial execution in a receding horizon context, although our primary goal here is to find a fully collision-free plan for the horizon T_p. The set of conflicts κ_i indicate the plan indices corresponding to arm i that have been found to be in conflict in the ancestors of the node. This book-keeping helps in avoiding redundant plans that did not succeed in earlier expansions. The node expansion generates successor nodes by exploring alternatives for the conflicting arms i and j using two distinct strategies (lines 20-24). First, `Rebranch` (Algorithm 2) creates successors by trying alternative, pre-existing plans from $\mathcal{P}$ for both i and j. Second, `Repair` (Algorithm 3) uses the dual-arm model $\boldsymbol{\epsilon}_{\theta_2}$ to generate B new candidate sequences for both i and j, specifically aiming to avoid the detected conflict c, and creates successors for each successful repair. All generated successors are added to $\mathcal{F}$, allowing the search to explore other alternatives based on their estimated costs.

Termination: The search continues until a collision-free node is extracted or termination conditions (timeout or empty frontier) are met, returning the best plan found.

DG-MAP (Algorithm 1) operates within a closed-loop, receding horizon controller (Figure 2). At each step, it plans from the current state, returning action sequences $\{\Delta \boldsymbol{q}_i^{b_i}\}_{i=1}^{N}$ for horizon T_p and the predicted collision-free duration $t^* \leq T_p$. Actions are executed for t^* steps, the state is updated, and the process repeats. This interleaved planning and execution ensures adaptation, persistent collision avoidance, and progress towards goals $\{\boldsymbol{p}_i^{\text{goal}}\}_{i=1}^{N}$. Executing up to t^* provides a balance between reactivity and planning efficiency, avoiding the excessive planning time of single-step execution.

4 Experiments

Our experiments aim to answer the following key questions regarding DG-MAP:

Q1: How effectively does DG-MAP scale to planning for larger numbers of arms ($N = 3$ to $N = 8$) when trained only on single and dual-arm interaction data?
Q2: How does DG-MAP's performance compare to the learning-based methods, trained on limited interaction data?
Q3: Can DG-MAP remain competitive with the learning-based methods, which leverage significantly more complex higher-order interaction data during training?
Q4: Can DG-MAP be applied to practical applications apart from simple single goal-reaching tasks?

Setup: We focus on static single goal-reaching tasks with $N = 3$ to $N = 8$ manipulator arms, forming a foundation for applications like pick-and-place. Each arm has $d_i = 6$ DoF. Tasks are executed in the PyBullet simulator [24] and are considered successful if all N arms reach their target end-effector poses $\boldsymbol{p}_i^{\text{goal}}$ within a positional tolerance $\delta_{\text{pos}} = 0.03$ units and orientation tolerance $\delta_{\text{rot}} = 0.1$ radians. Each attempt is limited to 400 simulation steps. We generate a test dataset of

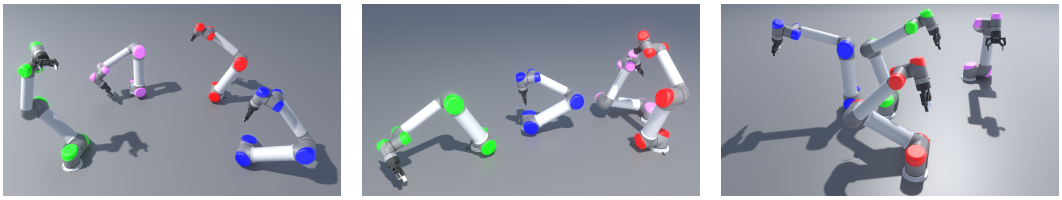

Figure 3: Visualization of test tasks for four arms across different difficulty levels: easy (left), medium (middle), and hard (right). Task difficulty is based on the maximum intersection between the arms' hemispherical workspaces [14], with easy tasks involving minimal arm interaction and hard tasks requiring significant overlap and coordination.

18,000 unique scenarios, with novel base poses $\boldsymbol{\xi}_i^{\text{base}}$ and target poses $\boldsymbol{p}_i^{\text{goal}}$ distinct from training data, evenly distributed across arm counts. Following [14], tasks are categorized by difficulty based on the maximum workspace intersection volume between pairs of arms, as illustrated in the Figure 3. Implementation specific details can be found in the Appendix.

For comparisons, we utilize two variants of the multi-arm motion planner from Ha et al. [14], which represents a valuable learning-based baseline with demonstrable scalability assurances.

1. **Baseline-LD (Limited Data):** This variant is trained using only single-arm and dual-arm interactions for up to 756.8M timesteps. This comparison directly assesses the effectiveness of our approach given the same limited interaction data.
2. **Baseline-ED (Extended Data):** This variant is trained on extended interaction including three and four arms for up to 668.2 M timesteps allowing us to quantify the performance differential between models that can leverage complex multi-arm coordination compared to simplistic pairwise interactions combined with planning.

Comparison with Baseline-LD: Table 1 shows that DG-MAP significantly outperforms Baseline-LD across all task difficulties and team sizes. While Baseline-LD achieves moderate success with 3 arms (41.2% on average), its performance quickly deteriorates with more arms, dropping below 10% and sometimes failing entirely on medium and hard tasks beyond 4 arms. In contrast, DG-MAP maintains consistently high average success rates above 90% across all settings. This stark contrast highlights the scalability of DG-MAP's planning-guided coordination in complex multi-arm environments, where learning-only methods like Baseline-LD struggle due to limited training on higher-order interactions addressing both **Q1** and **Q2**.

Arms	Easy		Medium		Hard		Average	
	Baseline LD	DG-MAP Ours (↑)	Baseline LD	DG-MAP Ours (↑)	Baseline LD	DG-MAP Ours (↑)	Baseline LD	DG-MAP Ours (↑)
3	0.349	**0.984**	0.426	**0.975**	0.460	**0.965**	0.412	**0.975**
4	0.032	**0.981**	0.024	**0.969**	0.060	**0.953**	0.039	**0.968**
5	0.224	**0.972**	-	**0.958**	-	**0.905**	0.075	**0.945**
6	0.145	**0.976**	0.011	**0.921**	0.008	**0.888**	0.055	**0.928**
7	0.113	**0.955**	-	**0.918**	-	**0.907**	0.038	**0.926**
8	0.067	**0.951**	-	**0.933**	-	**0.888**	0.022	**0.924**

Table 1: Success rates (%, higher is better) for Baseline-LD and DG-MAP across easy, medium, and hard tasks, evaluated with 3 to 8 arms. - indicates no successes.

Comparison with Baseline-ED: Table 2 presents success rates for Baseline-ED along with the relative improvement achieved by DG-MAP, measured as a percentage gap. While Baseline-ED maintains high performance across all settings (over 85% success), DG-MAP consistently outperforms it with substantial gains for larger teams, with relative improvements up to 3.8% on medium-difficulty tasks with 6 arms and 3.3% on hard tasks with 8 arms. This demonstrates DG-MAP's planning-guided coordination provides robust scalability and efficiency even in dense, high-interaction scenarios, making it competitive even with end-to-end learned policies that have been trained on higher-order interaction data answering **Q3**.

Arms	Easy		Medium		Hard		Average	
	Baseline ED	DG-MAP Gap (↑)	Baseline ED	DG-MAP Gap (↑)	Baseline ED	DG-MAP Gap (↑)	Baseline ED	DG-MAP Gap (↑)
3	0.980	**0.984**	0.973	**0.975**	0.943	**0.965**	0.965	**0.975**
4	0.973	**0.981**	0.961	**0.969**	0.950	**0.953**	0.961	**0.968**
5	0.963	**0.972**	0.947	**0.958**	0.891	**0.905**	0.934	**0.945**
6	0.950	**0.976**	0.887	**0.921**	0.882	**0.888**	0.906	**0.928**
7	0.950	**0.955**	0.913	**0.918**	0.891	**0.907**	0.917	**0.926**
8	**0.951**	**0.951**	0.911	**0.933**	0.859	**0.888**	0.907	**0.924**

Table 2: Success rates (%, higher is better) for Baseline-ED and DG-MAP across easy, medium, and hard tasks, evaluated with 3 to 8 arms.

Multi-Arm Pick-And-Place Task

To address **Q4**, we evaluate DG-MAP on a four-arm, 6-DoF pick-and-place simulation setup adapted from [14]. In this setup, four UR5 robots equipped with Robotiq 2F-85 grippers are positioned at the corners of a central bin. The task requires the arms to collaboratively pick objects from the ground and deposit them into the bin as shown in Figure. 4. Each trial involves randomly sampled grasp objects from a 7-object subset of the YCB dataset [25]. The motion consists of two collision-free phases starting from the initial pose to a pregrasp pose and from grasp to dump pose. Grasp execution is managed by a separate low-level controller using precomputed grasps generated from GraspIt! [26] with added noise [27]. Following [14], success is defined as all arms completing the pick-and-place cycle without inter-arm or ground collisions. Bin collisions and knocked-over objects are discarded, consistent with the training dataset that only incorporates arm states without the bin or objects.

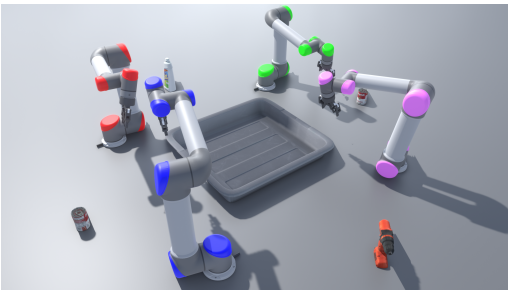

Figure 4: Visualization of the multi-arm pick-and-place task where arms are tasked to pick objects from the ground onto the central bin.

Method	Success (↑)	Steps (↓)
Baseline-LD	0.375	**4479**
Baseline-ED	0.714	6018
DG-MAP	**0.890**	5390

Table 3: Success rate (%, higher is better) along with average number of steps (lower is better) taken by the methods to complete the full cycle of multi-arm pick-and-place task.

Using this setup, DG-MAP is able to improve upon both Baseline-LD and Baseline-ED. Across 100 trials, DG-MAP achieves a success rate of 89% highlighting its effectiveness in enabling scalable, collision-free multi-arm coordination even in dense, shared workspaces.

5 Conclusion

This paper introduced DG-MAP, a diffusion-guided multi-arm motion planner addressing the scalability challenge in multi-arm motion planning. By integrating MAPF principles with specialized single-arm (ϵ_{θ_1}) and dual-arm (ϵ_{θ_2}) conditional diffusion models trained only on corresponding interaction data, DG-MAP efficiently coordinates multiple arms without requiring complex higher-order interaction data. Our experiments demonstrated that DG-MAP outperforms alternative learning-based baseline trained on identical limited data, achieving high success rates ($>88\%$) for up to eight arms. Furthermore, it remained competitive with a baseline trained on richer multi-arm data, highlighting the data efficiency and effectiveness of the structured pairwise resolution approach. The successful application to a complex pick-and-place task further validated its practical utility beyond simple goal-reaching tasks.

Acknowledgments

We gratefully acknowledge support from Defence Science and Technology Agency, Singapore. The authors also thank Annabel Gomez for her extensive feedback on the paper draft.

References

[1] M. Tomasello. *A Natural History of Human Thinking*. Harvard University Press, 2014. ISBN 9780674724778. URL http://www.jstor.org/stable/j.ctt6wpq11.

[2] N. Sebanz, H. Bekkering, and G. Knoblich. Joint action: bodies and minds moving together. *Trends in Cognitive Sciences*, 10(2):70–76, 2006. ISSN 1364-6613. doi:https://doi.org/10.1016/j.tics.2005.12.009. URL https://www.sciencedirect.com/science/article/pii/S1364661305003566.

[3] E. Salas, N. Cooke, and M. Rosen. On teams, teamwork, and team performance: Discoveries and developments. *Human factors*, 50:540–7, 07 2008. doi:10.1518/001872008X288457.

[4] S. M. LaValle. *Planning algorithms*. Cambridge university press, 2006.

[5] S. M. LaValle. Rapidly-exploring random trees : a new tool for path planning. *The annual research report*, 1998. URL https://api.semanticscholar.org/CorpusID:14744621.

[6] S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning, 2011. URL https://arxiv.org/abs/1105.1186.

[7] J. Kuffner and S. LaValle. Rrt-connect: An efficient approach to single-query path planning. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, volume 2, pages 995–1001 vol.2, 2000. doi:10.1109/ROBOT.2000.844730.

[8] A. H. Qureshi and Y. Ayaz. Intelligent bidirectional rapidly-exploring random trees for optimal motion planning in complex cluttered environments. *Robotics and Autonomous Systems*, 68: 1–11, June 2015. ISSN 0921-8890. doi:10.1016/j.robot.2015.02.007. URL http://dx.doi.org/10.1016/j.robot.2015.02.007.

[9] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996. doi:10.1109/70.508439.

[10] M. Seo, Y. Cho, Y. Sung, P. Stone, Y. Zhu, and B. Kim. Presto: Fast motion planning using diffusion models based on key-configuration environment representation, 2025. URL https://arxiv.org/abs/2409.16012.

[11] H. Huang, B. Sundaralingam, A. Mousavian, A. Murali, K. Goldberg, and D. Fox. Diffusionseeder: Seeding motion optimization with diffusion for rapid motion planning, 2024. URL https://arxiv.org/abs/2410.16727.

[12] J. Chen, J. Li, Y. Huang, C. Garrett, D. Sun, C. Fan, A. Hofmann, C. Mueller, S. Koenig, and B. C. Williams. Cooperative task and motion planning for multi-arm assembly systems, 2022. URL https://arxiv.org/abs/2203.02475.

[13] R. Stern, N. Sturtevant, A. Felner, S. Koenig, H. Ma, T. Walker, J. Li, D. Atzmon, L. Cohen, T. Kumar, et al. Multi-agent pathfinding: Definitions, variants, and benchmarks. In *Proceedings of the International Symposium on Combinatorial Search*, volume 10, pages 151–158, 2019.

[14] H. Ha, J. Xu, and S. Song. Learning a decentralized multi-arm motion planner. In *Conference on Robotic Learning (CoRL)*, 2020.

[15] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole. Score-based generative modeling through stochastic differential equations, 2021. URL https://arxiv.org/abs/2011.13456.

[16] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models, 2020. URL https://arxiv.org/abs/2006.11239.

[17] J. Carvalho, A. T. Le, M. Baierl, D. Koert, and J. Peters. Motion planning diffusion: Learning and planning of robot motions with diffusion models, 2024. URL https://arxiv.org/abs/2308.01557.

[18] M. Janner, Y. Du, J. B. Tenenbaum, and S. Levine. Planning with diffusion for flexible behavior synthesis, 2022. URL https://arxiv.org/abs/2205.09991.

[19] Z. Zhu, M. Liu, L. Mao, B. Kang, M. Xu, Y. Yu, S. Ermon, and W. Zhang. Madiff: Offline multi-agent learning with diffusion models, 2025. URL https://arxiv.org/abs/2305.17330.

[20] Y. Shaoul, I. Mishani, S. Vats, J. Li, and M. Likhachev. Multi-robot motion planning with diffusion models, 2024. URL https://arxiv.org/abs/2410.03072.

[21] M. Erdmann and T. Lozano-Perez. On multiple moving objects. In *Proceedings. 1986 IEEE International Conference on Robotics and Automation*, volume 3, pages 1419–1424, 1986. doi:10.1109/ROBOT.1986.1087401.

[22] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant. Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelligence*, 219:40–66, 2015.

[23] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023.

[24] E. Coumans and Y. Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning, 2016.

[25] 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*, 22(3):36–52, 2015. doi:10.1109/MRA.2015.2448951.

[26] A. Miller and P. Allen. Graspit! a versatile simulator for robotic grasping. *IEEE Robotics & Automation Magazine*, 11(4):110–122, 2004. doi:10.1109/MRA.2004.1371616.

[27] J. Weisz and P. K. Allen. Pose error robust grasping from contact wrench space metrics. *2012 IEEE International Conference on Robotics and Automation*, pages 557–562, 2012. URL https://api.semanticscholar.org/CorpusID:2488065.

[28] J. Gao, B. Sarkar, F. Xia, T. Xiao, J. Wu, B. Ichter, A. Majumdar, and D. Sadigh. Physically grounded vision-language models for robotic manipulation. In *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024.

[29] A. Nichol and P. Dhariwal. Improved denoising diffusion probabilistic models, 2021. URL https://arxiv.org/abs/2102.09672.

[30] Z. Wang, J. J. Hunt, and M. Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning, 2023. URL https://arxiv.org/abs/2208.06193.

A Appendix

A.1 Limitations

While DG-MAP demonstrates significant promise and effectiveness across various multi-arm tasks, several aspects offer avenues for future research and enhancement. Currently, the approach leverages forward simulation of the predicted plans to check for collisions implying that the performance can be limited and highly complex environments might challenge real-time execution. Furthermore, the training relies on low-level state information like joint values and link positions. This results in models specialized to the specific arm morphologies used during training, ensuring high precision for those systems when transferring them to real manipulators but limiting direct transferability to different robot arms or heterogeneous setups. Future work could explore incorporating morphology-agnostic representations, potentially through visual perception or with VLMs [28] to enhance generalization across platforms.

The core design of DG-MAP focuses on decomposing complexity via single and dual-arm interactions, which significantly reduces training data requirements and directly addresses the most frequent conflict types. However, complex coordination strategies involving three or more arms simultaneously might not be fully captured by this pairwise approach. Investigating methods to efficiently incorporate or adaptively switch to higher-order interaction models, perhaps when pairwise resolution fails or when richer data is available, could further improve performance in highly complex scenarios. Additionally, a key limitation is the planner's dependence on the single-arm model (ϵ_{θ_1}). If all initial trajectory proposals contain intrinsic collisions like with itself or the plane, the system cannot recover, as the dual-arm model (ϵ_{θ_2}) exclusively addresses inter-arm conflicts. Finally, the closed-loop, interleaved planning and execution cycle provides reactivity but introduces computational latency during planning, as illustrated in Figure 5. While acceptable for some robotic tasks with less outside interventions, applications demanding extremely high-speed reactions to unpredictable events, such as close human interaction, might require complementary approaches. Finally, although DG-MAP focuses its efforts on coordinating multiple arms effectively, there is still potential improvement in solving collaborative tasks as described in [12]. These points highlight opportunities to build upon the DG-MAP framework, extending its applicability and performance range in future iterations.

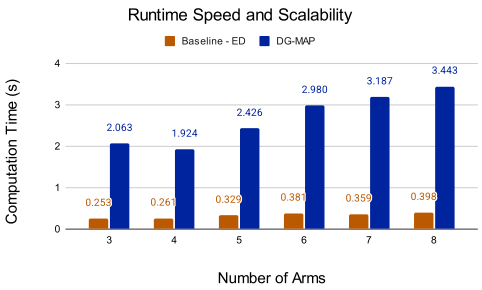

Figure 5: Computation time of Baseline-ED compared to DG-MAP in seconds

A.2 Implementation Details

The diffusion models (ϵ_{θ_1} and ϵ_{θ_2}) were trained, and all planning experiments conducted, on a system equipped with a 32-core Intel i9-14900K CPU and an NVIDIA GeForce RTX 4090 GPU. The algorithm is implemented in Python and used PyBullet [24] as the simulator, building upon the official codebase released by Ha et al. [14] for environment simulation and integrating the diffusion policy framework from Chi et al. [23].

The observation vector for the single-arm model (ϵ_{θ_1}) comprised 6 joint values, 7 end-effector pose values (position + quaternion), 7 target end-effector pose values, 30 values representing key link positions, and 7 values for the fixed base pose, resulting in a dimension $|o_i| = 57$ [14]. The dual-arm model (ϵ_{θ_2}) used a concatenated observation from the interacting pair, resulting in dimension $|\hat{o}_i| = 114$. We adopted the CNN-based UNet architecture with FiLM conditioning as detailed in [23], employing the squared cosine noise schedule proposed in iDDPM [29].

Within the DG-MAP search (Algorithm 1), nodes represent combined states for all N arms. The quality of expanding a node with a specific combination of candidate plan segments $\{\Delta \boldsymbol{q}_i^{b_i}\}_{i=1}^N$, where $\Delta \boldsymbol{q}_i^{b_i}$ is the b_i-th plan segment for arm i over horizon T_p is evaluated using a cost function as defined below,

$$g(\mathcal{N}) = \sum_{i=1}^{N} \|\Delta \boldsymbol{q}_i^{b_i}(t)\|_2 + d_{\text{pos}}(\boldsymbol{p}_i^{\text{ee}}(\boldsymbol{q}_i(T_p-1)), \boldsymbol{p}_i^{\text{goal}}) + d_{\text{rot}}(\boldsymbol{p}_i^{\text{ee}}(\boldsymbol{q}_i(T_p-1)), \boldsymbol{p}_i^{\text{goal}}) + P_{\text{coll}},$$

where $\|\Delta \boldsymbol{q}_i^{b_i}(t)\|_2$ is the action magnitude, $d_{\text{pos}}(\boldsymbol{p}_i^{\text{ee}}(\boldsymbol{q}_i(T_p-1)), \boldsymbol{p}_i^{\text{goal}})$ is the position residual and $d_{\text{rot}}(\boldsymbol{p}_i^{\text{ee}}(\boldsymbol{q}_i(T_p-1)), \boldsymbol{p}_i^{\text{goal}})$ is the orientation residual at the end of the segment T_p, and P_{coll} is a large penalty applied if *any* collision occurs within the combined plan segment over the horizon $[0, T_p)$. Specific hyperparameter values used are listed in Table 4.

Table 4: Hyperparameters for DG-MAP Training and Planning.

Parameter	**Value**
Diffusion Model & Training	
Positional encoding size	256
Number of denoising steps	100
UNet layers (channels)	[256, 512, 1024]
Training epochs	100
Batch size	4096
Observation horizon T_o	2
Prediction horizon T_p	16
Action sequence length T_a (used for training)	1
$\boldsymbol{\epsilon}_{\theta_1}$ Observation dim $\lvert o_i \rvert$	57
$\boldsymbol{\epsilon}_{\theta_2}$ Observation dim $\lvert \hat{o}_i \rvert$	114
$\boldsymbol{\epsilon}_{\theta_1}$ Action dim $\lvert \Delta \boldsymbol{q}_i \rvert$	6
$\boldsymbol{\epsilon}_{\theta_2}$ Action dim $\lvert \Delta \boldsymbol{q}_i \rvert$	6
Policy learning rate	1e-4
Learning rate weight decay	1e-6
Polyak update coefficient (EMA decay)	0.001
Optimizer	AdamW
Planning & Cost Function	
Planning batch size B (candidate samples per arm)	10
Planning timeout	60s
Workspace radius (defines task difficulty)	0.85
Plan segment collision penalty P_{coll}	10

A.3 Effect of value-based diffusion models

To address the challenge of out-of-distribution performance in diffusion models, the DiffusionQL [30] method was recently introduced. Motivated by this, we trained two new models, $\boldsymbol{\epsilon}_{\theta_1}^{\texttt{QL}}$ and $\boldsymbol{\epsilon}_{\theta_2}^{\texttt{QL}}$, to test the effectiveness of DiffusionQL within our planning framework. First, we generated offline RL datasets compatible with DiffusionQL training. This was done by taking the data collected for our original single-arm and dual-arm models and "retracing" it through the environment simulator, applying the reward function described in [14]. Using these offline RL datasets, we then adapted the DiffusionQL critic for use in a receding horizon context. Specifically, we modified the critic to evaluate entire action sequences over the full prediction horizon. Finally, during the planning stage, we sampled 50 candidate action sequences per arm from the trained DiffusionQL actors. We subsequently employed the adapted critic to score these

Method	**Success** (↑)	**Steps** (↓)
DG-MAP ($\boldsymbol{\epsilon}_{\theta_1}, \boldsymbol{\epsilon}_{\theta_2}$)	0.890	5390
DG-MAP ($\boldsymbol{\epsilon}_{\theta_1}^{\texttt{QL}}, \boldsymbol{\epsilon}_{\theta_2}^{\texttt{QL}}$)	**0.908**	**5254**

Table 5: Success rate (%, higher is better) along with average number of steps (lower is better) taken by the DG-MAP variants to complete the full cycle of multi-arm pick-and-place task.

sequences and selected the top 10 sequences for each arm. These selected sequences were then used as input for the remainder of our established planning algorithm.

Table 5 presents results on the multi-arm pick-and-place task. Here, the DG-MAP variant utilizing DiffusionQL models achieves a slightly higher success rate (90.8% vs. 89.0%) and completes the task in marginally fewer steps on average (5254 vs. 5390) compared to the standard variant. This suggests that incorporating reward signals via Q-learning might offer a small advantage in optimizing for success and efficiency on this complex, multi-stage application task, potentially by better handling scenarios that deviate from the expert demonstrations.

Arms	**Average**			
	Baseline LD	Baseline ED	DG-MAP $(\boldsymbol{\epsilon}_{\theta_1}, \boldsymbol{\epsilon}_{\theta_2})$	DG-MAP $(\boldsymbol{\epsilon}^{\texttt{QL}}_{\theta_1}, \boldsymbol{\epsilon}^{\texttt{QL}}_{\theta_2})$
3	0.412	0.965	**0.975**	0.973
4	0.039	0.961	0.968	**0.970**
5	0.075	0.934	0.945	**0.955**
6	0.055	0.906	**0.928**	0.923
7	0.038	0.917	**0.926**	0.924
8	0.022	0.907	**0.924**	0.919

Table 6: **Average Success Rates**(%, higher is better) for 3–8 arms across different methods

However, examining the average success rates across the general goal-reaching benchmarks (Table 6), the performance difference between the two DG-MAP variants is minimal and inconsistent across different numbers of arms. For $N = 4, 5$, the QL variant shows a slight edge, while for $N = 3, 6, 7, 8$, the standard variant performs marginally better. The absolute difference in average success rate is typically less than 1% between the two. This indicates that for the broad set of goal-reaching tasks, both the behavioral cloning and Diffusion Q-Learning approaches provide highly effective underlying generative models for trajectory proposals and repairs within the DG-MAP planning framework. The core benefit appears to stem from the planner's structure combined with the generative capabilities of diffusion models, rather than a strong preference for either the standard or QL training objective in the general case.