Collaborative Multi-Robot Non-Prehensile Manipulation via Flow Matching Co-Generation

Yorai Shaoul[1], Zhe Chen[2,*], Naveed Gul Mohamed[2,*], Federico Pecora[2], Maxim Likhachev[1], Jiaoyang Li[1]

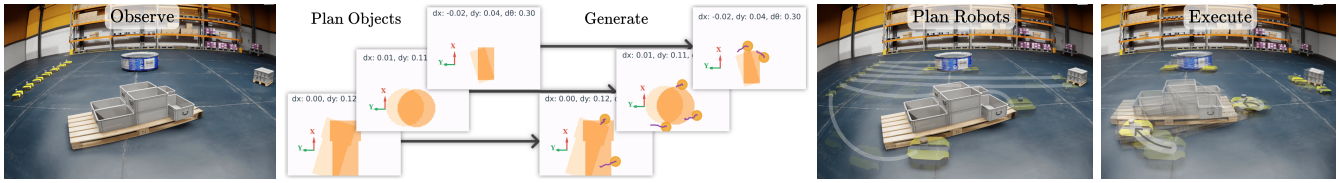

Fig. 1: The Generative Collaboration (GCo) framework learns components that are hard to model and plans those that are easy. From left to right: one iteration of GCo, coordinating seven robots collaboratively manipulating three large objects in a shared space. Given image observations, it proposes motions for all objects, jointly generates contact points and manipulation trajectories with flow matching co-generation, and plans multi-robot paths to convey the team to manipulation sites.

***Abstract*— Coordinating a team of robots to reposition multiple objects in cluttered environments requires reasoning jointly about where robots should establish contact, how to manipulate objects once contact is made, and how to navigate safely and efficiently at scale. Prior approaches typically fall into two extremes–either learning the entire task or relying on privileged information and hand-designed planners–both of which struggle to handle diverse objects in long-horizon tasks. To address these challenges, we present a unified framework for collaborative multi-robot, multi-object non-prehensile manipulation that integrates flow matching co-generation with anonymous multi-robot motion planning. Within this framework, a generative model co-generates contact formations and manipulation trajectories from visual observations, while a novel motion planner conveys robots at scale. Crucially, the same planner also supports coordination at the object level, assigning manipulated objects to larger target structures and thereby unifying robot- and object-level reasoning within a single algorithmic framework. Experiments in challenging simulated environments demonstrate that our approach outperforms baselines in both motion planning and manipulation tasks, highlighting the benefits of generative co-design and integrated planning for scaling collaborative manipulation to complex multi-robot, multi-object settings. Visit `gco-paper.github.io` for demonstrations.**

I. Introduction

Multi-robot collaboration is a central ambition in robotics, promising systems of robots that can accomplish tasks beyond the capabilities of an individual. A particularly compelling example is multi-robot manipulation, where robots must physically coordinate to move and position objects in their environment. Applications range from the home, where teams of robots tidy living spaces, to warehouse logistics, where robots transport and align inventory.

Multi-robot manipulation poses unique challenges. Unlike purely navigational coordination, robots must reason about their own motions, the coupled dynamics of objects under contact, and how to assign and reassign interactions across multiple robots and objects. This coupling makes the problem complex: the space of possible robot–object assignments, contact formations, and coordinated motions grows quickly with the number of robots and objects.

Prior methods approach this challenge from two extremes. Learning-based approaches aim to capture the entire process from data, which can model complex dynamics but often sacrifices generalization and scalability [1]. Planning-based approaches [2] leverage structure to compute coordinated trajectories, but rely heavily on perfect models and can be brittle when deployed in the real world. In contrast, we argue for a middle ground: planning should be used wherever structure allows, and learning should be applied where models are uncertain or intractable to design by hand.

In this paper, we present a unified framework for collaborative manipulation that embodies this principle, with three contributions: (i) The Generative Collaboration framework (GCo) for scalable multi-robot, multi-object collaborative manipulation. (ii) A flow matching co-generation interaction model that proposes feasible and diverse contact and motion strategies for objects observed through images alone. To the best of our knowledge, this is the first use of discrete-continuous flow matching co-generation in robotics. And (iii) a new multi-robot motion planner (GSPI) that computes safe, coordinated trajectories for conveying multiple entities to goal configurations. We show that the GCo framework, and individually the GSPI algorithm, both effectively solve more complex and larger scale problems than existing methods to manipulation and multi-robot motion planning.

II. Problem Formulation

We study the problem of multi-robot collaborative manipulation, where a team of robots jointly transports multiple objects to designated goal configurations through non-prehensile interactions (i.e., pushing).

Let $\mathcal{R} = \{R^1, R^2, \ldots, R^N\}$ denote the set of N holonomic *robots* operating in a shared workspace $\mathcal{W} \subseteq \mathbb{R}^2$, modeled as disks of fixed radius $r > 0$. The configuration of robot R^i is $q_R^i \in \mathcal{Q}_R^i \subseteq \mathbb{R}^2$, and a trajectory of horizon H is defined as a sequence of timed configurations and termed τ^i, with τ_t^i denoting the robot configuration at time t.

[1]Carnegie Mellon University, {yshaoul,maxim,jiaoyanl}@cs.cmu.edu
[2]Amazon Robotics, {zhecm,naveedg,fpecora}@amazon.com
During part of the work, Yorai interned at Amazon Robotics. *Equal contribution.

The environment also contains a set of M movable *objects*, $\mathcal{O} = \{o_1, \ldots, o_M\}$, where a configuration of o_j is $q_o^j \in \mathcal{Q}_o^j \subseteq SE(2)$. The workspace may additionally contain static *obstacles* that cannot be intersected by robots or objects.

A multi-object manipulation task specifies M goal configurations $q_{o,\text{goal}}^1, \ldots, q_{o,\text{goal}}^M$ for objects. We adopt an anonymous assignment between objects and goals: any object can occupy any goal, and a problem is solved once all goals are filled. This naturally captures applications like warehouse distribution, where packages can be routed to any available shipping station within a set of acceptable options.

The task objective is to compute a set of coordinated robot trajectories $\mathcal{T} = \{\tau^1, \cdots, \tau^N\}$ such that, when executed, the resulting coupled robot–object dynamics cause all objects in $\mathcal{O}$ to reach the set of goal configurations $\{q_{o,\text{goal}}^i\}_{i=1}^M$. Importantly, the trajectories need not remain fixed throughout execution. To cope with model mismatch and real-world uncertainty, our framework interleaves planning with execution, allowing dynamic replanning and reassignment of robot–object interactions as needed.

III. Background

We now situate our work within the broader literature. We survey related work in Sec. III-A and provide relevant background on flow matching for generative modeling (III-B) and on anonymous multi-robot motion planning (III-C).

A. Related Work: Multi-Robot Manipulation

Much of the research on multi-robot coordination has centered on geometric collision avoidance, where robots are tasked with reaching goal positions while avoiding one another and obstacles, mostly in simplified 2D grid environments [3–5], with some applied to robotic arms and drones [6–8]. Far less attention has been given to collaboration through interaction, where robots must not only avoid each other but also physically engage with objects.

Early approaches addressed only parts of the problem or relied on strong assumptions, avoiding the combinatorial and physical challenges of jointly reasoning about assignments, motions, and contact dynamics. For instance, Hartmann *et al.* [9], addressed the task-and-motion planning (TAMP [10]) formulation of multi-robot rearrangement, scaling impressively with objects and robots, but disregarded robot-object physical interactions. Tang *et al.* [2] considered long-horizon planning for physical manipulation, but restricted attention to a single object whose exact shape and physical parameters are known, enabling geometric analysis.

Recent work has sought to relax these assumptions by learning all aspects of collaborative manipulation directly. Feng *et al.* [1], along with related efforts [11, 12], pursued hierarchical reinforcement learning (HRL), in which low-level controllers handled locomotion while higher-level policies determined how robots engage with objects. Although in principle such approaches could scale, in practice they have only been demonstrated on drastically simplified problems—typically no more than two or three robots, a single object, and environments with few or no obstacles. Even within these restricted settings, HRL policies often struggle with large transformations and obstacle avoidance. By contrast, our work introduces algorithms that scale with the number of robots and objects, even in cluttered spaces where robots may be outnumbered by objects or vice versa.

In this work, we depart from approaches that either plan under strong modeling assumptions or learn from scratch in restricted settings, and instead leverage structure where possible while learning only what is hard to model. We treat collision-free navigation and task assignment as structured subproblems amenable to planning, and learn contact strategies (i.e., how robots engage objects and coordinate their influence) from data. This perspective is supported by the recent success of diffusion and flow matching methods for learning local skills [13–15] and by the ability of planning algorithms to globally coordinate large robot teams [5, 16].

B. Background: Flow Matching Co-Generation

Flow matching, a flexible method for generative modeling, has recently found success in various domains, including imitation learning for robotics [14], image generation [17], and protein design [18]. In this work, we leverage flow matching models to determine where robots should create contact with objects, how many contact points to create, and how to move once contact is established. Let us begin with background on three flavors of flow matching: continuous, discrete, and continuous-discrete co-generation.

Continuous Flow Matching casts generation as learning a time-dependent velocity field u_t that transports a base distribution p_0 into the data distribution p_1 [17]. For an interpolated sample ${}^tX := ({}^1X - {}^0X)t + {}^0X$ between a noisy sample ${}^0X \sim p_0$ and a data sample ${}^1X \sim p_1$, the flow matching objective is to learn an estimated velocity field u_t^θ that matches the sample velocity.

$$\theta^* = \arg\min_\theta \mathbb{E}_{{}^0X \sim p_0, {}^1X \sim p_1, t \in [0,1]} \|({}^1X - {}^0X) - u_t^\theta({}^tX)\|^2.$$

Generating new samples ${}^1\hat{X}$ is done by integrating the ODE

$$\frac{d}{dt}{}^tX = u_t^\theta({}^tX),\ {}^0X \sim p_0, \quad {}^1X = {}^0X + \int_0^1 u_t^\theta({}^tX)dt. \quad (1)$$

While flow matching is powerful for continuous data (e.g., trajectory waypoints), categorical domains where samples are sequences of values from a fixed set (e.g., choosing contact points from a finite set of image pixels or text generation [19]) benefit from a different treatment.

Discrete Flow Matching [19] is a framework for generating sequences K of discrete samples when elementes can only take on values from a discrete, finite, state space $\mathscr{K}$. Let a sample from the space of discrete sequences of length $B \in \mathbb{Z}^{>0}$ elements be $K := \{K^{(1)}, \cdots K^{(B)} \mid K^{(b)} \in \mathscr{K}\}$. Given "noise" and data sequences ${}^0K \sim p_{\text{d},0}$ and ${}^1K \sim p_{\text{d},1}$, an interpolated sample ${}^tK$ is obtained by mixing

$${}^tK^{(b)} \sim \text{Cat}\big((1-t)\delta_{{}^0K^{(b)}} + t\delta_{{}^1K^{(b)}}\big),$$

where δ is the Dirac delta function and Cat is the categorical distribution. Simply put, value $K^{(b)} \in {}^tK$ takes on the base value ${}^0K^{(b)}$ with probability $(1-t)$ or the data value ${}^1K^{(b)}$ with probability $t \in [0,1]$.

The discrete flow matching objective is to predict $\delta_{{}^1K}$ directly from an interpolated ${}^tK$. This prediction, which is sampled from the generated distribution $u^{\phi}_{\mathrm{d},t}({}^tK)$ named the discrete velocity, is supervised with a cross-entropy (CE) loss

$$\phi^* = \arg\min_{\phi} \mathbb{E}_{{}^0K,{}^1K,t}\left[\sum_{b=1}^{B} \mathrm{CE}\Big(u^{\phi}_{\mathrm{d},t}({}^tK^{(b)}),\ \delta_{{}^1K^{(b)}}\Big)\right]$$

with ${}^0K \sim p_{\mathrm{d},0}$, ${}^1K \sim p_{\mathrm{d},1}$, $t \in [0,1]$. Generating new discrete samples ${}^1\hat{K}$ is done by simulating a categorical Markov chain, where each element $K^{(b)}$ transitions according to the predicted discrete velocity $u^{\phi}_{\mathrm{d},t}$ over $t \in [0,1]$,

$$ {}^0K \sim p_{\mathrm{d},0}, \quad {}^{t+\Delta t}K^{(b)} \sim \mathrm{Cat}\big(u^{\phi}_{\mathrm{d},t}({}^tK^{(b)})\big) \tag{2}$$

The base distribution $p_{\mathrm{d},0}$ we adopt in this work is the "mask" distribution where all elements are set to a single value `[M]`.

Flow Matching Co-Generation [18] attempts to jointly generate tightly coupled discrete and continuous samples. Let a joint state be $({}^tX, {}^tK)$ where ${}^tK$ denotes discrete valued sequence and ${}^tX$ denotes continuous valued samples. A co-generative flow matching model learns a pair of fields

$$u^{\phi}_{\mathrm{d},t}({}^tX, {}^tK), \quad u^{\theta}_{\mathrm{c},t}({}^tX, {}^tK),$$

where $u^{\theta}_{\mathrm{c},t}$ is a continuous velocity and $u^{\phi}_{\mathrm{d},t}$ is a discrete velocity, both conditioned on the discrete and continuous variables. The training objective is the sum of continuous and discrete flow matching losses, and generating new joint samples $({}^1\hat{X}, {}^1\hat{K})$ is done by simultaneously integrating the continuous ODE (Eq. 1) and evolving the discrete state ${}^tK$ via a categorical Markov chain with transition probabilities given by $u^{\phi}_{\mathrm{d},t}({}^tX, {}^tK)$ (Eq. 2), both starting from $({}^0X, {}^0K) \sim (p_0, \delta_{\texttt{[M]}})$. In this work, we show that flow matching co-generation can be an effective choice for learning contact points and manipulation trajectories.

C. Background: AMRMP

Anonymous multi-robot motion planning (AMRMP) is a foundational problem in robotics: given N robots with start configurations $Q_{R,\text{start}} \in \mathcal{Q}_R^1 \times \mathcal{Q}_R^2 \times \cdots \times \mathcal{Q}_R^N$ and a set of interchangeable goals $Q_{R,\text{goal}}$, the task is to compute safe trajectories $\mathcal{T}$ such that the collection of terminal states coincides with $Q_{R,\text{goal}}$.

Most prior work on AMRMP studied the graph-based abstraction known as anonymous multi-agent path finding (AMAPF), where robots are modeled as points moving along vertices and edges [16, 20–22]. Within this setting, most methods focus on optimal or bounded sub-optimal path finding and goal assignments, often building on Conflict-Based Search (CBS) [3] to leverage its strong theoretical guarantees. Other approaches emphasize scalability over optimality. TSWAP [16], for instance, is a highly efficient rule-based algorithm that detects potential deadlocks and resolves them by swapping goals between robots.

Encouraged by the success of algorithms in these discrete point-robot domains, recent works have extended MAPF (i.e., the non-anonymous counterpart to AMAPF) to continuous settings by replacing grid edges with *motion primitives*, short kinematically feasible transitions between configurations [6, 7, 23]. Unfortunately, similarly adapting AMAPF algorithms is not trivial: CBS-based methods lose much of their efficiency without grid-specific enhancements, and highly efficient rule-based methods like TSWAP can deadlock when applied to non-point robots and motion primitives.

In this work, we introduce GSPI ("Giuseppe"), a new AMRMP algorithm that combines the efficiency of rule-based MAPF methods with the deadlock resolution strategies inspired by TSWAP and C-UNAV [24], adapted to motion-primitive domains without introducing deadlocks or livelocks in practice. GSPI forms a central building block of our generative collaboration framework, and with this background in place we now turn to its introduction.

Algorithm 1 GCo: Generative Collaboration for Multi-Robot Manipulation

Require: World state $\mathcal{X}_w$, object set $\mathcal{O}$, goal set $\{q^i_{o,\text{goal}}\}_{i=1}^M$, robot set $\mathcal{R}$, policy π_θ, planner GSPI

1 **while** not all objects in goal set **do**
2 $\quad \mathcal{X}_w,\ \{\mathcal{I}^j\}_{j=1}^M \leftarrow$ OBSERVE()
3 $\quad \{\tau^j_{\text{obj}}\}_{j=1}^M \leftarrow$ PLAN(GSPI, $\mathcal{O}, \mathcal{X}_w$)
4 $\quad \{T^j\}_{j=1}^M \leftarrow$ EXTRACTTRANSFORMS($\{\tau^j_{\text{obj}}\}$)
5 $\quad \{B^j\}_{j=1}^M \leftarrow$ BUDGETALLOCATION($\mathcal{R}, \mathcal{O}$)
6 $\quad \{\mathcal{K}^j, \mathcal{T}^j_{\text{manip}}\}_{j=1}^M \leftarrow \pi_\theta(\{\mathcal{I}^j, T^j, B^j\}_{j=1}^M)$
7 $\quad \mathcal{T}_{\text{coop}} \leftarrow$ PLAN(GSPI, $\mathcal{R}, \{\mathcal{K}^j\}$)
8 $\quad$ EXECUTE($\mathcal{T}_{\text{coop}} \oplus \cup\{\mathcal{T}^j_{\text{manip}}\}_{j=1}^M$)

IV. Generative Collaboration

We introduce GCo, a framework for interleaving learning and planning in collaborative multi-robot systems. The core principle of GCo is simple: **plan what can be modeled well, and learn what cannot.**

In the context of non-prehensile manipulation, we identify two distinct operation modes that must be addressed by a team of robots: they must generate manipulation motions that define how robots interact with objects to induce desired transformations, and they must generate cooperative motions that bring robots into the appropriate configurations to realize those interactions. Manipulation motions depend on objects' physical dynamics as well as stochasticity in execution—making them natural candidates for learning. In contrast, cooperative motions involve the geometric feasibility of moving multiple entities through clutter, a domain amenable to planning. GCo unifies these two modes in a closed-loop framework, interleaving generation of data-driven manipulation motions and planning of cooperative trajectories to realize them (Alg. 1).

Each iteration begins with perception (line 2). Objects o_j are observed via images $\mathcal{I}^j$, which are $w \times w$ occupancy matrices, and the current world state $\mathcal{X}_w$ is updated with the states of robots and obstacles. Next, the planner GSPI is invoked at the object-level to generate short-horizon trajectories τ^j_{obj} for each object (line 3), from which short transformation subgoals T^j are extracted (line 4). Robots are then budgeted[1] across all objects with non-zero T^j, each receiving B^j robots (line 5). Finally, the learned manipulation policy, termed π_θ, co-generates object-centric contact

[1] In this work, we allocate robots as evenly as possible across all objects that need to be manipulated, with idle robots sent to wait at safe locations. When objects outnumber robots, robots are assigned by proximity.

points $\mathcal{K}^j$ and manipulation trajectories $\mathcal{T}^j_{\text{manip}}$ conditioned on images, transforms, and robot budgets (line 6). To realize these motions, GSPI is called a second time at the robot level to compute cooperative trajectories $\mathcal{T}_{\text{coop}}$, which bring robots to contact points (line 7). Once computed, these free-space cooperative trajectories are concatenated with the corresponding manipulation trajectories that begin at the reached contact points and are then executed (line 8). This process advances the system toward the object-goal set, and the loop repeats until all object-goals have been achieved.

We detail GCo's manipulation module in Sec. IV-A, describe GSPI in Sec. IV-B, and present experiments comparing GCo and GSPI to existing baselines in Sec. V.

A. Flow Matching Co-Generation for Manipulation

Our objective is to endow teams of robots with the ability to generate object-level manipulation motions that achieve target transformations $T^j \in SE(2)$. Central to this problem is deciding both *where* robots should make contact with objects and *how* they should move once contact is established. We pursue a flow matching approach and introduce three instantiations: GCo$_{C\mathcal{T}}$, which treats the task as a monolithic synthesis problem by directly generating continuous trajectories without explicit contact modeling; GCo$_{CC}$, which co-generates contact points and trajectories jointly in continuous space, parameterizing contacts in the object frame and trajectories as short waypoint sequences; and GCo$_{DC}$, which anchors contact points in the perceptual space through discrete selection while generating trajectories continuously. All three methods are effective and consistently outperform baselines, while the co-generation variants provide added stability, with the discrete–continuous formulation proving the most reliable in challenging tasks (Sec. V).

*1) Direct Continuous Trajectory Generation (*GCo$_{C\mathcal{T}}$*):* In GCo$_{C\mathcal{T}}$, multi-robot manipulation is posed as generating robot trajectories, treating their first state as a contact point:

$$\mathcal{T}^j_{\text{manip}} \in \mathbb{R}^{B_{\max} \times H \times 2}.$$

I.e., $\mathcal{T}^j_{\text{manip}}$ contains $B_{\max}$ trajectories (a maximum budget), each of length H, whose values are unconstrained. Executing $\mathcal{T}^j_{\text{manip}}$ should transform o_j by T^j. To generate, noise ${}^0\mathcal{T}^j_{\text{manip}} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ is evolved into structured trajectories via integration of a learned conditional velocity field u^θ_t:

$$\mathcal{T}^j_{\text{manip}} = \int_0^1 u^\theta_t({}^t\mathcal{T}^j_{\text{manip}} \mid \mathcal{I}^j, T^j, B^j)dt.$$

While conceptually simple, generating unconstrained sequences of waypoints for all robots can be difficult to learn.

*2) Continuous–Continuous Co-Generation (*GCo$_{CC}$*):* GCo$_{CC}$ decomposes generation into two continuous components: contact points in the local object frame, $\mathcal{K}^j$, and manipulation trajectories rooted at the origin, $\mathcal{T}^j_{\text{manip}}$,

$$\mathcal{K}^j = \{k^{j,b} \in \mathbb{R}^2\}_{b=1}^{B_{\max}}, \qquad \mathcal{T}^j_{\text{manip}} = \{\tau^{j,b} \in \mathbb{R}^{H\times 2}\}_{b=1}^{B_{\max}}.$$

For execution, $\tau^{j,b}$ are translated to begin at points $k^{j,b}$.

A flow matching model learns two velocity fields, u^θ_t and u^ϕ_t for contacts and trajectories, and jointly evolves $(\mathcal{K}^j, \mathcal{T}^j_{\text{manip}})$ from noisy seeds $({}^0\mathcal{K}^j, {}^0\mathcal{T}^j_{\text{manip}})$. In practice, we couple Euler integration for both continuous components:

$$\begin{aligned} {}^{t+\Delta t}\mathcal{K}^j &= {}^t\mathcal{K}^j + u^\theta_t({}^t\mathcal{K}^j \mid {}^t\mathcal{T}^j_{\text{manip}}, \mathcal{I}^j, T^j, B^j)\Delta t \\ {}^{t+\Delta t}\mathcal{T}^j_{\text{manip}} &= {}^t\mathcal{T}^j_{\text{manip}} + u^\phi_t({}^t\mathcal{T}^j_{\text{manip}} \mid {}^t\mathcal{K}^j, \mathcal{I}^j, T^j, B^j)\Delta t. \end{aligned}$$

This decomposition reduces complexity by explicitly modeling structure—where to create contact, and how to move—but still requires continuous contact generation, which unnecessarily reasons over a space larger than that perceived.

*3) Discrete–Continuous Co-Generation (*GCo$_{DC}$*):* Our key contribution is the discrete–continuous formulation GCo$_{DC}$, which couples the two components in different representational spaces. We introduce discrete contact point generation, where contacts are selected directly from the finite set of pixels in the observed image. We write a state $\mathcal{K}^j$, holding information about $B_{\max}$ contact points for an object o_j, as a sequence of $2B_{\max}$ discrete tokens $\{c^j_i\}_{i=1}^{2B_{\max}}$, with each consecutive pair being equivalent to an (x, y) pixel coordinate in the image frame. Formally:

$$\mathcal{K}^j := \Big\{ \underbrace{c^j_1, c^j_2}_{k^j_1}, \ldots, \underbrace{c^j_{2B_{\max}-1}, c^j_{2B_{\max}}}_{k^{j,B_{\max}}} \;\Big|\; c^j_i \in \mathscr{K} = \{1, \ldots w\} \cup \{\texttt{[M]}\} \Big\}.$$

Notationally, we write $(\mathcal{K}^j)^{(i)}$ to denote the ith token in the state. This formulation anchors the contact point space to perceptual evidence while the set of manipulation motions remains flexible in $\mathbb{R}^{B_{\max} \times H \times 2}$. Like before, each trajectory is rooted at the origin and later shifted to its contact point.

To generate new contact points and manipulation trajectories $\mathcal{K}^j, \mathcal{T}^j_{\text{manip}}$, we couple Euler integration (continuous component) with Markov chain transitions (discrete component):

$$\begin{aligned} ({}^{t+\Delta t}\mathcal{K}^j)^{(i)} &\sim \text{Cat}\left(\left(u^\theta_{\text{d},t}({}^t\mathcal{K}^j \mid {}^t\mathcal{T}^j_{\text{manip}}, \mathcal{I}^j, T^j, B^j\right)^{(i)}\right) \\ {}^{t+\Delta t}\mathcal{T}^j_{\text{manip}} &= {}^t\mathcal{T}^j_{\text{manip}} + u^\phi_{\text{c},t}({}^t\mathcal{T}^j_{\text{manip}} \mid {}^t\mathcal{K}^j, \mathcal{I}^j, T^j, B^j)\Delta t. \end{aligned}$$

4) Variable Robot Budget: In practice, only a limited number of robots may be available for manipulation. Training separate models for each team size would be inefficient. Instead, we train a single model that respects a robot budget B. Our model always produces $B_{\max}$ contacts and manipulation trajectories, and masks entries that are not needed. We use the special token `[M]` to denote unused outputs: in the discrete case, $k^{j,b} = (\texttt{[M]}, \texttt{[M]})$; in the continuous case, contacts within ε of the mask token's value are treated as masked, i.e., when $\|k^{j,b} - (\texttt{[M]}, \texttt{[M]})\| < \varepsilon$. In training, supervision penalizes generated budgets $\hat{B}^j$ exceeding the allocated budget B^j: $\mathcal{L}_{\text{budget}} = \sum_{j=1}^{M} \text{ReLU}(\hat{B}^j - B^j)$.

B. GSPI: Robot- and Object-Level AMRMP

In GCo, planned objects transformations are supplied to π_θ (Sec. IV-A) for generating manipulation strategies, and robots plan motions to reach contact points. Both planning stages falls under AMRMP: entities move in continuous space, are indistinguishable, and must reach a set of interchangeable goals even when operating in tight proximity to obstacles. Existing AMRMP planners are difficult to apply directly; while they handle open spaces well, they struggle

when robots must approach surfaces or navigate narrow clearances (Fig. 4)—a key requirement in our setting.

One might instead extend scalable AMAPF algorithms from point-robot grids to continuous motion-primitive domains, but naïve adaptations either impose coarse discretizations or introduce deadlocks (Sec. III-C). To address these challenges, we combine the efficiency of the grid-based method PIBT with goal-swapping ideas inpired by TSWAP and C-UNAV to develop GSPI, a new AMRMP algorithm that scales robustly in continuous, cluttered environments. Specifically, GSPI generalizes PIBT—a non-anonymous algorithm originally designed for point robots on regular grids—to the anonymous setting with circular robots and motion primitives. The remainder of this section details the design of GSPI, starting from PIBT and outlining how our proposed algorithm overcomes its limitations.

Algorithm 2 GSPI: Goal Swapping with Priority Inheritance
Blue lines are modifications to the PIBT algorithm.

Require: Robot set $\mathcal{R} = \{R^1, \ldots, R^N\}$ with configurations $q_R^i \in \mathcal{Q}_R^i$, anonymous goals $Q_{R,\text{goal}}$, motion-primitive generators $\mathcal{G}^i(q_R^i)$, initial priority order $\prec$, current target map $g : \mathcal{R} \to Q_{R,\text{goal}}$

1 Initialize $\prec$ to $\prec_{\text{init}}$, random non-repeating, numbers in $[0, 1]$.
2 **while** some R^i not at any goal **do**
3 $\quad \mathcal{E} \leftarrow \emptyset$ ▷ Edge reservations (per step).
4 $\quad$ **for each** R^i in $\prec$ **do** TRYSWAP(R^i, g)
5 $\quad$ **for each** R^i in $\prec$ **do** TRYMOVE(R^i)
6 $\quad$ Commit $\mathcal{E}$; update q_R^i; demote R^i at goals to $\prec_{\text{init}}$, increment others
7 **return** $\mathcal{T}$
8 **function** TRYMOVE(R^i)
9 $\quad \mathcal{E}^i \leftarrow \mathcal{G}^i(q_R^i)$; order by heuristic to $g(R^i)$
10 $\quad$ **for** each $e^i \in \mathcal{E}^i$ **do** ▷ Choose candidate primitives (i.e., edges).
11 $\quad\quad$ **if** $\exists e^j \in \mathcal{E}$ with edge–edge conflict, **continue** ▷ Primitive blocked
12 $\quad\quad U \leftarrow \{R^j \neq R^i \mid R^j$ unreserved in $\mathcal{E}\}$ ▷ Lower-priority.
13 $\quad\quad \mathcal{S} \leftarrow \{R^j \in U \mid e^i$ edge–state conflicts $q_R^j\}$ ▷ "Stepped-on" set.
14 $\quad\quad$ **if** $\mathcal{S} = \emptyset$ **then** $\mathcal{E} \leftarrow \mathcal{E} \cup \{e^i\}$; **return true**
15 $\quad\quad \mathcal{E}_{\text{curr}} \leftarrow \mathcal{E}$ ▷ Record current edge reservations.
16 $\quad\quad \mathcal{E} \leftarrow \mathcal{E} \cup \{e^i\}$ ▷ Tentatively commit to edge.
17 $\quad\quad$ **for** each permutation π of $\mathcal{S}$ **do**
18 $\quad\quad\quad$ ok $\leftarrow$ **true**
19 $\quad\quad\quad$ **for** each $R^j \in \pi$ **do**
20 $\quad\quad\quad\quad$ **if not** TRYMOVE(R^j) **then** ok $\leftarrow$ **false**; **break**
21 $\quad\quad\quad$ **if** ok **then return true** ▷ All recursive calls succeeded.
22 $\quad\quad\quad$ **else** $\mathcal{E} \leftarrow \mathcal{E}_{\text{curr}}$ ▷ Roll back.
23 $\quad e_{\text{wait}}^i \leftarrow (q_R^i \to q_R^i)$; $\mathcal{E} \leftarrow \mathcal{E} \cup \{e_{\text{wait}}^i\}$; **return false**
24 **function** TRYSWAP(R^i, g)
25 $\quad$ **for each** R^j with $R^i \prec R^j$ **do** ▷ Consider lower priority robots.
26 $\quad\quad d_{i,a}, d_{i,b} \leftarrow \text{dist}(q_R^i, g(R^i)), \text{dist}(q_R^i, g(R^j))$
27 $\quad\quad d_{j,a}, d_{j,b} \leftarrow \text{dist}(q_R^j, g(R^j)), \text{dist}(q_R^j, g(R^i))$
28 $\quad\quad$ **if** $d_{i,b} < d_{i,a}$ ▷ Swap benefits higher-priority robot.
$\quad\quad\quad$ **and** $d_{i,b} + d_{j,b} \leq d_{i,a} + d_{j,a}$ **then** ▷ Benefits system.
29 $\quad\quad\quad$ swap $g(R^i) \leftrightarrow g(R^j)$
30 $\quad\quad\quad$ swap $\prec(R^i) \leftrightarrow \prec(R^j)$

1) Priority Inheritance with Backtracking (PIBT): PIBT [5] is a multi-agent path-finding algorithm for point robots traversing edges and vertices on a shared graph. Shown in Alg. 2 in black, PIBT assigns each robot a dynamically changing priority (line 1) and operates in discrete timesteps. At each step, PIBT iterates through robots by priority and asks each to commit to an action (TRYMOVE in line 5). On a robot's turn, it first sorts its available actions (transitions along edges incident to its current vertex) by a heuristic—typically the progress an action makes toward its goal (line 9)—and prunes out any actions that collide with obstacles or higher-priority robots. From the remaining valid transitions, if its best option does not collide with any other robot's position, it commits to that action (line 14). Otherwise, it identifies the blocking robot and requests that it moves away (line 20). This is done by calling TRYMOVE on the blocking robot, temporarily granting it the higher robot's priority so it can ask lower-priority robots to make way for it. This recursive chain continues until all robots commit to collision-free actions, accommodating the request of the initiating robot. If a recursive call fails—meaning a robot cannot find a feasible move—the algorithm backtracks and tries the next-best action for the parent robot. If all options fail, the robot waits (line 23). Finally, all committed motions are executed together, completing one timestep (line 6).

2) Limitations of PIBT: An important detail in PIBT is that priorities of robots en route to their goals gradually increase, while those of robots that reach their goals are reset to their lowest level (line 6). This rule gives active robots precedence over stationary ones. However, it can also create *livelocks*: when goals and paths overlap, a robot at its goal may be pushed away by another. Once the displacing robot reaches its own goal, its priority drops in turn, allowing the first robot to push back. The two can continue alternating endlessly. This failure mode stems from PIBT fixed goal assignments. In our manipulation framework, any robot can occupy any contact point. Capitalizing on this flexibility, we develop **G**oal **S**wapping with **P**riority **I**nheritance (GSPI): an AMRMP planner for non-point robots with motion primitives that shares PIBT's lightweight structure without suffering from livelocks in practice.

3) The GSPI *Algorithm:* At its core, GSPI follows the structure of PIBT and operates in iterations, where in each iteration robots commit to actions and execute them. In PIBT, each iteration consists solely of a *move* stage (line 5), where robots select and execute actions. In GSPI, we add the *swap* stage (line 4), where robots may exchange goals.

The *move stage* resembles that of PIBT but includes modifications that extend it to continuous domains with disk-shaped robots. In PIBT, each candidate action corresponds to traversing a graph edge and can, at most, be blocked by a single other robot occupying the destination vertex. In contrast, GSPI operates with motion primitives and non-zero robot footprints. As a result, a candidate action may intersect several lower-priority robots if their configurations overlap with the primitive. To handle this, GSPI constructs the set $\mathcal{S}$ of all affected robots (line 13) and asks them to yield to the higher-priority robot by creating a recursive TRYMOVE call for each one, with later recursive calls treating earlier ones as higher-priority (line 20). GSPI explores all orderings of affected robots in $\mathcal{S}$ (line 17); if one ordering succeeds, the motion is accepted, and otherwise, the system rolls back (line 22) and tries the next candidate primitive.

In the *swap stage*, GSPI iterates over robots by priority and evaluates whether exchanging goals and priorities between pairs would improve overall progress. A swap is proposed for a pair when two conditions hold (line 28): (i) the higher-priority robot is closer to the other's goal than to its own, and (ii) the exchange does not increase the total team distance. If both are satisfied, the robots swap goals and priorities (lines 29 and 30). The goal-swapping stage serves two key

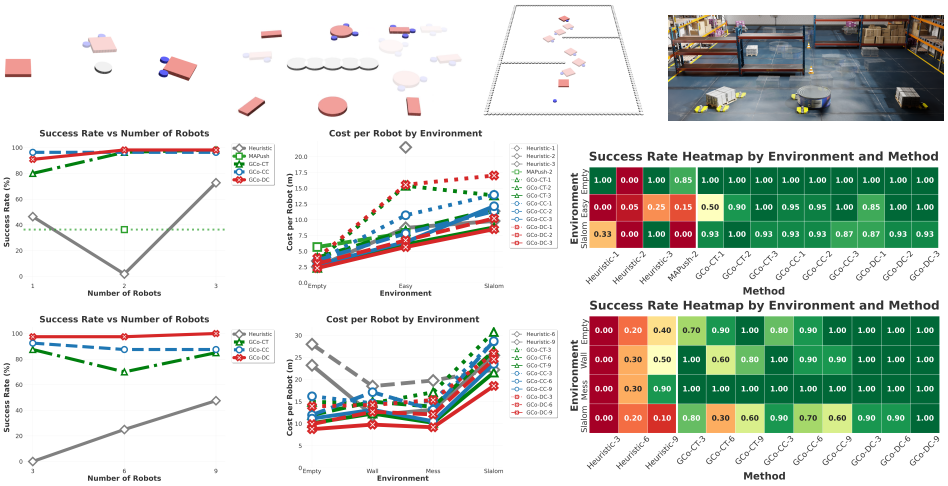

Fig. 2: Multi-robot multi-object manipulation results. GCo consistently outperformed MAPush and Heuristic baselines with GCo_{DC} performing the best. Top row: *Easy*, *Wall*, and *Slalom* (single and multi-object versions) setups. Middle and bottom rows: single- and multi-object manipulation, respectively. We report overall success rates (left column), overall average distance traveled per robot (middle), and a breakdown of success rates. Number in method names is the number of robots available in the scene.

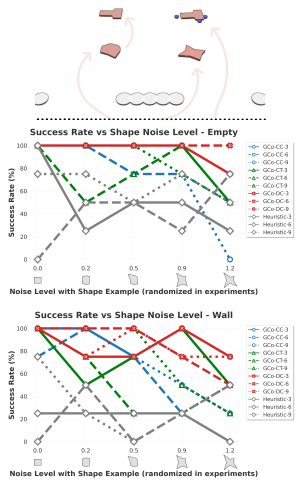

Fig. 3: Robustness to shape perturbations. Top: three robots manipulating objects drastically different from their training set. Middle and bottom: success rates in the *Empty* and *Wall* maps, both with 4 objects.

purposes. First, it relaxes fixed robot–goal assignments and promotes efficient routing in the *anonymous* case, where any robot may occupy any goal. Second, in practice, it eliminates the livelocks that arise in PIBT's fixed-goal formulation, where robots can repeatedly push one another without either remaining at a goal. Intuitively, livelocks occur in PIBT when a high-priority robot pushes a lower-priority robot off its goal and continues to push until reaching its own goal. In GSPI, such a situation triggers a goal swap: the higher-priority robot takes the closer goal, removing the need for the two robots to move head-to-head, and avoids a livelock. As we show in Sec. V-B, GSPI is a fast and scalable AMRMP algorithm.

In our manipulation framework GCo, we use GSPI both to convey robots to their assigned contact points (Alg. 1, line 7) and to compute transformations T^j for each object o_j (Alg. 1, lines 3 and 4). For the former, we run GSPI at the robot level until all robots occupy goals, and for the latter, we invoke GSPI at the object level for two iterations.

V. Experimental Evaluation

We evaluated GCo and GSPI across a diverse set of manipulation and motion-planning scenarios designed to probe performance in increasing coordination difficulty, horizon length, scale, and object complexity.

A. Manipulation Experiments

We constructed 300 problems spanning three regimes: single-object manipulation, enabling direct comparison with HRL baselines (Fig. 2, middle row); multi-object manipulation, testing scalability and inter-object coordination (Fig. 2, bottom row); and out-of-distribution (OOD) scenarios, evaluating robustness to novel objects drastically different from those seen in training (Fig. 3). Success required all objects within 15 cm and 0.5 rad of targets, and cost was the average distance traveled per robot. Single-object problems were repeated 10 times, with the rest repeating 5 times.

1) Evaluated Methods: We included all GCo variants, trained on a shared dataset.[2] We compared GCo against MAPush [1], a recent reinforcement-learning method for multi-agent single-object pushing[3], and a geometric baseline inspired by [26] referred to as Heuristic, which mirrors the GCo pipeline but replaces the learned policy π_θ with a hand-designed contact and trajectory generator. Succinctly, given an object o_j and requested transformation T^j, Heuristic draws evenly spaced lines (as many as the budget allows) from the object's centroid against its requested direction of travel and selects contact points where these lines intersect with the object's boundary; emulating execution-time robot motion to contact. Heuristic generates manipulation trajectories by applying T^j to the contact points and linearly interpolating between the original and transformed points. Robot radii were 0.1 meters.

2) Single-Object Manipulation Problems: In the simplest regime, we considered a single square object and robot teams of one to three. We conducted our experiments in three maps: *Empty*, *Easy*, and *Slalom*. In *Empty* and *Easy*, robots were asked to translate the object by 1 m without rotation or by 2 m while rotating by 2 rad, respectively. In both settings, we requested motions along $\pm x$ and $\pm y$; *Empty* was obstacle-free, while *Easy* contained a single obstacle between start and goal. In *Slalom*, the object was asked to reach the other side of the maze-like environment from bottom to top, emphasizing long-horizon manipulation (Fig. 2).

3) Multi-Object Manipulation Problems: Scaling up, we requested teams of 3, 6, or 9 robots to manipulate up to five objects with randomized shapes and sizes over translations

[2]Our training set contained 20,000 samples of variable-sized boxes and cylinders. Data were collected in MuJoCo [25] by simulating teams of varying sizes executing diverse contact formations and recording outcomes.

[3]We used the official implementation and provided 2-robot models. Additionally, we simplified the problems given to MAPush to resemble those it has been trained on: we allowed a translation tolerance of 0.5 m and ignored orientation errors (github.com/collaborative-mapush/MAPush).

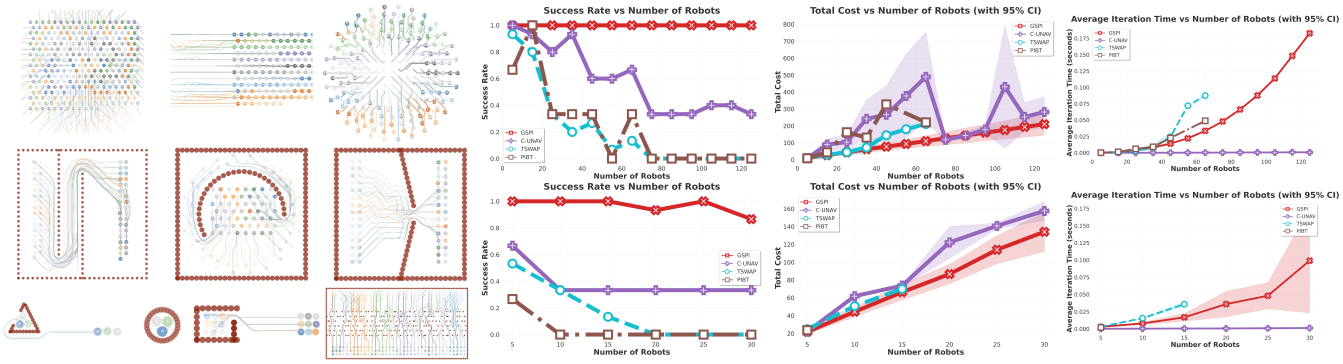

Fig. 4: GSPI analysis. Illustrations on the left show planned trajectories for all robots in freespace (top row), cluttered (middle row), and stress-test (bottom row) scenarios. The plots show success rates, costs, and iteration times in freespace (top row) and cluttered (bottom row) scenarios. GSPI solved significantly more problems than baselines with on-par or better cost.

of up to six meters and rotations of roughly 2.5 radians. For our tests, we reused *Empty* and *Slalom* maps, this time with more objects, and added two new settings: *Wall* and *Mess*. In the *Empty* and *Wall* maps, up to five objects were placed in a row and were required to move to a parallel row of goals 2 m away. *Wall* had a wide obstacle blocking the center. In the *Mess* setting, four scattered objects were required to be arranged in a square formation, and in the *Slalom* map, objects were asked to traverse tight corridors in a maze, where manipulation errors could result in blocked progress. In all problems, we shuffled the initial object–goal assignment, requiring goal swaps for efficient solutions.

4) Out-Of-Distribution Problems: Despite being trained only on boxes and cylinders, we evaluated GCO on general polygon and "T-shaped" objects. Polygons with noise σ were generated by radially shifting vertices on a rectangle (placed mid-edges and on corners) by $\mathrm{ReLU}(\epsilon)$, where $\epsilon \sim \mathcal{N}(0, \sigma)$.

5) Manipulation Results: Overall, our results indicate that GCO scales well to multi-robot, multi-object manipulation and that GCO_{DC} is consistently the most stable as task difficulty increases. In single-object tasks (Fig. 2, middle row), all GCO variants achieved higher success rates and lower costs than the baselines across most settings. MAPUSH performed well in the simplest case but quickly degraded on longer-horizon problems. HEURISTIC was inconsistent and sensitive to team size, with failures often caused by brittle contact choices that led to slip. In contrast, GCO solved nearly all problems, with cost per robot decreasing with team size, suggesting efficient use of additional robots. In multi-object tasks (Fig. 2, bottom row), a clear hierarchy emerged: GCO_{DC} achieved the highest success rates and lowest costs, with GCO_{CC} and $\text{GCO}_{C\mathcal{T}}$ trailing but still outperforming the heuristic baseline; MAPUSH was not applicable in this regime. In OOD tests, despite being trained only on boxes and cylinders, GCO remained robust on deformed shapes, with GCO_{DC} showing a slow degradation as noise increased (Fig. 3). We allowed methods long computation limits to evaluate only their manipulation capacity: 100 iterations for GCO and HEURISTIC, and 2 simulation minutes for MAPUSH. Overall, we find these manipulation results especially promising since our problem sets extend the difficulty of common prior settings in object count and shape, team size, horizon length, and obstacles [1, 11, 12, 27].

B. GSPI *AMRMP Experimental Analysis*

To evaluate GSPI in isolation from GCO, we constructed 274 AMRMP tasks with up to 125 robots spanning three regimes: obstacle-free environments for large-scale tests (Fig. 4, top plots and illustrations), obstacle-dense maps requiring tight coordination (Fig. 4, bottom plots and middle illustrations), and targeted stress tests designed to expose common failure modes (Fig. 4, bottom left illustrations). A run was successful if all robots safely reached goals; solution cost was the sum of travel distances. Problems were repeated five times with random initial robot-goal assignments.

1) Evaluated Methods: We compared GSPI to state-of-the-art and adapted baselines. The primary comparator was C-UNAV [24], which uses ORCA [28] for navigation and a goal-swapping mechanism.[4] We also included TSWAP and PIBT, adapted to operate over motion primitives rather than grid edges. GSPI, PIBT, and TSWAP used the same primitive set: short (5 cm) motions in eight cardinal directions. GSPI, C-UNAV, and TSWAP leverage goal swaps, while PIBT does not; we included PIBT due to its algorithmic similarity to GSPI to isolate the contribution of goal swapping within a PIBT-like structure. As all methods commit one multi-robot move per iteration, we also report *iteration time*, defined as the average compute time required to commit to a move.

2) Freespace AMRMP Problems: Seeking to quantify GSPI's ability to scale to large team sizes, we created three obstacle-free problem families with up to 125 robots. Our setups included transitions between formations where a high degree of symmetry and proximity between robots arose.

3) Cluttered AMRMP Problems: For evaluating coordination among obstacles and bottlenecks, we created three additional maps with team sizes up to 30 robots. These include formation transitions around obstacles, a slalom-style map that forces robots through a narrow passage, and a funnel-like variant that further tightens the bottleneck.

4) Stress-Test AMRMP Problems: To probe common failure modes under tight coupling and constrained geometry, we created five targeted stress-test problems (illustrated in Fig. 4, bottom left). These included tight layouts with densely packed goals, where robots needed to coordinate group

[4]We implemented C-UNAV by adding goal-swapping to the ORCA code from the same authors (github.com/PathPlanning/ORCA-algorithm).

motions to progress. We also included an enclosed map where starts and goals overlap, but robots cannot move. They must exchange goals in order to find a solving assignment.

5) AMRMP Results: GSPI consistently solved more instances than competing methods while achieving lower or comparable solution costs. PIBT frequently failed due to livelocks, and TSWAP often deadlocked, particularly in symmetric settings. C-UNAV performed better in open space but struggled in scenarios with higher robot- or obstacle-density. This behavior persisted despite substantial ORCA parameter tuning. In contrast, GSPI remained robust across all tested scales, including teams exceeding 100 robots. Qualitatively, the trajectories in Fig. 4 that show robots rarely crossing paths indicate that GSPI performed effective goal swapping and quickly improved on the initial random robot-goal assignments. Otherwise, trajectories would have been intertwined. Finally, the stress tests were particularly discriminative. GSPI solved 100% of the 80 runs, whereas TSWAP solved 12.5%, PIBT solved 2.5%, and C-UNAV solved 0%. These results suggest that GSPI not only scales well and effectively negotiates heavily contested regions, but also circumvents common failure modes.

VI. Conclusion

We introduced GCo, a unified framework for collaborative multi-robot, multi-object non-prehensile manipulation that combines the strengths of generative modeling and multi-robot motion planning. At its core, GCo leverages flow matching co-generation to propose contact formations and manipulation trajectories directly from perception, and integrates them with GSPI, a new scalable algorithm for anonymous multi-robot motion planning. This enables robots not only to decide *how* to interact with objects, but also to coordinate efficiently across large teams and dense clutter.

Through extensive experiments, we demonstrated that GCo consistently outperformed reinforcement learning and heuristic baselines in single- and multi-object manipulation, and that the discrete–continuous instantiation delivered the best performance. Moreover, we showed that GSPI achieved substantial scalability in AMRMP, successfully solving dense problems with more than one hundred robots where prior approaches struggle. In this light, we find that generative co-design, coupled with lightweight planning, is a powerful recipe for collaborative manipulation.

References

[1] Y. Feng, C. Hong, Y. Niu, S. Liu, Y. Yang, and D. Zhao, "Learning multi-agent loco-manipulation for long-horizon quadrupedal pushing," in *IEEE International Conference on Robotics and Automation*, 2025.

[2] Z. Tang, Y. Feng, and M. Guo, "Collaborative planar pushing of polytopic objects with multiple robots in complex scenes," *arXiv preprint arXiv:2405.07908*, 2024.

[3] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Conflict-based search for optimal multi-agent pathfinding," *Artificial Intelligence*, 2015.

[4] J. Li, W. Ruml, and S. Koenig, "Eecbs: A bounded-suboptimal search for multi-agent path finding," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 14, 2021, pp. 12 353–12 362.

[5] K. Okumura, M. Machida, X. Défago, and Y. Tamura, "Priority inheritance with backtracking for iterative multi-agent path finding," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, 2019.

[6] Y. Shaoul, I. Mishani, M. Likhachev, and J. Li, "Accelerating search-based planning for multi-robot manipulation by leveraging online-generated experiences," in *34th International Conference on Automated Planning and Scheduling*, 2024.

[7] Y. Shaoul, R. Veerapaneni, M. Likhachev, and J. Li, "Unconstraining multi-robot manipulation: Enabling arbitrary constraints in ecbs with bounded sub-optimality," in *Proceedings of the International Symposium on Combinatorial Search*, 2024.

[8] W. Hönig, J. A. Preiss, T. S. Kumar, G. S. Sukhatme, and N. Ayanian, "Trajectory planning for quadrotor swarms," *IEEE Transactions on Robotics*, 2018.

[9] V. N. Hartmann, A. Orthey, D. Driess, O. S. Oguz, and M. Toussaint, "Long-horizon multi-robot rearrangement planning for construction assembly," *IEEE Transactions on Robotics*, 2023.

[10] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Pérez, "Integrated task and motion planning," *Annual Review of Control, Robotics, and Autonomous Systems*, 2021.

[11] O. Nachum, M. Ahn, H. Ponte, S. Gu, and V. Kumar, "Multi-Agent Manipulation via Locomotion using Hierarchical Sim2Real," 2019.

[12] Z. Xiong, B. Chen, S. Huang, W.-W. Tu, Z. He, and Y. Gao, "Mqe: Unleashing the power of interaction with multi-agent quadruped environment," in *International Conference on Intelligent Robots and Systems (IROS)*, 2024.

[13] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song, "Diffusion policy: Visuomotor policy learning via action diffusion," *The International Journal of Robotics Research*, 2023.

[14] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, *et al.*, "π: A vision-language-action flow model for general robot control," *CoRR*, 2024.

[15] J. Carvalho, A. T. Le, M. Baierl, D. Koert, and J. Peters, "Motion planning diffusion: Learning and planning of robot motions with diffusion models," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023.

[16] K. Okumura and X. Défago, "Solving simultaneous target assignment and path planning efficiently with time-independent execution," *Artificial Intelligence*, p. 103946, 2023.

[17] Y. Lipman, R. T. Chen, H. Ben-Hamu, M. Nickel, and M. Le, "Flow matching for generative modeling," in *The Eleventh International Conference on Learning Representations*, 2022.

[18] A. Campbell, J. Yim, R. Barzilay, T. Rainforth, and T. Jaakkola, "Generative flows on discrete state-spaces: Enabling multimodal flows with applications to protein co-design," in *International Conference on Machine Learning*. PMLR, 2024.

[19] I. Gat, T. Remez, N. Shaul, F. Kreuk, R. T. Chen, G. Synnaeve, Y. Adi, and Y. Lipman, "Discrete flow matching," *Advances in Neural Information Processing Systems*, 2024.

[20] J. Yu and S. M. LaValle, "Multi-agent path planning and network flow," in *Algorithmic Foundations of Robotics X: Proceedings of the Tenth Workshop on the Algorithmic Foundations of Robotics*, 2013.

[21] W. Hönig, S. Kiesel, A. Tinka, J. Durham, and N. Ayanian, "Conflict-based search with optimal task assignment," in *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems*, 2018.

[22] Y. Tang, S. Koenig, and J. Li, "Ita-ecbs: A bounded-suboptimal algorithm for combined target-assignment and path-finding problem," in *International Symposium on Combinatorial Search*, 2024.

[23] B. J. Cohen, S. Chitta, and M. Likhachev, "Search-based planning for manipulation with motion primitives," in *2010 IEEE International Conference on Robotics and Automation*, 2010, pp. 2902–2908.

[24] S. Dergachev and K. Yakovlev, "Decentralized unlabeled multi-agent navigation in continuous space," in *International Conference on Interactive Collaborative Robotics*. Springer, 2024, pp. 186–200.

[25] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *IEEE/RSJ international conference on intelligent robots and systems*, 2012.

[26] J. Chen, M. Gauci, W. Li, A. Kolling, and R. Groß, "Occlusion-based cooperative transport with a swarm of miniature mobile robots," *IEEE Transactions on Robotics*, vol. 31, no. 2, pp. 307–321, 2015.

[27] S. Jeon, M. Jung, S. Choi, B. Kim, and J. Hwangbo, "Learning whole-body manipulation for quadrupedal robot," *IEEE Robotics and Automation Letters*, 2023.

[28] J. Van Den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Robotics Research: The 14th International Symposium ISRR*. Springer, 2011, pp. 3–19.