

# PushingBots: Collaborative Pushing via Neural Accelerated Combinatorial Hybrid Optimization

Zili Tang, Ying Zhang and Meng Guo

**Abstract**—Many robots are not equipped with a manipulator and many objects are not suitable for prehensile manipulation (such as large boxes and cylinders). In these cases, pushing is a simple yet effective non-prehensile skill for robots to interact with and further change the environment. Existing work often assumes a set of predefined pushing modes and fixed-shape objects. This work tackles the general problem of controlling a robotic fleet to push collaboratively numerous arbitrary objects to respective destinations, within complex environments of cluttered and movable obstacles. It incorporates several characteristic challenges for multi-robot systems such as online task coordination under large uncertainties of cost and duration, and for contact-rich tasks such as hybrid switching among different contact modes, and under-actuation due to constrained contact forces. The proposed method is based on combinatorial hybrid optimization over dynamic task assignments and hybrid execution via sequences of pushing modes and associated forces. It consists of three main components: (I) the decomposition, ordering and rolling assignment of pushing subtasks to robot subgroups; (II) the keyframe guided hybrid search to optimize the sequence of parameterized pushing modes for each subtask; (III) the hybrid control to execute these modes and transit among them. Last but not least, a diffusion-based accelerator is adopted to predict the keyframes and pushing modes that should be prioritized during hybrid search; and further improve planning efficiency. The framework is complete under mild assumptions. Its efficiency and effectiveness under different numbers of robots and general-shaped objects are validated extensively in simulations and hardware experiments, as well as generalizations to heterogeneous robots, planar assembly and 6D pushing.

## I. INTRODUCTION

Humans often interact with objects via non-prehensile skills such as pushing and rolling, especially when prehensile skills such as stable grasping is infeasible. This aspect is however less exploited in robotic systems. Most existing work treats pushing as a complementary skill to pick-and-place primitives for a single manipulator within simple environments, e.g., [1], [2], [3], [4]. Nonetheless, pushing can be particularly beneficial for low-cost mobile robots that are not equipped with a manipulator, e.g., ground vehicles, quadruped robots, and even underwater vehicles [5]. For instance, obstacles can be pushed out of the path, and target objects can be pushed to desired positions. In addition, several robots could improve the feasibility and efficiency by collaboratively and simultaneously pushing the same object at different points with different forces, which might be otherwise too heavy for individual robots. On a larger scale, as shown in Fig. 1, a fleet of such

The authors are with the School of Advanced Manufacturing and Robotics, Peking University, Beijing 100871, China. This work was supported by the National Key Research and Development Program of China under grant 2023YFB4706500; the National Natural Science Foundation of China (NSFC) under grants 62203017, U2241214, T2121002.

Corresponding author: Meng Guo, meng.guo@pku.edu.cn.



Fig. 1. Snapshots of the PushingBots system, during the simulated planar assembly via 12 robots and 14 objects (**Left**); and the hardware experiments of swapping 2 objects via 4 mini-ground vehicles (**Right**).

robots can be deployed to transport concurrently numerous objects, yielding a much higher capacity and reliability.

Despite its intuitiveness, the collaborative pushing task imposes several challenges that are typical for contact-rich tasks [6]. Different motions of the object, such as translation and rotation, can be generated by different modes of pushing such as long-side, short-side, diagonal, and caging, however with varying quality depending on the physical properties of the object [1], e.g., mass distribution, shape, and friction coefficients. Thus, long-term pushing in complex environments would require: (I) planning under tight kinematic and geometric constraints such as narrow passages and sharp turns [3]; (II) hybrid optimization to switch between different contact modes with desired pushing forces [7]; and (III) underactuated control due to constrained contact forces and unmodeled friction or slipping [8], [9], yielding a difficult problem not only for modeling and planning, but also for hybrid control. Lastly, the coordination of multiple robots to push numerous objects is also non-trivial, due to: (I) the tight coupling of object trajectories, both spatial and temporal to avoid collisions and deadlocks [10], [11]; and (II) the inherent uncertainties in the feasibility and duration of subgroups pushing different objects during online execution [12].

To tackle these challenges, this work addresses the problem of controlling mobile robots to collaboratively push arbitrary-shaped objects to goal positions in a complex environment without predefined contact modes. As illustrated in Fig. 2, the overall problem is formulated as a combinatorial hybrid optimization (CHO) including the dynamic assignment of robots to different objects, and the hybrid optimization of pushing modes and forces for each object. To begin with, the pushing task of all objects is decomposed into a set of temporally-ordered subtasks via the multi-agent path finding (MAPF) algorithm for irregular-shaped vehicles and spatial

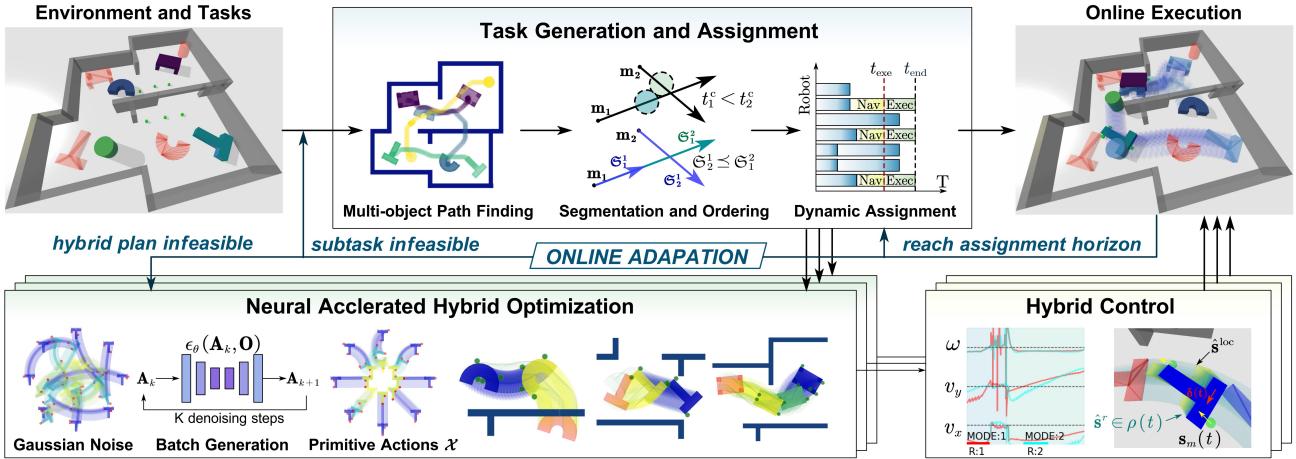


Fig. 2. Overall proposed online planning and adaptation framework, consisting of the MAPF-based task decomposition, subtask generation with partial ordering, the online rolling assignment of subtasks, the neural accelerated hybrid optimization, and the online hybrid control.

segmentation. Then, to handle the uncertain execution time of each subtask, a receding-horizon coordination algorithm is proposed to dynamically assign the robots to different subtasks based on the estimated feasibility and duration. Moreover, a learning-while-planning hybrid search algorithm iteratively decomposes subtasks via keyframe sampling and generates suitable modes to minimize feasibility loss, transition cost, and control efforts. Lastly, the online execution and adaptation strategy is presented for both the task assignment and the hybrid control. The diffusion-based accelerator is learned offline and online, to predict the sequence of keyframes and pushing modes for different subtasks, along with the estimated cost. Theoretical guarantees on completeness and performance are ensured under mild assumptions. Extensive simulations and hardware experiments are conducted to validate its performance and the effectiveness of the learned accelerator.

The main contributions of this work are two-fold: (I) the novel combinatorial-hybrid optimization algorithm with neural acceleration, for the multi-robot planar pushing problem of arbitrary objects, without any predefined contact modes or primitives; (II) the theoretical condition for feasibility and completeness, w.r.t. an arbitrary number of robots and general-shaped objects. To the best of our knowledge, this is the first work that provides such results.

## II. RELATED WORK

### A. Task and Motion Planning

The predominant direction in task and motion planning focuses on navigation and prehensile manipulation, particularly on grasping strategies and object manipulation sequences [13]. Representative tasks include sequential assembly [6], [14], rearrangement [15], and structural construction [16]. In contrast, non-prehensile planar manipulation poses unique challenges due to the lack of stable grasps. The classical single-pusher-single-slider model [1] highlights the importance of online adaptation of discrete contact decisions and force constraints, often resulting in exponential planning complexity. To manage this, motion primitives such as sticking and sliding can be encoded into integer programs [3], and various search-based optimization approaches have been proposed [7], [17]. However,

most existing methods focus on single manipulators and fixed-shape objects in simple settings, limiting their applicability to multi-robot, multi-object scenarios.

Coordinating multiple robots for multi-object tasks introduces significant complexity. Typically, tasks must be decomposed into subtasks with well-defined dependencies, which are subsequently assigned to robot subgroups, as comprehensively surveyed in [18], [19], [20]. Since many task planning problems are NP-hard or NP-complete [18], meta-heuristic methods such as local search and genetic algorithms [20] are commonly employed to improve computational efficiency. However, these approaches often assume deterministic costs, limiting their applicability in dynamic environments where task outcomes are uncertain and evolve over time. Recent work has addressed the multi-robot scheduling problem under ordered and uncertain tasks via the graph-based learning [21], distributed coordination [22], and search-based adaptation [23], demonstrating strong performance in dynamic settings. These methods operate on the abstract task graphs without incorporating the underlying physical constraints. In contrast, the decomposition and ordering of the pushing tasks for different objects in this work are not predefined, rather evaluated online based on the assigned robot subteams and interaction constraints.

### B. Collaborative Pushing

As an application of multi-robot systems, collaborative object transportation [24] can be realized via pushing, grasping, lifting, pulling, or caging. Pushing is particularly appealing since it often requires no additional hardware: robots can interact with objects using any part of their body. Early heuristic strategies, such as hand-tuned state machines for a few robots in free space [11] or occlusion-based pushing rules for convex objects [25], do not generalize well to cluttered or more complex environments. A hybrid optimization problem is formulated in [26] for several objects, but with a set of predefined pushing primitives and cost estimation. Similarly, sequential pushing steps are manually defined for heterogeneous robots in [12] to clear movable rectangular boxes in the workspace, where only one robot is assigned

for each box. Recent work [27] adopts several quadruped robots to push objects of unknown mass and friction, which however allows a limited set of pushing modes. Another line of work addresses a similar problem, but only for a single object in free [28], [29] or in cluttered [30], [31] space. Current pushing controllers broadly fall into two categories: force-based and kinematic. The work in [28] exemplifies the former. Our method, along with [25], [29], [31], belongs to the latter, relying on a kinematic controller without force sensing. Separately from the control paradigm, prior work also differs in kinematic constraints. Both [25], [29] use differential-drive robots, where [29] explicitly considers nonholonomic constraints by Jourdain's principle. By contrast, our work, together with [28], [31], targets holonomic platforms; nonetheless, our pipeline can be partially generalized to nonholonomic settings. Furthermore, several works adopt distributed schemes for collaborative pushing [28], [29], [31], which optimize interaction modes online via distributed coordination and control, enabling accurate path tracking in free space [28], [29] and even in arbitrarily cluttered environments [31], the latter being the most similar to our task setup but considers only a single object. In contrast, this work adopts a centralized framework to tackle multi-robot, multi-object pushing in cluttered environments, with a strong focus on multi-object spatiotemporal coordination, online adaptation and neural acceleration.

### C. Neural Policies for Task and Motion Planning

Due to its uncertainty and complexity, pushing a T-block has been one of the standard benchmarks in neural motion policies, e.g., self-supervised reinforcement learning [8], implicit behavioral cloning [32] and diffusion-based methods [33]. Impressive precision and robustness have been shown using only visual inputs. Similar neural policies are applied to a quadruped robot for planar pushing of regular objects with whole-body motion control [5]. For long-term tasks that require switching among different objects and tools, neural policies are proposed to predict the contact sequence [34] and the sequence of action primitives in [15], [35]. However, these aforementioned works mostly rely on a *single* manipulator or quadruped robot via one-point pushing, and human demonstrations. For the multi-robot setup in this work, a simultaneous multi-point pushing strategy is required, making it difficult for humans to provide such high-quality demonstrations. Furthermore, due to the multi-modality of both contact modes and pushing strategies, the long-term pushing task in complex environments is particularly challenging for the traditional methods of imitation learning [36]. Diffusion models on the other hand have shown great potential in modeling such multi-modality, i.e., as trajectory samplers for online motion planning [33], [37]. However, the planning efficiency and generalization of the resulting policy have not been fully explored in multi-body systems with coupled and switching dynamics. Lastly, the performance guarantee of most aforementioned methods relies on the test scenario to be close to the training datasets, which however may not always hold and can lead to failures during online execution [36]. Thus, it remains challenging to provide formal verification and performance guarantees when executing such learned policies.

## III. PROBLEM DESCRIPTION

### A. Model of Workspace and Robots

Consider a team of  $N$  robots  $\mathcal{R} \triangleq \{R_n, n \in \mathcal{N}\}$  that collaborate in a shared 2D workspace  $\mathcal{W} \subset \mathbb{R}^2$ , where  $\mathcal{N} \triangleq \{1, \dots, N\}$ . The robots are homogeneous, each with a circular or rectangular footprint. The state of each robot  $R_n$  at time  $t \geq 0$  is defined by  $\mathbf{s}_n(t) \triangleq (\mathbf{x}_n(t), \psi_n(t))$ , where  $\mathbf{x}_n(t) \in \mathbb{R}^2$  is its position and  $\psi_n(t) \in (-\pi, \pi]$  its orientation. The associated generalized velocity is  $\mathbf{p}_n(t) \triangleq (\mathbf{v}_n(t), \omega_n(t))$  with its linear velocity  $\mathbf{v}_n(t) \in \mathbb{R}^2$  and angular velocity  $\omega_n(t) \in \mathbb{R}$ . Let  $R_n(t) \subset \mathcal{W}$  denote the area occupied by robot  $R_n$  at time  $t$ . Moreover, each robot tracks the desired velocity  $\mathbf{u}_n(t) \triangleq (\hat{\mathbf{v}}_n(t), \hat{\omega}_n(t))$  with a low-level feedback controller, which induces second-order dynamics driven by the controller and external contact forces. Let  $\mathbf{S}_{\mathcal{N}}(t) \triangleq \{\mathbf{s}_n(t)\}$  denote the state of all robots. In addition, the workspace is cluttered with a set of obstacles, denoted by  $\mathcal{O} \subset \mathcal{W}$ . Thus, the free space is defined by  $\widehat{\mathcal{W}} \triangleq \mathcal{W} \setminus \mathcal{O}$ .

Lastly, there is a set of  $M > 0$  target objects  $\Omega \triangleq \{\Omega_1, \dots, \Omega_M\} \subset \widehat{\mathcal{W}}$ , where the shape of each object  $\Omega_m$  is an arbitrary polygon formed by  $V_m \geq 3$  ordered vertices  $p_1 p_2 \cdots p_{V_m}$ ,  $\forall m \in \mathcal{M} \triangleq \{1, \dots, M\}$ . Similarly, the state of each object  $\Omega_m$  at time  $t \geq 0$  is defined by  $\mathbf{s}_m(t) \triangleq (\mathbf{x}_m(t), \psi_m(t))$ , and the associated generalized velocity is  $\mathbf{p}_m(t) \triangleq (\mathbf{v}_m(t), \omega_m(t))$ , with its position  $\mathbf{x}_m(t) \in \mathbb{R}^2$ , orientation  $\psi_m(t) \in (-\pi, \pi]$ , linear velocity  $\mathbf{v}_m(t) \in \mathbb{R}^2$ , and angular velocity  $\omega_m(t) \in \mathbb{R}$ . Its occupied area at time  $t$  is denoted by  $\Omega_m(t) \subset \mathcal{W}$ . In addition, its physical parameters are known a priori, including its mass  $M_m$ , the moment of inertia  $I_m$ , the pressure distribution at the bottom surface, the coefficient of lateral friction  $\mu_m^c > 0$ , and the coefficient of ground friction  $\mu_m^s > 0$ .

**Remark 1.** The obstacles are assumed to be static and known above for simplicity, which can be relaxed to be movable, unknown or dynamic, as discussed in the sequel. ■

### B. Collaborative Pushing Modes

The robots can collaboratively push the objects by making contacts with the objects at different boundary contact points, called *collaborative pushing modes*. More specifically, as is illustrated in Fig. 1 and 2, a pushing mode for object  $\Omega_m \in \mathcal{M}$  is defined by  $\xi_m \triangleq (\xi_m, \mathbf{F}_m, \mathcal{N}_m)$ , where (I)  $\mathcal{N}_m \triangleq \{1, \dots, N_m\} \subseteq \mathcal{N}$  is the subgroup of robots assigned to push object  $\Omega_m$ ; (II)  $\xi_m \triangleq \mathbf{c}_1 \mathbf{c}_2 \cdots \mathbf{c}_{N_m}$  with  $\mathbf{c}_n \in \partial \Omega_m$  being the contact point on the boundary of the object for the  $n$ -th robot,  $\forall n \in \mathcal{N}_m$ ; and (III)  $\mathbf{F}_m \triangleq \mathbf{f}_1 \mathbf{f}_2 \cdots \mathbf{f}_{N_m}$  with  $\mathbf{f}_n \in \mathbb{R}^2$  being the contact force exerted by the  $n$ -th robot. Since the set of contact points are *not* pre-defined, the complete set of all pushing modes is potentially infinite, denoted by  $\Xi_m$ . Thus, under different pushing modes, robots can apply different pushing forces at different contact points, leading to distinct object motions (translation and rotation).

**Remark 2.** Detailed modeling of the coupled dynamics during pushing, including the decomposition of the interaction forces and frictions, can be found in the Appendix A. ■

### C. Problem Statement

The planning objective is to compute a hybrid plan, including an object trajectory  $\mathbf{s}_m(t)$ , a sequence of pushing modes  $\xi_m(t)$ , the pushing robots  $\mathcal{N}_m(t)$ , the required control  $\mathbf{u}_{N_m}$ , such that each object  $\Omega_m \in \mathcal{M}$  is moved from a given initial state  $\mathbf{s}_m^0$  to a given goal state  $\mathbf{s}_m^G$ ,  $\forall m \in \mathcal{M}$ . Meanwhile, the robots and the objects should avoid collision with each other and with all obstacles at all times. More precisely, it is formulated as a combinatorial hybrid optimization (CHO) problem as follows:

$$\begin{aligned} & \min_{T, \{\mathbf{u}_n(t), \forall n\}, \{(\mathbf{s}_m(t), \xi_m(t)), \forall m\}} \left\{ T + \alpha \sum_{t \in \mathcal{T}} \sum_{m \in \mathcal{M}} J_m(\xi_m(t), \mathbf{S}_{\mathcal{N}}(t), \mathbf{s}_m(t)) \right\} \\ & \text{s.t. } \mathbf{s}_m(0) = \mathbf{s}_m^0, \mathbf{s}_m(T) = \mathbf{s}_m^G, \forall m; \\ & \quad \mathcal{N}_{m_1}(t) \cap \mathcal{N}_{m_2}(t) = \emptyset, \\ & \quad \Omega_{m_1}(t) \cap \Omega_{m_2}(t) = \emptyset, \forall m_1 \neq m_2, \forall t; \\ & \quad \Omega_m(t) \subset \widehat{\mathcal{W}}, R_n(t) \subset \widehat{\mathcal{W}}, \forall m, \forall n, \forall t; \end{aligned} \quad (1)$$

where  $T > 0$  is the task duration to be optimized as the largest over all objects;  $\mathcal{T} \triangleq \{0, 1, \dots, T\}$ ;  $\forall t, \forall m$  and  $\forall n$  are short for  $\forall t \in \mathcal{T}, \forall m \in \mathcal{M}, \forall n \in \mathcal{N}$ , respectively; the cost function  $J_m : \Xi_m \times \mathbb{R}^{3N} \times \mathbb{R}^3 \rightarrow \mathbb{R}_+$  is a given function to measure the feasibility, stability, robustness and control cost, of choosing a certain mode and the forces given the desired object trajectory, which is different among the objects due to different intrinsics; and  $\alpha > 0$  is the weighting between the task duration and the control performance. The constraints require that: the object should reach the goal position; each robot can only participate in the pushing of one object at any time; the feasibility conditions detailed in Appendix A should hold; and all objects and robots should be collision-free.

Note that the design of function  $J_m(\cdot)$  is non-trivial and technically involved; details are given in the sequel. Moreover, the CHO problem in (1) has an extremely large decision space: there are  $M^N$  robot-object allocations per decision epoch and exponentially many timed schedules of pushing modes and contact forces over the horizon. Even ignoring discrete choices, the continuous variables already scale as  $\approx (3N+3M)\bar{T}$  when optimizing all objects jointly, rendering the overall feasible set high-dimensional and highly nonconvex, where  $\bar{T}$  denotes the task duration.

## IV. PROPOSED SOLUTION

The proposed solution consists of three interleaved components in the top-down structure, which are triggered at different conditions and rates, as shown in Fig. 2. In particular, the overall task of multi-object pushing is first decomposed into numerous subtasks via MAPF with ordering constraints in Sec. IV-A, which are assigned dynamically to subgroups of robots in a receding-horizon fashion. Then, for each subgroup to execute the assigned subtask, the optimal hybrid plan is generated via a hybrid search algorithm in Sec. IV-B, which is accelerated by a diffusion-based predictor for keyframes and pushing modes. Finally, to cope with uncertainties during execution, the object trajectory, the hybrid plan and the task assignment are all updated online, as described in Sec. IV-C. Discussions are provided in Sec. IV-D.

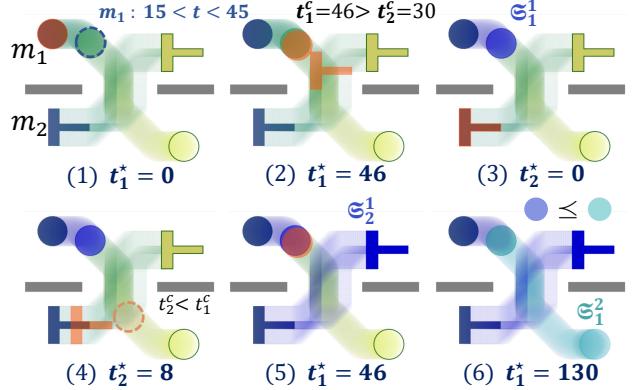


Fig. 3. Illustration of the decomposition and ordering of pushing tasks via Alg. 1, yielding 3 subtasks  $\{\mathfrak{S}_1^1, \mathfrak{S}_2^1, \mathfrak{S}_1^2\}$  with ordering  $\mathfrak{S}_2^1 \leq \mathfrak{S}_1^1$  and  $\mathfrak{S}_1^1 \leq \mathfrak{S}_2^1$ . Note that object  $m_1$  waits for object  $m_2$  to pass the corridor.

### A. Decomposition and Assignment of Pushing Tasks

Since the desired goal state of each object can be far away and the objects might be blocking each other, the multi-agent path finding (MAPF) algorithm is adopted first to synthesize a timed path for each object to reach its goal positions. Although these paths are spatially and temporally collision-free, the object velocity during collaborative pushing is hard to predict and control, making it difficult to follow the paths accurately in time and space. Thus, these paths are further decomposed into smaller segments with temporal ordering, as subtasks of the overall pushing task. Lastly, to cope with the uncertainties during execution, a dynamic task assignment algorithm is proposed to assign the subtasks to subgroups of robots as coalitions online in a receding-horizon fashion.

1) *Decomposition and Ordering via MAPF*: Given the initial and goal states  $(\mathbf{s}_m^0, \mathbf{s}_m^G)$  of each object  $m \in \mathcal{M}$ , a MAPF algorithm [38] is employed to generate a set of timed path for all objects:  $\widehat{\mathfrak{S}} \triangleq \{\mathfrak{S}_m, \forall m \in \mathcal{M}\}$ , where each path  $\mathfrak{S}_m$  of length  $L > 0$  for object  $m$  is defined as:

$$\mathfrak{S}_m \triangleq (t_0, \mathbf{s}_m(t_0)) \cdots (t_\ell, \mathbf{s}_m(t_\ell)) \cdots (t_L, \mathbf{s}_m(t_L)), \quad (2)$$

with  $\{t_\ell\}$  being the uniform time steps and  $\mathbf{s}_m(t_\ell)$  being the object state at time  $t_\ell$ . Note that MAPF is a centralized algorithm which ensures that the objects do not collide with each other nor the obstacles along the timed paths, i.e.,  $\Omega_{m_1}(t_\ell) \cap \Omega_{m_2}(t_\ell) = \emptyset, \forall m_1 \neq m_2, \forall t_\ell$ . Due to the combinatorial complexity w.r.t. the number of robots and the collision detection of non-convex polytopes, a sequential planning approach can also be adopted to reduce computation time, e.g., ordered by the length of shortest path to the goal states. Although the paths in  $\widehat{\mathfrak{S}}$  derived above are collision-free, they can be difficult to follow both spatially and temporally. Thus, to ensure safety during execution, these paths are further decomposed into smaller segments that are temporally ordered. For ease of notation, let  $\mathfrak{S}_m(t^s, t^e)$  denote the segment of  $\mathfrak{S}_m$  between the time period  $[t^s, t^e]$ , i.e.,  $\mathfrak{S}_m \triangleq (t^s, \mathbf{s}_m(t^s)) \cdots (t^e, \mathbf{s}_m(t^e))$ . Then, each path can be decomposed into a sequence of segments.

**Definition 1** (Path Segments). Each path  $\mathfrak{S}_m \in \widehat{\mathfrak{S}}$  can be decomposed into  $K_m \geq 1$  segments according to its spatial

**Algorithm 1:** Segmentation and Partial Ordering

**Input:** Paths  $\{\mathfrak{S}_m\}$   
**Output:** Segments  $\{\bar{\mathfrak{S}}_m\}$

- 1 Initialize  $t_m^* = 0, \bar{\mathfrak{S}}_m = \emptyset \forall m \in \mathcal{M};$
- 2 **while**  $\exists m \in \mathcal{M}, t_m^* < t_L$  **do**
- 3   **for**  $m \in \mathcal{M}$  and  $t_m^* < t_L$  **do**
- 4     Determine the next instance  $t_m^s$  by (5);
- 5     **if**  $t_m^s$  not found **then**  $t_m^s$  set to  $t_L$ ;
- 6     **if**  $t_m^s = t_m^*$  **then Continue**;
- 7     Add  $\mathfrak{S}_m(t_m^*, t_m^s)$  to  $\bar{\mathfrak{S}}_m$ ;
- 8      $t_m^* \leftarrow t_m^s$ ;
- 9 Determine the partial relations for  $\{\bar{\mathfrak{S}}_m\}$  by Def. 2;

intersection with other paths, i.e.,

$$\mathfrak{S}_m \triangleq \mathfrak{S}_m^1 \cup \dots \cup \mathfrak{S}_m^k \cup \dots \cup \mathfrak{S}_m^{K_m}, \quad (3)$$

where  $\mathfrak{S}_m^k \triangleq \mathfrak{S}_m(t_m^{k,s}, t_m^{k,e})$  is the  $k$ -th segment of  $\mathfrak{S}_m$  between the starting time  $t_m^{k,s}$  and the ending time  $t_m^{k,e}$ , where  $0 \leq t_m^{k,s} < t_m^{k,e} = t_m^{k+1,s} < t_m^{k+1,e} \leq L$ . ■

Thus, denote by  $\bar{\mathfrak{S}}_m \triangleq \{\mathfrak{S}_m^k, k = 1, \dots, K_m\}$  the set of segments for each object, and  $\bar{\mathfrak{S}} \triangleq \{\bar{\mathfrak{S}}_m, \forall m \in \mathcal{M}\}$  for all objects, respectively. In addition, the cumulative covered area of each segment  $\mathfrak{S}_m^k$  is defined as:  $S_m^k \triangleq \Omega_m(t_m^{k,s}) \cup \dots \cup \Omega_m(t_m^{k,e})$ , where  $\Omega_m(t)$  is the covered area of object  $m$  at  $s_m(t)$ . More importantly, the segments are temporally ordered by a partial relation defined as follows.

**Definition 2** (Partial Ordering). The partial relation  $\leq \subset \bar{\mathfrak{S}} \times \bar{\mathfrak{S}}$  satisfies that  $\mathfrak{S}_{m_1}^{k_1} \leq \mathfrak{S}_{m_2}^{k_2}$  if one of the following cases hold: (I)  $m_1 = m_2$  and  $k_1 < k_2$ ; (II)  $m_1 \neq m_2$  and  $t_{m_1}^{k_1,c} \leq t_{m_2}^{k_2,c}$  holds, where  $t_{m_1}^{k_1,c}, t_{m_2}^{k_2,c}$  are the earliest time instances when the segments  $\mathfrak{S}_{m_1}^{k_1}$  and  $\mathfrak{S}_{m_2}^{k_2}$  collide, i.e.,

$$t_{m_1}^{k_1,c} = \min \left\{ t_\ell \in [t_{m_1}^{k,s}, t_{m_1}^{k,e}] \mid \Omega_{m_1}(t_\ell) \cap S_{m_2}^k \neq \emptyset \right\}, \quad (4)$$

and  $t_{m_2}^{k_2,c}$  is defined analogously; (III)  $m_1 \neq m_2, S_{m_1}^{k_1} \cap S_{m_2}^{k_2} = \emptyset$ , and there exists  $\mathfrak{S}_m^k \in \bar{\mathfrak{S}} \setminus \{\mathfrak{S}_{m_1}^{k_1}, \mathfrak{S}_{m_2}^{k_2}\}$  such that  $\mathfrak{S}_{m_1}^{k_1} \leq \mathfrak{S}_m^k \leq \mathfrak{S}_{m_2}^{k_2}$  holds. ■

Namely, all segments of the same object are ordered by their temporal sequence, while the segments of different objects that can potentially intersect are ordered by the time that intersection first occurs. Denote by  $\text{Pre}(\mathfrak{S}_{m_2}^{k_2}) \triangleq \{\mathfrak{S}_{m_1}^{k_1} \in \bar{\mathfrak{S}} \mid \mathfrak{S}_{m_1}^{k_1} \leq \mathfrak{S}_{m_2}^{k_2}\}$  these subtasks that are ordered before  $\mathfrak{S}_{m_2}^{k_2}$ .

**Problem 1.** Given the paths  $\{\mathfrak{S}_m\}$ , find the segments  $\{\bar{\mathfrak{S}}_m\}$  such that the partial ordering  $\leq$  above is satisfied. ■

As summarized in Alg. 1 and shown in Fig. 3, an iterative algorithm is proposed to derive the segments  $\bar{\mathfrak{S}}$  from the paths  $\tilde{\mathfrak{S}}$ . For each object  $m$ , the variable  $t_m^*$  is introduced to track the largest splitting time instance, which is set to zero initially. Then, the next splitting instance  $t_m^s$  is given by:

$$t_m^s \triangleq \min_{m' \neq m} \left\{ t_m^c \mid t_m^c > t_m^s, \text{ by (4) given } \tilde{\mathfrak{S}}_m, \tilde{\mathfrak{S}}_{m'} \right\}, \quad (5)$$

where  $\tilde{\mathfrak{S}}_m \triangleq \mathfrak{S}_m(t_m^*, t_L)$  and  $\tilde{\mathfrak{S}}_{m'} \triangleq \mathfrak{S}_{m'}(t_{m'}^*, t_L)$  are the remaining paths of objects  $m$  and  $m'$  that have not been

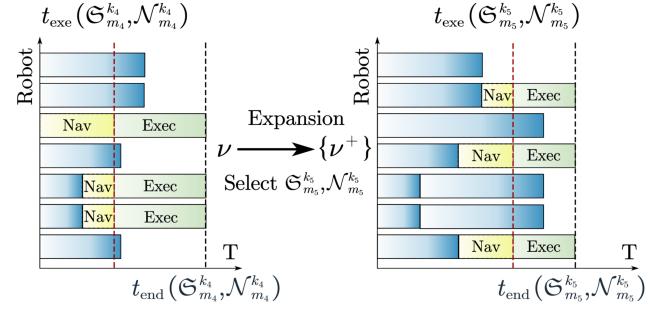


Fig. 4. Selection and expansion during the proposed receding-horizon assignment of the 12 subtasks and 7 robots, given the current planning time (red dashed line) and the horizon  $H = 5$  (blue dashed line).

segmented. If the splitting instances can not be found,  $t_m^s$  is set to  $t_L$ . Otherwise, a new segment  $\mathfrak{S}_m(t_m^*, t_m^s)$  is generated and  $t_m^*$  is updated to  $t_m^s$ . This process is repeated until  $t_m^* = t_L$  holds,  $\forall m \in \mathcal{M}$ . Lastly, the partial ordering among the segments is determined by the rules in Def. 2.

The above strategy of decomposition not only divides the overall pushing task of an object into subtasks, but also ensures a collision-free execution under uncertain task durations, as long as the partial ordering among the subtasks is satisfied.

2) *Dynamic Task Assignment:* The set of segments  $\bar{\mathfrak{S}}$  above represents the subtasks of the overall pushing task, each of which could be accomplished by a subgroup of robots. In other words, each robot would execute a sequence of subtasks, i.e., push different objects along different segments.

**Definition 3** (Task plan). The local task plan of robot  $i \in \mathcal{N}$  is denoted by  $\tau_i \triangleq (t_1, \mathfrak{S}_{m_1}^{k_1})(t_2, \mathfrak{S}_{m_2}^{k_2}) \dots (t_{L_i}, \mathfrak{S}_{m_{L_i}}^{k_{L_i}})$ , where the segment  $\mathfrak{S}_{m_e}^{k_e} \in \bar{\mathfrak{S}}$  is executed by robot  $i$  from time  $t_\ell$ ; and  $L_i > 0$  is the total length. The overall task plan of the fleet is denoted by  $\bar{\tau} \triangleq \{\tau_i, i \in \mathcal{N}\}$ . ■

The overall task plan  $\bar{\tau}$  is called *valid* if the following two conditions hold: (I) all partial orderings in  $\leq$  are respected, i.e., if  $\mathfrak{S}_{m_1}^{k_1} \leq \mathfrak{S}_{m_2}^{k_2}$ , then the segment  $\mathfrak{S}_{m_1}^{k_1}$  should be completed before the segment  $\mathfrak{S}_{m_2}^{k_2}$  is started by the assigned subgroup of robots. (II) the subgroup of robots assigned to each segment should be sufficient to push the object along the segment. (III) there should be enough time for each robot  $i \in \mathcal{N}$  to navigate between consecutive segments in its plan.

**Problem 2.** The objective of the task assignment for  $\bar{\mathfrak{S}}$  is to find a valid task plan such that the estimated time to complete all tasks is minimized, i.e.,  $\min_{\bar{\tau}} \left\{ \max_{\mathfrak{S}_m \in \bar{\mathfrak{S}}} \{t_{\text{end}}(\mathfrak{S}_m, N_m^k)\} \right\}$ . ■

To begin with, the above problem includes the job-shop problem as a special instance [18], thus is also NP-hard. The most straightforward solution is to formulate a Mixed Integer Linear Program (MILP). However, the computation complexity and the lack of intermediate solutions make it unsuitable for this application, particularly so when the pushing subtasks have large uncertainties in execution time.

Thus, an adaptive receding-horizon task assignment algorithm is proposed, which incrementally assigns subtasks in  $\bar{\mathfrak{S}}$  to robot subgroups via node expansion. The details are omitted here due to limited space. Briefly speaking, it processes at most  $H > 0$  subtasks per planning horizon to avoid intractable

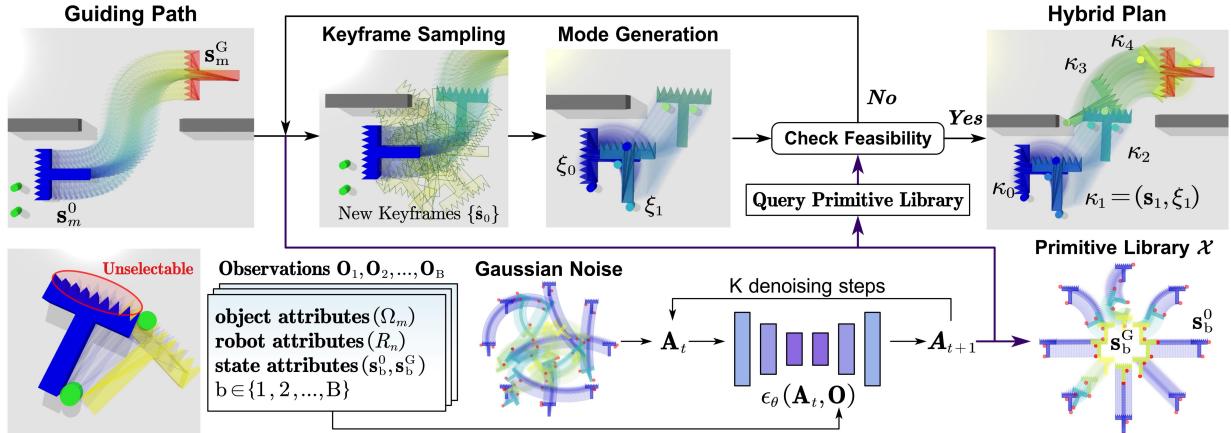


Fig. 5. Illustration of the keyframe-guided hybrid search algorithm (**Top**), which is accelerated by the diffusion-based predictor for keyframes and pushing modes (**Bottom**). Note that the multi-modal predictions are verified by the hybrid search scheme for feasibility and quality.

complexity and account for uncertainties during collaborative pushing. As illustrated in Fig. 4, the search begins with an empty assignment node, and expands it by assigning one subtask at a time, ensuring that preceding subtasks are assigned first as required by the ordering in  $\leq$ . Each robot subgroup assigned to a subtask is evaluated by the feasibility and the estimated completion time, which is described in the subsequent module. The nodes are prioritized based on their time-average efficiency. The search is terminated after all  $H$  subtasks are assigned. Thus, the node with the maximum efficiency is selected as the partial plan  $\bar{\tau}$ , by which the subtasks are executed. Once the replanning condition holds such as a fixed number of subtasks are fulfilled or certain subtasks are delayed, the set of subtasks  $\bar{\mathcal{S}}$  are updated and re-assigned.

More details about the dynamic task assignment can be found in the supplementary materials. Moreover, the above procedure for task decomposition and partial ordering have the following guarantees for completeness and correctness, of which the proofs are provided in the Appendix B.

**Lemma 1.** Alg. 1 terminates in finite steps and generates a strict partial ordering. ■

**Lemma 2.** Given the partially-ordered segments  $(\bar{\mathcal{S}}, \leq)$ , each object  $m \in \mathcal{M}$  can reach its goal state without collision, by traversing its sequence of segments  $\bar{\mathcal{S}}_m$  as follows: (I) each segment  $\bar{\mathcal{S}}_m^k \in \bar{\mathcal{S}}_m$  takes a bounded time to traverse; (II) if  $\mathcal{S}_{m_1}^{k_1} \leq \mathcal{S}_{m_2}^{k_2}$  holds, object  $m_2$  can be moved from the initial state  $s_{m_2}(t_{m_2}^{k_2, s})$  of  $\mathcal{S}_{m_2}^{k_2}$  only after object  $m_1$  has reached the ending state  $s_{m_1}(t_{m_1}^{k_1, e})$  of  $\mathcal{S}_{m_1}^{k_1}$ . ■

### B. Accelerated Hybrid Optimization for Collaborative Push

Given the local task plans  $\bar{\tau} = \{\tau_i^*\}$  from the previous section, a subgroup of robots  $\mathcal{N}_m$  is assigned to push object  $\Omega_m$  along the path segment  $\mathcal{S}_m^k$ . This section describes how to efficiently generate feasible hybrid plans for the subgroup as a sequence of pushing modes, forces and the reference trajectory, in a learning-while-planning manner.

**Problem 3.** Given  $\mathcal{S}_m^k$  and  $\mathcal{N}_m^k$  for object  $\Omega_m$ , determine the hybrid plan  $\{(\xi(t), \mathbf{S}_{\mathcal{N}_m^k}(t), \mathbf{s}_m(t))\}$  such that the local cost function in (1) is minimized. ■

**1) Keyframe-guided Hybrid Search:** A keyframe-guided hybrid search (KGHS) is proposed to solve Problem 3 as shown in Fig. 5. Instead of optimizing an entire segment at once, the algorithm recursively splits it into short *arc segments*, each of which is executable under a *single* pushing mode. A *keyframe* is an intermediate object state at which the contact mode is allowed to switch. Let  $s_\ell$  denote the  $\ell$ -th keyframe along the segment. Namely, the hybrid plan becomes a sequence of keyframes and the pushing modes between them, denoted by  $\vartheta \triangleq \kappa_0 \cdots \kappa_\ell \cdots \kappa_{L_\vartheta}$ , with the  $\ell$ -th stage  $\kappa_\ell \triangleq (s_\ell, \xi_\ell)$  being the keyframe  $s_\ell$  and the mode  $\xi_\ell = (\xi_\ell, \mathbf{F}_\ell)$ ,  $\forall \ell = 1, \dots, L_\vartheta$ . Note that  $s_0 = s_m^0$  and  $s_{L_\vartheta} = s_m^G$ , while  $\xi_\ell$  is the pushing mode for the arc segment  $\varrho_\ell \triangleq \overbrace{s_\ell s_{\ell+1}}$ . The search space, denoted by  $\Theta \triangleq \{\vartheta\}$ , is thus significantly reduced.

Starting from  $\vartheta_0 \triangleq (s_m^0, \emptyset)(s_m^G, \emptyset)$ , the search space  $\Theta$  is explored via iteratively selecting the promising node and adding keyframes as needed. To begin with, the node  $\vartheta^*$  with the lowest estimated cost is selected, i.e.,

$$J_{\text{hyb}}^m(\vartheta) \triangleq \sum_{l=0}^{L_\vartheta-1} (J_{\text{MF}}^m(\xi_\ell, \mathbf{P}_{\varrho_\ell}^B) + w_s J_{\text{sw}}^m(\xi_\ell, \xi_{\ell+1})) + w_n J_{\text{nv}}^m(s_\ell, s_{\ell+1}), \quad (6)$$

where  $J_{\text{MF}}^m(\cdot)$  is a multi-directional estimation of the feasibility of employing mode  $\xi_\ell$  for the body-frame velocity  $\mathbf{p}_{\varrho_\ell}^B$  corresponding to the arc  $\varrho_\ell$ , with more detailed derivations provided in Appendix A3;  $J_{\text{sw}}^m(\cdot)$  measures the switching time from mode  $\xi_\ell$  to mode  $\xi_{\ell+1}$ ;  $J_{\text{nv}}^m(\cdot)$  measures the navigation time cost along the arc  $\varrho_\ell$ ; and  $w_s, w_n > 0$  are the weighting parameters. These cost terms are designed to reflect the objective function in Problem 1. Unlike a hard feasibility constraint [28], the soft measure  $J_{\text{MF}}^m(\cdot)$  keeps the hybrid search *solvable and robust* under modeling uncertainty or when the constraint is structurally unsatisfiable (e.g., single-contact pushing), while still permitting explicit enforcement of feasibility, as detailed in Appendix A3. Afterwards, the selected node  $\vartheta^*$  is expanded by finding the first keyframe that has not been assigned with a mode, e.g.,  $(s_\ell, \emptyset)$  for the segment  $\varrho_\ell$ . Then, the node expansion is performed in four steps:

(I) If the arc  $\varrho_\ell$  intersects with an obstacle, a new keyframe  $(\hat{s}_\ell, \emptyset)$  is inserted between  $s_\ell$  and  $s_{\ell+1}$  by selecting an intermediate collision-free state  $\hat{s}_\ell$ , i.e.,  $\vartheta^+ \triangleq$

**Algorithm 2:** Neural Accelerated Hybrid Optimization: HybDIF( $\cdot$ )

---

**Input:** Subgroup  $\mathcal{N}_m^k$ , subtask  $\mathfrak{S}_m^k$ , Library  $\mathcal{X}$   
**Output:** Hybrid Plan  $\vartheta_m^{k,*}$ , control  $\mathbf{u}_{\mathcal{N}_m^k}$ .

```

/* Offline Generation */ 
1 Collect state pairs  $\{(\mathbf{s}_s, \mathbf{s}_e)\}$  along the segment  $\mathfrak{S}_m^k$ ;
2 Batch-generate hybrid plans for all state pairs:
    $\{\hat{\vartheta}^*\} \leftarrow \text{DIF}(\Omega_m, \mathcal{N}_m^k, \{(\mathbf{s}_s, \mathbf{s}_e)\})$ ;
3 Update library  $\mathcal{X} \leftarrow \mathcal{X} \cup \{\hat{\vartheta}^*\}$ ;
/* Keyframe-guided Hybrid Search */ 
4 Initialize  $\Theta = \{\vartheta_0\}$ ;
5 while not terminated do
6   Select  $\vartheta^*$  by (6);
7   Find first keyframe  $(\mathbf{s}_\ell, \emptyset) \in \vartheta^*$  without a mode;
8   if Arc  $\varrho_\ell$  has collision then
9     |  $\hat{\vartheta}^* \leftarrow (\mathbf{s}_\ell, \emptyset)(\hat{\mathbf{s}}_\ell, \emptyset), \hat{\mathbf{s}}_\ell \in \mathfrak{S}_m^k$ ;
10  else
11    /* Sequentially run the following steps
       until  $\hat{\vartheta}^*$  becomes feasible */ 
12    |  $\hat{\vartheta}^* \leftarrow (\hat{\kappa}_1^*, \dots, \hat{\kappa}_h^*) = \mathcal{X}(\Omega_m, \mathcal{N}_m^k, \mathbf{s}_\ell, \mathbf{s}_{\ell+1})$ ;
13    |  $\hat{\vartheta}^* \leftarrow \text{DIF}(\Omega_m, \mathcal{N}_m^k, \mathbf{s}_\ell, \mathbf{s}_{\ell+1})$ ;
14    |  $\hat{\vartheta}^* \leftarrow \text{IterSamp}(\vartheta^*, \varrho_\ell)$ ;
15    |  $\hat{\vartheta}^* \leftarrow (\mathbf{s}_\ell, \emptyset)(\hat{\mathbf{s}}_\ell, \emptyset), \hat{\mathbf{s}}_\ell \in \mathfrak{S}_m^k$ ;
16    | if  $\|\varrho_\ell\| < \epsilon$  then
17      |   |  $\hat{\vartheta}^* \leftarrow \text{SeqArcApprox}(\varrho_\ell, \mathcal{N}_m^k)$ ;
18  Expand  $\vartheta^*$  by replace  $(\mathbf{s}_\ell, \emptyset)$  with  $\hat{\vartheta}^*$ ;

```

---

$\dots \kappa_\ell(\hat{\mathbf{s}}_\ell, \emptyset) \kappa_{\ell+1} \dots$ ; (II) If the arc  $\varrho_\ell$  is collision-free, a mode is generated for the segment  $\varrho_\ell$  by minimizing the loss function  $J_{\text{MF}}^m(\cdot)$  in (6) to measure feasibility, i.e.,

$$\xi_\ell^* = \underset{\xi_\ell \in \Xi_m}{\operatorname{argmin}} \{J_{\text{MF}}^m(\xi_\ell, \mathbf{p}_{\varrho_\ell}^B)\}, \quad (7)$$

which results in a combinatorial and nonlinear optimization problem. To solve (7), we employ a parallel multi-start search over contact modes. Candidate modes are initialized by random sampling and iteratively refined by adjusting contact points. For each candidate  $\xi_\ell$ , we first check *force* feasibility via a loss related to force balance, and then *practical* feasibility by simulating the pushing process along  $\varrho_\ell$ . A mode is accepted only if both criteria are satisfied; see Appendix A3 for details. Thus, if a feasible mode is found, the mode is assigned to the segment  $\varrho_\ell$ .

(III) If no feasible mode is found between  $\mathbf{s}_\ell$  and  $\mathbf{s}_{\ell+1}$ , an iterative sampling procedure is proposed to generate a sequence of keyframes and modes that further split the segment  $\varrho_\ell$  into  $h > 0$  sub-segments, i.e.,

$$\hat{\vartheta}^* \triangleq (\hat{\kappa}_1^*, \dots, \hat{\kappa}_h^*) \triangleq \text{IterSamp}(\mathbf{s}_\ell, \mathbf{s}_{\ell+1}). \quad (8)$$

Keyframes are initially sampled uniformly along  $\varrho_\ell$  and then perturbed with diminishing variance; for each perturbed segment, modes are generated as in (II) and evaluated using (6). The procedure terminates when all sub-segments admit practically feasible modes or a maximum iteration count is reached.

(IV) Lastly, if the subplan  $\hat{\vartheta}^*$  is still infeasible, a new keyframe  $(\hat{\mathbf{s}}_\ell, \emptyset)$  is inserted between  $\mathbf{s}_\ell$  and  $\mathbf{s}_{\ell+1}$ , thereby

splitting  $\varrho_\ell$  into two shorter segments. When the arc length  $\|\varrho_\ell\|$  is below a threshold  $\epsilon > 0$ , we approximate the original arc by concatenating pre-validated modes,

$$\hat{\vartheta}^* \triangleq \text{SeqArcApprox}(\varrho_\ell, \mathcal{N}_m^k), \quad (9)$$

which serves as a “last resort” hybrid plan for the subgroup  $\mathcal{N}_m^k$  due to its frequent mode switching. The following lemma summarizes the theoretical guarantee of the hybrid search; the proof is given in Appendix B.

**Lemma 3.** Given that the subgroup  $\mathcal{N}_m^k$  is mode-sufficient as defined in Appendix A4, and the path  $\mathfrak{S}_m^k$  is collision-free, the proposed hybrid search scheme finds a feasible hybrid plan  $\vartheta^*$  in finite steps.

2) *Diffusion-based Neural Acceleration:* A key aspect of the hybrid search scheme above is to generate suitable keyframes  $\{\kappa_\ell\}$  along the path segments and the associated pushing modes  $\{\xi_\ell\}$ . To further improve the planning efficiency and reduce planning time mainly for large-scale scenarios, a diffusion-based neural accelerator is proposed to generate plausible keyframes and pushing modes for the hybrid search scheme. Consequently, high-quality pushing strategies are found earlier for any given path segment and even unseen objects, in both free and cluttered environments. In particular, it consists of three main components as described below: the generation of training data, the training of proposed network architecture, and the deployment of the learned network.

**Dataset Generation.** As shown in Fig. 6, a large variety of pushing problems are instantiated for randomly deformed basic shapes, a random number of robots with random configurations, and random initial and target states. Then, the proposed keyframe-guided hybrid search scheme is used to solve each problem instance, yielding multiple candidate solutions. Instead of terminating upon finding a feasible solution, a fixed number of iterations is enforced to obtain better solutions. Each candidate is evaluated in simulation, and the one with minimum task duration is selected. For each selected solution, we store the problem description and its optimal hybrid plan:

$$\mathfrak{D} \triangleq \left\{ ((\mathbf{s}_m^0, \mathbf{s}_m^G, \Omega_m, \{\mathbf{x}_n^0\}), (\vartheta_m^*, T_m^*)) \right\}. \quad (10)$$

To improve data efficiency, we also exploit two rotational symmetries. We randomly rotate the object’s body frame by an angle  $\theta$ , expressing all contact points in the rotated frame and shifting the object orientation by  $-\theta$  (a pure change of reference), and use global in-plane rotational invariance to normalize each trajectory to a common zero terminal state, thereby reducing the output degrees of freedom and letting the model predict in this canonical frame.

**Network Architecture and Training.** The noise prediction network follows the U-net architecture in [33], with action generation conditioned on observations  $\mathbf{O}$  via Feature-wise Linear Modulation (FiLM) [39]. The dataset  $\mathfrak{D}$  in (10) is distilled into observational inputs  $\mathbf{O}$  and outputs  $\mathbf{A}$ . The observations  $\mathbf{O}$  include: (I) object attributes, such as coordinates and normal vectors of a fixed number of candidate contact points and the maximum friction force and torque; (II) robot attributes, including size and maximum pushing forces; and

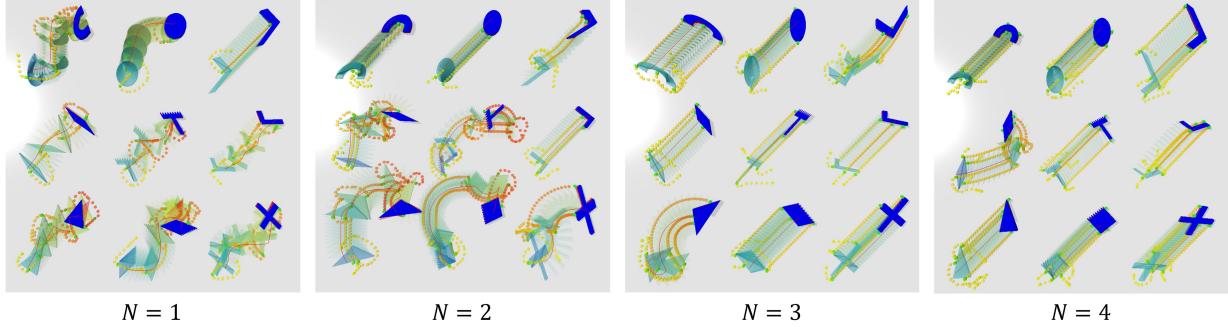


Fig. 6. Dataset generation for training the diffusion-based generator of primitive pushing modes, where the initial state of the objects are randomized (in light green), the object shape is scaled and twisted, and the number of robots is varied from 1 to 4. The resulting robot trajectories are shown in red lines.

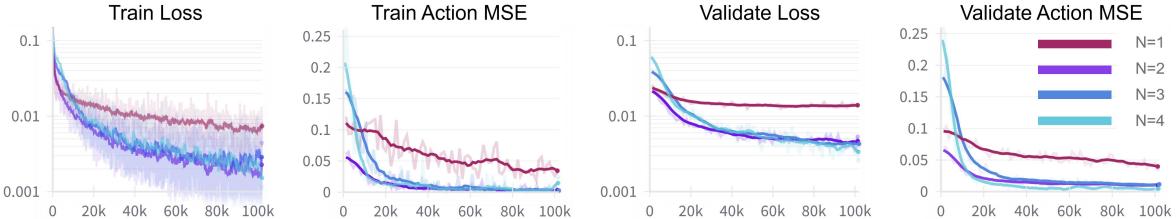


Fig. 7. The training loss, the validation loss, the MSE of the hybrid plan during training and validation, for the pushing tasks shown in Fig. 6.

(III) the initial and target object states  $\mathbf{s}^0$  and  $\mathbf{s}^G$ . The outputs  $\mathbf{A}$  are sequences of keyframes and pushing modes,

$$(\mathbf{s}_1, \xi_1) \cdots (\mathbf{s}_h, \xi_h) \leftarrow \text{DIF}(\Omega_m, \mathcal{N}_m^k, \mathbf{s}^0, \mathbf{s}^G), \quad (11)$$

where  $\text{DIF}(\cdot)$  is the diffusion-based predictor.

The network is trained following the standard Denoising Diffusion Probabilistic Model (DDPM) procedure [39]. At a randomly sampled diffusion step  $k$ , Gaussian noise  $\epsilon^k$  is added to each action  $\mathbf{a} \in \mathbf{A}$ , and the network is trained to predict  $\epsilon^k$  from  $(\mathbf{o}, \mathbf{a} + \epsilon^k, k)$  using the MSE loss

$$\mathcal{L}_{\theta} \triangleq \text{MSE}(\epsilon^k, \epsilon_{\theta}(\mathbf{o}, \mathbf{a} + \epsilon^k, k)),$$

where  $\theta$  denotes the network parameters. As shown in Fig. 7, the training and validation losses, together with the MSE of the predicted hybrid plans, decrease steadily over epochs. Fig. 8 further illustrates hybrid plans generated by the trained model for different objects and initial poses, where the keyframes and modes are progressively refined during denoising and align well with the ground-truth plans in the dataset.

**Deployment within Hybrid Search.** The most straightforward way to deploy the learned neural network is to replace the hybrid search and directly execute its predicted plans, or to replace the optimization-based mode generation in (7). However, this often leads to collisions and failures when test scenarios differ from the training data. Instead, we use the diffusion model to propose *initial* hybrid plans, which are then verified and, if necessary, locally refined by the model-based scheme. Specifically, when the arc between  $\mathbf{s}_{\ell}$  and  $\mathbf{s}_{\ell+1}$  is collision-free, the diffusion model generates candidate keyframes and pushing modes connecting these states. The best candidate is inserted into the search tree and then checked for feasibility and performance. To further accelerate planning, we maintain an online library  $\mathcal{X}$  of previously verified pushing strategies, indexed by relative object displacements. The library is queried first; if no suitable entry is found, the hybrid search with the diffusion model is invoked to generate a new verified strategy,

which is then stored in  $\mathcal{X}$ . This “learning while planning” setup both speeds up solving a single problem and improves performance over multiple tasks.

**Remark 3.** Unlike approaches that directly execute neural plans [15], [33], the predicted keyframes and modes are only prioritized within the hybrid search and are always *verified*, which is crucial in unseen complex workspaces. ■

### C. Online Execution and Adaptation

Given the high-level assignment algorithm of the pushing tasks and the hybrid optimization of the pushing policy, the robotic fleet can start executing the pushing tasks as follows.

1) *Mode Execution:* Given the hybrid plan  $\vartheta_m^{k,*}$  for each object  $m \in \mathcal{M}$  and subtask  $\mathfrak{S}_m^k$ , the robots need execute the plan by tracking the desired trajectories. More specifically, a local reference trajectory is generated as the arc segment  $\varrho(t)$  that connects the current state  $\mathbf{s}_m(t)$  of the object and the next keyframe state  $\mathbf{s}_m^*$  along the plan, which is updated at each control loop. Then, the next reference state  $\hat{\mathbf{s}}_m^r$  is selected as the first state along the planned arc  $\varrho(t)$  satisfying  $|\hat{\mathbf{s}}_m^r - \mathbf{s}_m(t)| > \delta_c$ , where  $\delta_c > 0$  is a fixed threshold. Given this reference object state, the corresponding contact point and orientation for each robot  $n \in \mathcal{N}_m^k$  as specified by the hybrid plan are denoted by  $\hat{\mathbf{c}}_n$  and  $\hat{\psi}_n$ , respectively, yielding the desired robot state  $\hat{\mathbf{s}}_n^r \triangleq (\hat{\mathbf{c}}_n, \hat{\psi}_n)$ . Then, the desired translational and rotational velocities are given by:

$$\hat{\mathbf{v}}_n = K_{\text{vel}} (\hat{\mathbf{c}}_n - \mathbf{c}_n), \quad \hat{\omega}_n = K_{\text{rot}} (\hat{\psi}_n - \psi_n); \quad (12)$$

where  $K_{\text{vel}}$  and  $K_{\text{rot}}$  are positive gains. These velocity commands are then passed to the low-level controllers, yielding actuation forces  $\mathbf{Q}_{\text{drv}}$  for robot  $R_n$ . The exact implementation of the low-level controllers depends on the actuation hardware. Moreover, to handle contact slippage and deformation, pushing is paused when any contact point deviates beyond a threshold, triggering load-free motion to re-establish valid contact.

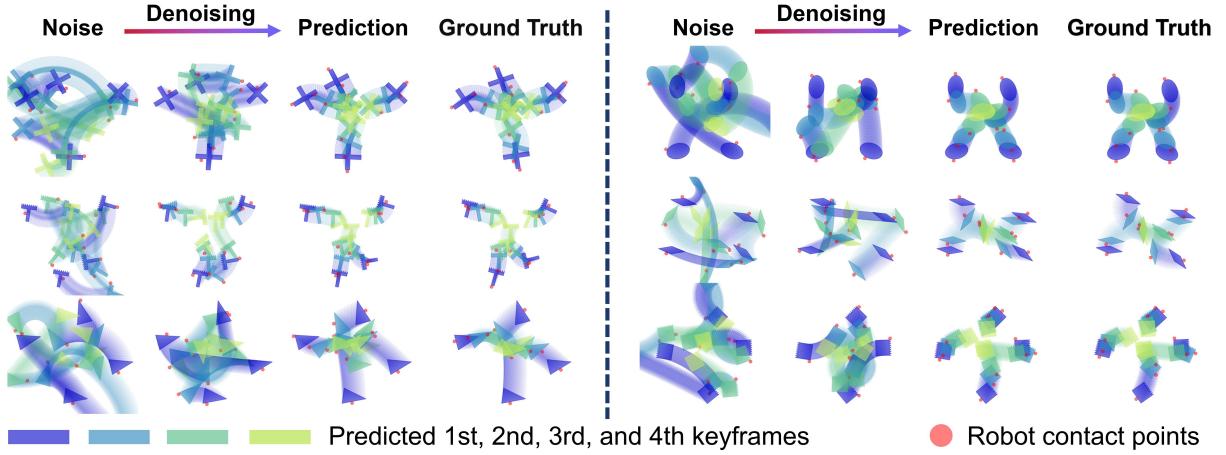


Fig. 8. Illustration of generating hybrid plans with a horizon of 3 via the trained diffusion model, for 6 different objects starting from 4 distinctive initial positions and a common target at the origin. Each predicted hybrid plan consists of 4 keyframes and 3 pushing modes in between. During the denoising process, both the mode and keyframe quality improve progressively, with the final predictions align well with the ground truth in the dataset.

Lastly, for high precision tracking at centimeter level, a more deliberate control scheme might be required, e.g., distributed coordination and control [28], [29]. Note that  $\mathbf{u}_n$  in (12) updates in real-time, relying solely on the online computation of the analytical arc segment  $q_t$  at each time step.

**Remark 4.** As neither the simulated nor physical robots in our work are equipped with force sensors, the kinematic control in (12) are adopted instead of explicit force-based control. Thus, contact forces emerge implicitly from physical interactions, reducing the reliance on accurate force measurements. More analyses on the feasibility of this kinematic control scheme can be found in Appendix A3. ■

**Remark 5.** Although the controller above is designed for holonomic models, extensions to non-holonomic robots such as quadruped and differential-drive robots can be achieved by local adjusting schemes, e.g., turn-and-forward. More numerical examples are given in Sec. V-A3. ■

2) *Online Adaptation upon Failures:* As discussed earlier, due to model uncertainties such as communication delays, actuation noises and slipping, the collaborative pushing task is inherently uncertain in terms of both object trajectory and task duration. In addition, congestion or deadlock during navigation can also lead to failed execution. Thus, online adaptation to such contingencies is essential for both the task assignment and the hybrid search of the pushing strategy. More specifically, the execution of the  $\ell$ -th segment of the current hybrid plan  $\vartheta_m^{k,*}[\ell] = (\mathbf{s}_\ell, \xi_\ell)$  is considered to be *failed* if one of the following conditions holds:

$$\begin{aligned} \text{dist}(\mathbf{s}_m(t'), \varrho_\ell) &\geq \delta_f, \exists t' \in [t, t - T_c]; \\ \|\mathbf{s}_m(t') - \mathbf{s}_m(t'')\| &< r_{\text{stuck}}, \forall t', t'' \in [t, t - T_c]; \end{aligned} \quad (13)$$

where  $\text{dist}(\cdot)$  measures the minimum distance from a point to a 2D curve; the first condition checks whether the object deviates from the target arc  $\varrho_\ell$  by more than  $\delta_f > 0$  over a recent time window  $T_c > 0$ ; the second detects if the object is stuck, i.e., its position change remains below a threshold  $r_{\text{stuck}} > 0$  during the same period. Note that  $\delta_f$  bounds the tracking error, while setting  $\delta_f$  too small would trigger frequent replanning and reduce efficiency. Moreover, if the object is too close

TABLE I  
CONDITIONS FOR COMPLETENESS

Cond.	Description
C1	The geometric and physical properties of the objects and robots are known in advance.
C2	Inflating each target object by $(1 + \epsilon_r)$ times the maximum robot diameter, for arbitrarily small $\epsilon_r$ , yields a feasible multi-agent path finding (MAPF) instance.
C3	For every target object $\Omega_m$ , there exists a mode-sufficient robot subgroup $\mathcal{N}_m \subseteq \mathcal{N}$ (see Appendix A4).

to obstacles, or new obstacles appear, or several robots fail, the hybrid search algorithm in Alg. 2 is re-executed with the current system state and functional robots, yielding an adapted hybrid plan  $\vartheta_m^{k,*}$ , and the control policy in (12) is reactivated. Last but not least, in case these measures can still not resolve the failure, the high-level task assignment  $\bar{\tau}$  is updated by receding horizon planning with the current system state and remaining subtasks. This can be effective when several robots failed and the remaining robots in the subgroup are not sufficient to push the object. The above procedures of online execution and adaptation are summarized in Fig. 9.

#### D. Discussion

1) *Computation Complexity:* The task-level complexity is dominated by MAPF, task decomposition, and task assignment. For  $M$  polygonal objects, a complete MAPF algorithm such as CBS [40] is exponential, so we adopt a sequential A\* scheme with complexity  $\mathcal{O}(\|V_{\text{st}}\| \log \|V_{\text{st}}\| \cdot M^2)$ , where  $\|V_{\text{st}}\|$  is the size of the space-time graph. Task decomposition and assignment cost  $\mathcal{O}(M^2)$  and  $\mathcal{O}(M^H N^{\min(M,H)})$ , respectively. For the hybrid search in Alg. 2, the cost is mainly determined by mode generation and the search-tree size. The feasibility check via  $J_{\text{MF}}$  in (20) corresponds to a linear program with complexity  $\mathcal{O}(N^{3.5})$  [41]. Overall, the time complexity of the hybrid search is  $\mathcal{O}(N^{3.5} W_k^{2\alpha/\alpha_{\min}})$ , where  $W_k$  is the maximum number of nodes generated in a single expansion.

Lastly, as summarized in Table I, the key assumptions include that: the intrinsic properties of robots and objects are known, a feasible MAPF solution exists for all objects, and there is always a mode-sufficient subgroup of robots for each

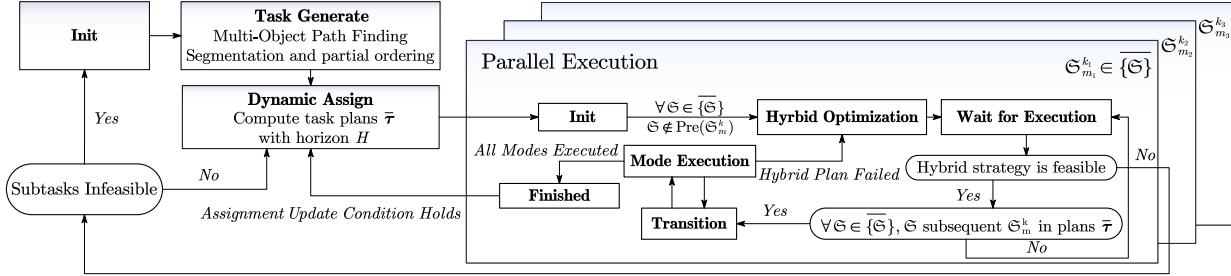


Fig. 9. The proposed scheme of online execution and adaptation to uncertainties, delays and failures, as detailed in Sec. IV-C. Note that the pushing subtasks of different objects are executed in parallel. Adaptation is triggered first within the subgroup and then the whole group.

object. Then, the completeness guarantee of the overall scheme is stated below, with detailed proofs in Appendix B.

**Theorem 1.** *Under the conditions in Table I, the proposed hybrid optimization scheme yields valid solutions to Problem 1.*

2) *Generalization:* The proposed framework can be generalized in the following notable directions:

(I) **Movable obstacles.** Movable obstacles can be treated as additional objects that are pushed in the same way as targets, either to unblock infeasible scenarios or to shorten overall task duration. They are included in the MAPF instance without fixed goal positions, so that timed paths implicitly specify where and how they should be moved, and the resulting subtasks include pushing these movable obstacles. (II) **Heterogeneous robots.** When robots have different capabilities, the high-level assignment should account for their efficiency: heavier objects are allocated to more powerful robots, while lighter ones are handled by smaller teams. The interaction mode can be extended to  $\xi \triangleq (\mathbf{c}_1, \mathbf{f}_1, R_1) \cdots (\mathbf{c}_N, \mathbf{f}_N, R_N)$ , so that contact points  $\mathbf{c}_n$  and forces  $\mathbf{f}_n$  reflect individual limits, and the transition cost  $J_{sw}(\cdot)$  in (6) is adapted accordingly.

(III) **6D pushing tasks.** For 6D object poses in 3D scenes, the framework extends under a quasi-static, zero-gravity setting. The 2D arc transition becomes a 3D spiral transition, and the hybrid search space is lifted to 6D pose, while the overall structure of the pipeline remains unchanged. (IV) **Planar assembly.** When only a global assembly pattern is specified, goal poses for individual objects can be generated by geometric solvers or generative methods [42]. The resulting object goals are then passed to our pipeline to complete the assembly.

3) *Limitation:* The proposed approach has several limitations. (I) As discussed in Appendix A1, the quasi-static analysis of collaborative pushing modes holds only for slow motions with small inertia and persistent contact. For microgravity or floating objects, our method can still operate by actively decelerating, but this can be inefficient and calls for further study. (II) The mass, shape, and friction coefficients of all objects are assumed known for feasibility evaluation. Small deviations can be absorbed by the tracking controller in (12), whereas large uncertainties would require active identification and multi-objective planning. (III) The task decomposition relies on a centralized MAPF solver, and the hybrid search is executed centrally for each subteam. For large-scale scenarios, decentralized MAPF approaches [43] could reduce complexity, and designing a fully decentralized pushing scheme remains an interesting direction for future work.

## V. NUMERICAL EXPERIMENTS

To further validate the proposed method, extensive numerical simulations and hardware experiments are presented in this section. The proposed method is implemented in Python3 and tested on a laptop with an Intel Core i7-1280P CPU. Numerical simulations are run in PyBullet [44], while ROS is adopted for the hardware experiments. The GJK package from [45] is used for collision checking, and the CVXOPT package from [46] for solving linear programs. Simulation and experiment videos are available in the supplementary material.

### A. Numerical Simulations

1) *Setup of Simulation Environments:* All simulations are conducted in PyBullet with a fixed integration time step of  $\Delta t = \frac{1}{240}$  s. Unless otherwise specified, the target object has a mass of 10kg, a ground friction coefficient of 0.8, and a side friction coefficient of 0.2. Robots are homogeneous cylinders of 0.3m diameter and have a maximum pushing force of 100N. A task is considered complete when the object is within 0.05m and 0.15rad of the goal pose.

**Control Scheme.** As discussed in Sec. IV-C1, each robot is controlled by a two-level scheme. The desired velocity to track the arc segments is given by  $\mathbf{u}_n = (\hat{\mathbf{v}}_n, \hat{\omega}_n)$  as in (12), with gains  $K_{vel} = 5$  and  $K_{rot} = 1$ . A low-level proportional controller then generates the forces  $\mathbf{f}_{n,drv} = 400(\hat{\mathbf{v}}_n - \mathbf{v}_n)$  and the torques  $\chi_{n,drv} = 20(\hat{\omega}_n - \omega_n)$ , which are clipped by the actuation limits and applied through the force/torque API of PyBullet. The default robot-ground friction is disabled, so robots move only under the commanded and clipped forces/torques. Instead of explicitly modeling the contact dynamics of Mecanum wheels [47], this formulation serves as an abstraction of the ideal omni-directional platform, while the actuation limits bound the realizable pushing forces.

**Physics Interaction.** PyBullet resolves contacts via a sequential-impulse scheme that approximates a Mixed Linear Complementary Problem (MLCP) under the non-penetration and Coulomb friction constraints. However, its lateral and spinning friction are modeled independently, which is inconsistent with the coupled limit-surface friction model in the proposed mode-generation module, and may under-represent the coupled nature of real-world contact. To mitigate this, the object-ground friction of PyBullet is replaced by the ellipsoidal limit-surface model in the force-moment space [2], with more implementation details in Appendix A2. A sensitivity study towards different friction models is provided in

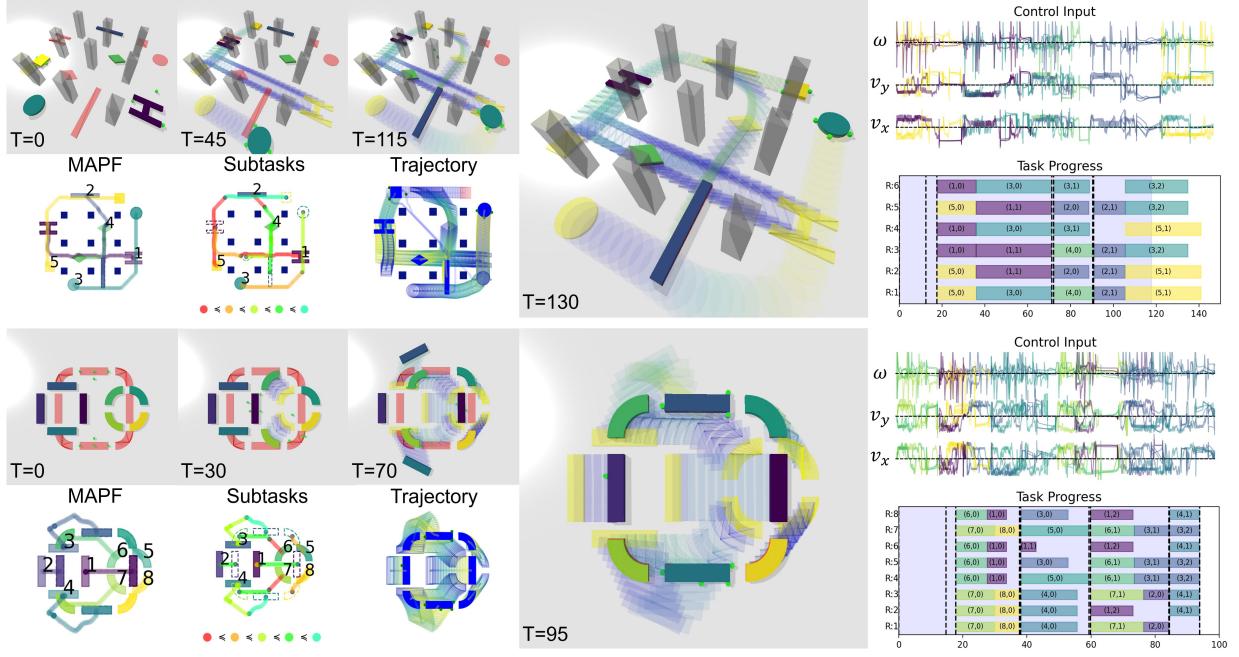


Fig. 10. Simulation results of the proposed PushingBot system in two scenarios, including the timed path  $\{\mathfrak{S}_m\}$ , generated subtasks with strict partial order (numbered and in different colors), hybrid plans and final trajectories  $S^*$ , robot control inputs, and the task overall assignment results (indexed by the object ID and order of segments). **Top:** 5 objects (diamond, circular, triangular, square and H-shape), 9 subtasks, and 8 robots within a forest-like workspace; **Bottom:** 8 objects (4 rectangular and 4 quarter-circle), 13 subtasks and 8 robots.

TABLE II  
SUMMARY OF RESULTS FOR NOMINAL SCENARIOS

Metric	Scen. I	Scen. II	Scen. III
Objects/Robots	4/6	5/6	8/8
MAPF Time (s)	9.6	12.2	15.3
Subtasks Generated	9	9	13
Longest Task Dependency	5	5	5
Task Assignment Time (s)	2.0	2.1	2.5
Hybrid Optimization Time (s)	0.4	0.4	0.4
Pushing Modes	12	20	18
Completion Time (s)	70	130	95
Total Planning Time (s)	16.5	23.5	28.3

the supplementary material, including the relative ranking of all methods and the main conclusions remain unchanged.

**Algorithm configuration.** The planning horizon for the dynamic task assignment is set to 4. Assignments are re-evaluated once the first two tasks are completed or after 80s. The tracking threshold  $\delta_c$  is set to 0.1m. The desired velocity is updated at 60Hz while low-level controllers are executed at 240Hz, synchronized with the simulation step. More details are provided in the supplementary material.

**2) Nominal Scenarios:** As shown in Fig. 2 and 10, three distinctive scenarios are tested: (I) 4 objects (T-shape, cylinder, semi-circular ring, desk-like) are pushed by 6 robots; (II) 5 objects (circular, triangular, square, H-shape, and diamond) are pushed through passages between dense prismatic obstacles by 6 robots to their goals; (III) 8 objects (4 rectangular bars and 4 quarter-circle segments) are rearranged from a square and circular ring to a rounded square by 8 robots. The proposed framework successfully completes all pushing tasks in all three scenarios, with the planning time of each component summarized in Table II. The solution time for MAPF increases with the number of objects, while the longest dependency chain remains constant at 5 across three scenarios as primarily

determined by path overlap. The time for the task assignment remains consistent, due to the receding-horizon scheme. Moreover, the planning time for the hybrid optimization per subtask across all objects takes around 0.4s, which is decoupled from overall task complexity due to the neural accelerated scheme. The task completion time is collectively influenced by total path length, the task dependency, and the object intrinsics.

**3) Generalization:** As discussed in Sec. IV-D2 several notable generalizations of the proposed method are evaluated.

**(I) Movable obstacles.** As shown in Fig. 11, 3 target objects are pushed by 6 robots in a  $20m \times 20m$  workspace with fixed (dark grey) and movable (light grey) obstacles. If movable obstacles are treated as fixed (in the top row of the figure), the objects must take inefficient detours and narrow passages, yielding longer MAPF planning times, more complex trajectories, more frequent mode switching, and increased task duration. In contrast, treating the movable obstacles as pushable objects without predefined goals allows our framework to push these obstacles into the nearest freespaces, creating wider passages and shortcuts for the target objects. This generalization reduces the task duration from 83.2s to 63.3s by 23%, while retaining safety and liveness.

**(II) Heterogeneous robots.** A scenario of pushing race is designed to demonstrate the capability of the proposed method to handle heterogeneous robots and objects. As shown in Fig. 14, the setup involves two T-shape objects and two triangular objects. For each shape, one object has a mass of 5kg and the other has a mass of 20kg. These objects are placed in a row at one end of a  $20m \times 20m$  workspace, with the target position at the opposite end. In total, 6 heterogeneous robots are deployed, i.e., the maximum pushing forces  $f_{n,\max}$  in (14) are 100N, 30N and 10N and each force level is associated with two robots. The assignment and planning strategy as

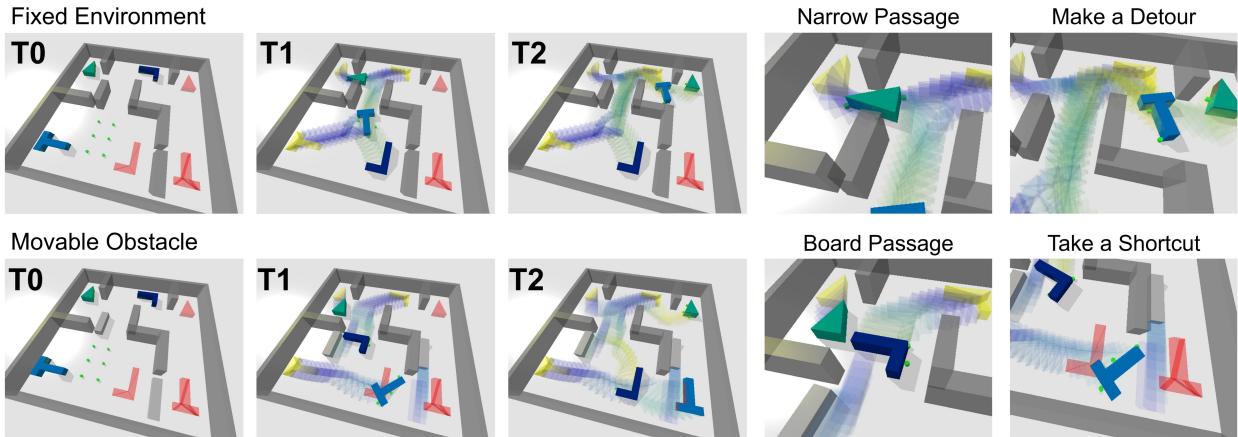


Fig. 11. Generalization of the proposed method to scenarios with movable obstacles: all obstacles are treated as fixed (**Top**); movable obstacles, shown in lighter gray, are pushed to create wider passage for both triangular and L-shape objects and provide a shortcut for the T-shape object. (**Bottom**).

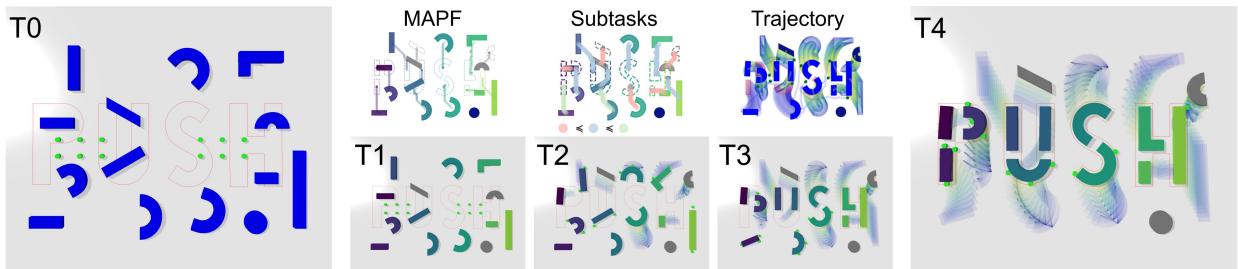


Fig. 12. Generalization of the proposed method to planar assembly with 14 objects and 12 robots. Note that 11 objects are selected and pushed to suitable target positions, while the other objects (in grey) are treated as movable obstacles.

described in Sec. IV-D2 is adopted, by which all tasks are completed with different duration but all below 40s. Compared to the nominal method which neglects these heterogeneity, the assignments and pushing modes are intuitive, i.e., (i) heavier objects are always pushed by more and stronger robots, while lighter objects are handled by the remaining robots. (ii) robots with larger forces are always assigned to the contact point that requires more force. Although the task can still be completed via the nominal method, the task duration is much longer (by minimum 23% and maximum 188%). Last but not least, as shown in Fig. 15, quadruped and differential-drive robots are adopted for the collaborative pushing task. We adopt a soft “turn-and-forward” scheme that smoothly scales linear speed by the heading error, which suffices for our task settings. For higher-precision pushing with nonholonomic platforms (e.g., centimeter-level path tracking), a more foundational treatment such as Jourdain’s principle [29] may be required.

**(III) Planar Assembly.** As shown in Fig. 12, a planar assembly task of assembling the word “PUSH” via 14 objects and 12 robots is demonstrated. The desired layout is highlighted in red at the center of a  $26m \times 20m$  area, with objects initially randomly placed. Heuristic programming method in [48] is adopted to select objects and optimize their goal positions. For instance, two identical semicircular ring-like objects are chosen to form the letters “S”, while L-shaped objects can be part of the letter “H”, and so on. Objects that are not required in the assembly, such as parallelograms or cylinders, are treated as movable obstacles. The selected objects and their timed paths are shown in Fig. 12. Once this procedure is completed, the proposed scheme is applied to

push the selected objects to their goal positions. The whole assembly is completed at  $t = 108s$ , during which 13 subtasks and 19 pushing modes and 6 mode switches are performed.

**(IV) 6D Pushing.** As shown in Fig. 13, the generalization to pushing tasks in complex 3D workspace is demonstrated. The scenario consists of 8 objects and 12 robots within a workspace of  $10m \times 10m \times 10m$ . Initially, all objects are placed at one vertex of a cube and should be pushed to another vertex, i.e., to swap their positions. The resistance coefficient of all objects are set to  $K = 100$ . Thus, the MAPF algorithm in 3D is first employed to generate 8 timed paths  $\{\mathcal{G}_m\}$  within 2.3s, which are then decomposed into 13 subtasks. All tasks complete at  $t = 165s$  with a total planning time of 15.2s and 21 pushing modes. The difficulty of stably pushing an object in 3D space depends on the diversity of its surface normal vectors. For instance, the triangular prism has only 5 faces, while the sphere has arbitrary normal vectors. Consequently, the proposed method assigns 6 robots to the triangular prism, but only 3 robots to the sphere.

**4) Comparison of Diffusion and Iterative Sampling:** We compare the proposed diffusion-based method with the iterative sampling method in Sec. IV-B1 in terms of sample efficiency and execution quality. As shown in Fig. 16 and summarized in Table III, the diffusion model, especially the DDIM variant, outperforms iterative sampling in both sample efficiency and execution time. With only 8 iterations and 0.15s planning time, DDIM achieves a comparable execution time of 20.6s to the iterative sampling method after 5000 iterations, demonstrating a 35-fold reduction in iterations. Iterative sampling with the mode library as introduced in

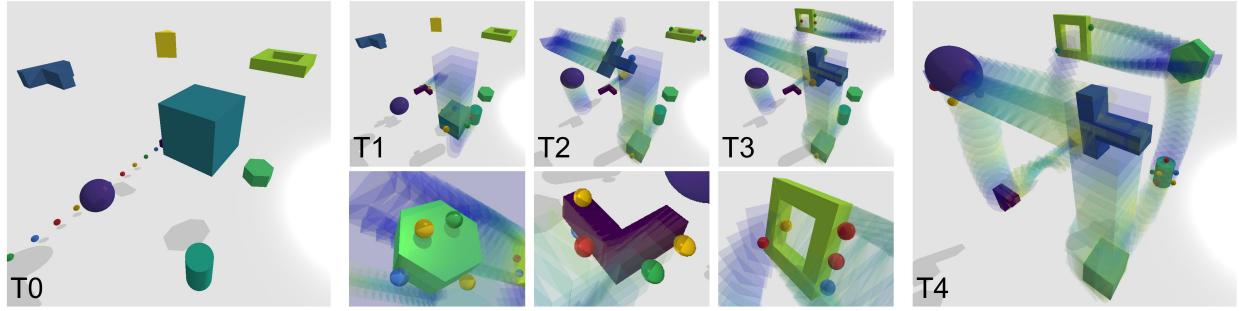


Fig. 13. Generalization of the proposed method to 6D pushing tasks, where the positions of 8 objects are swapped via 12 robots. Note that the robots switch subtasks frequently due to the cluttered workspace and the large number of partial order dependencies between the subtasks.

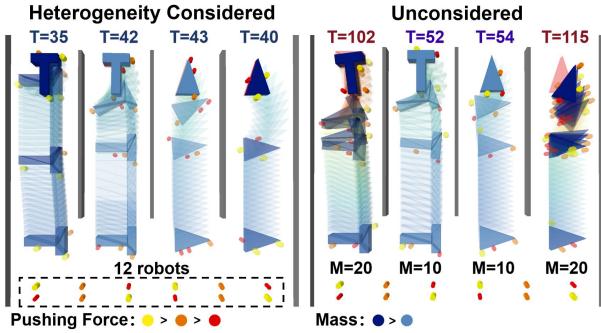


Fig. 14. Generalization of the proposed method to 12 heterogeneous robots with varying maximum pushing forces, and 2 objects with different masses: heterogeneity is considered (**Left**) and neglected (**Right**) in the hybrid optimization, resulting in significantly different execution times.

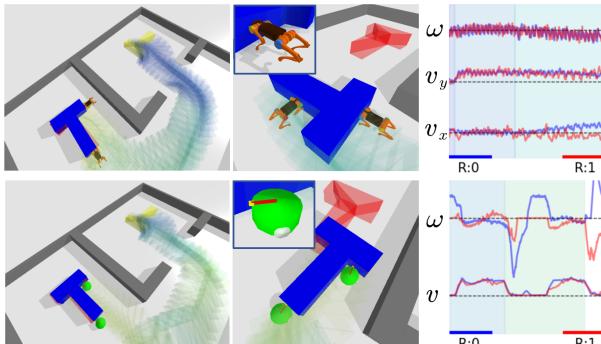


Fig. 15. Generalization of the proposed method to non-holonomic robots such as quadruped and differential-drive robots. Snapshots along with velocity control inputs are also shown.

Sec. IV-B2 shows diminishing returns beyond 1000 iterations, with execution time plateauing at around 20s despite a 408% increase in planning time. Without the diffusion policy or the mode library, the computational cost becomes prohibitive, requiring 453s for planning with only marginal improvement in execution time. This highlights that diffusion models distill knowledge from high-iteration sampling into efficient generative models, offering a computationally superior alternative. Furthermore, the diffusion-generated plans exhibit execution feasibility comparable to those from iterative sampling, confirming their accuracy despite a generation time of just 0.15s.

5) *Comparison of Single-object Pushing:* We compare the performance of the proposed hybrid optimization scheme (**HybDIF**) against six baselines in single-object pushing tasks: (I) **KGHS**, the keyframe-guided hybrid search without neural acceleration; (II) **FDIF**, a diffusion-based algorithm that generates a complete hybrid plan, which incorporates image

TABLE III  
COMPARISON BETWEEN DIFFUSION AND ITERATIVE SAMPLING

Method	ITR <sup>1</sup>	PT (s) <sup>2</sup>	ET (s) <sup>3</sup>
<b>Diffusion (DDPM)</b>	100	1.60	20.4
<b>Diffusion (DDIM)</b>	8	0.15	20.6
<b>Samp. (w/ mode library)</b>	10	0.05	46.8
	100	0.22	32.1
	1000	2.45	24.5
	5000	12.4	20.2
<b>Samp. (w/o mode library)</b>	5000	453	21.6

<sup>1</sup> Number of iterations or inference steps. <sup>2</sup> Planning time per trial. <sup>3</sup> Execution time per trial.

observations for cluttered environments; (III) **HMS**, which uses heuristic sampling for mode generation without further evaluation and optimization; (IV) **CMT**, adapted from [29] without co-optimizing the path and modes; (V) **MARL**, a multi-agent reinforcement learning baseline based on the PPO algorithm [49]; (VI) **PG-RL**, an extension of MARL with a guiding path planning module, from which a local target position is selected for guiding RL agents.

As shown in Fig. 17, we evaluate two scenarios: a nominal one with a T-shaped object and obstacles, and a perturbed one where the object is deformed and obstacles are rotated. FDIF, RL, and PGRL sample data only in the nominal scenario, while the perturbed environment is unseen. In contrast, HybDIF collects data exclusively in free space, independent of obstacle layouts, and *does not* include the deformed object. For both scenarios, initial and goal positions are random, and each algorithm is tested 30 times. Table IV reports five metrics (success rate, execution time, planning time, control cost, collision count) for  $N = 1, 2, 4$  robots.

In the nominal scenario, **HybDIF** achieves the highest success rate (100%) with the shortest execution and planning times across all robot numbers. It also maintains the lowest control cost and collision count, demonstrating superior overall performance. KGHS also achieves 100% success for  $N = 2, 4$ . However, its execution and planning times, along with control costs, are significantly higher than HybDIF. In contrast, FDIF has a lower success rate and higher planning time and control costs than HybDIF and KGHS. CMT achieves a success rate of 96% for  $N = 4$ , but for  $N = 1, 2$ , it has a much lower success rate (72% and 84%), as the smaller matching degree between the path and the contact modes. Although HMS achieves a 100% success rate for  $N = 2, 4$  it has the longest execution time and highest control cost. PGRL

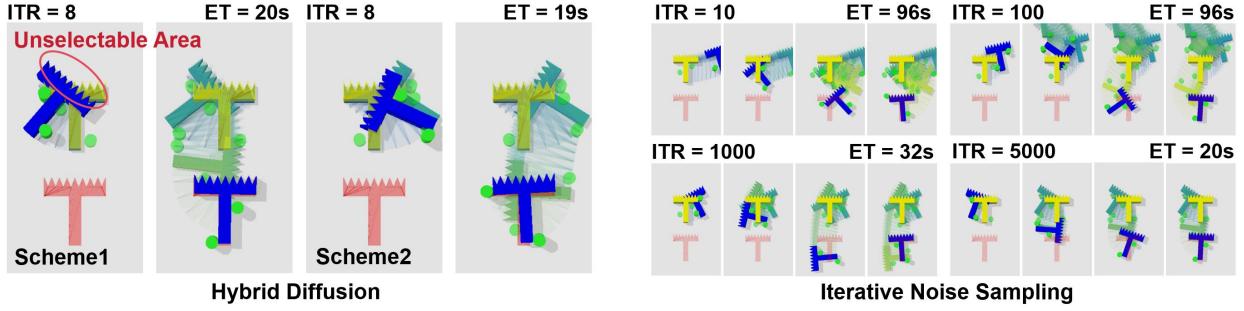


Fig. 16. Comparison of the diffusion-based method (**Left**) and iterative sampling (**Right**) for generating keyframes and pushing modes. The diffusion method generates multi-modal policies in 8 iterations with an execution time of 19s, while iterative sampling requires 5000 iterations for similar plans.

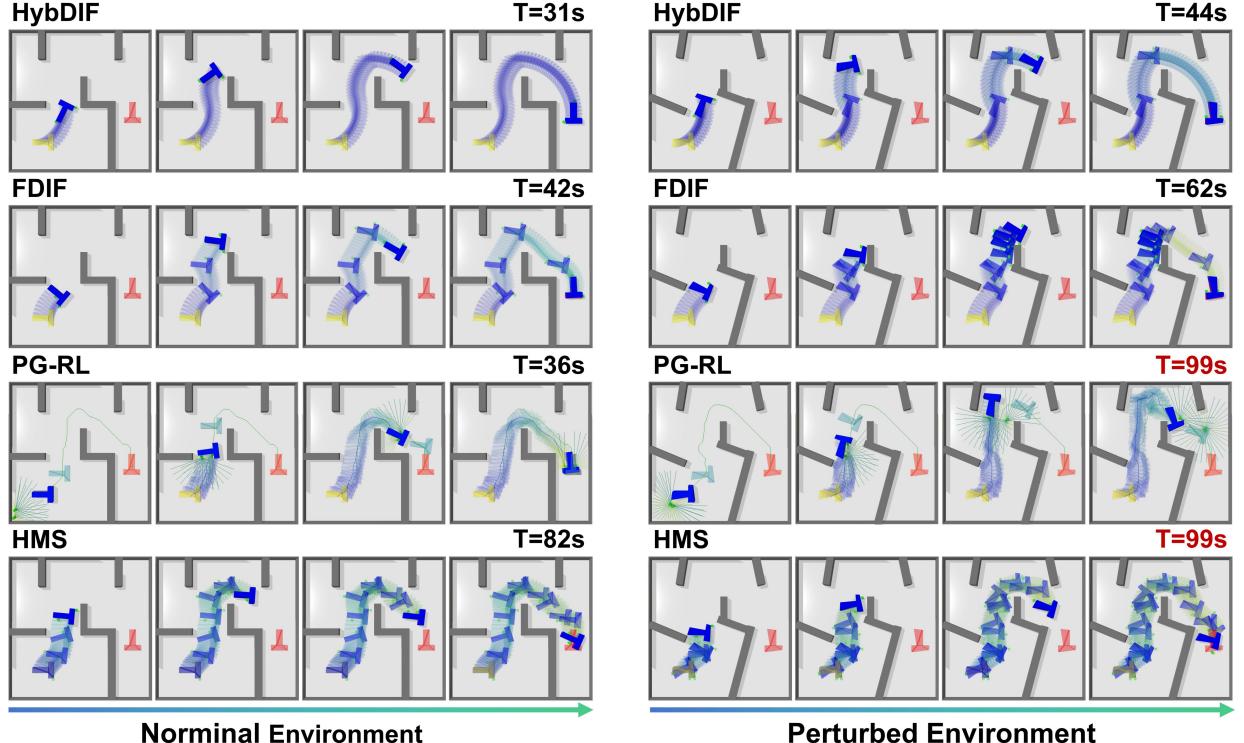


Fig. 17. Comparison of the proposed method with 5 baselines in the single-object pushing task, in a nominal environment that are similar to training datasets (**Left**) and a perturbed environment where obstacles are tilted and objects are deformed (**Right**).

performs competitively with a success rate around 96%, but its control cost and collision count exceed HybDIF. Lastly, MARL has the lowest success rate and highest collision count, indicating poor performance for long-distance pushing tasks. As analyzed in Fig. 19, PGRL reduces the difficulty of learning long-distance pushing strategies by selecting local goals.

More importantly, in the perturbed scenario, HybDIF maintains the best performance with a 100% success rate, the shortest execution time and the fastest planning time across all robot numbers, validating its robustness to perturbations in obstacle layout and object shape. In contrast, FDIF experiences a significant drop in success rate. As shown in Fig. 18, the diffusion model generates valid plans effectively in the nominal scenario. However, the generated keyframes are often *in collision* within the perturbed scenario due to insufficient training data that fails to cover a diverse range of obstacle distributions and object shapes, yielding a much higher collision count by 720%. Similar degradation can be found in MARL and PGRL, with a reduced success rate

(by 10% and 26%) and increased collision count (by 27% and 233%). Though PGRL still maintains an advantage compared to MARL, it experiences a more significant performance drop, indicating its vulnerability to the environment perturbations.

**6) Comparison of Multi-object Pushing:** To further evaluate the performance of the proposed neural accelerated combinatorial-hybrid optimization method (**NACHO**), the following three baselines are considered: (I) **CHO**, which uses the same model-based task assignment and hybrid optimization modules as NACHO, but without diffusion-based neural acceleration; (II) **SCE**, which disregards the heterogeneity of robots and objects and instead assumes uniform capabilities, e.g., identical object weights and robot force limits in both task assignment and mode generation; (III) **NTD**, which retains the task-allocation algorithm of [23] but omits the MAPF-based sub-task decomposition and partial-order construction, so the robots push the object directly along the raw MAPF path.

Fig. 20 shows snapshots of one execution in the nominal scenario with varying object masses and shapes. NACHO

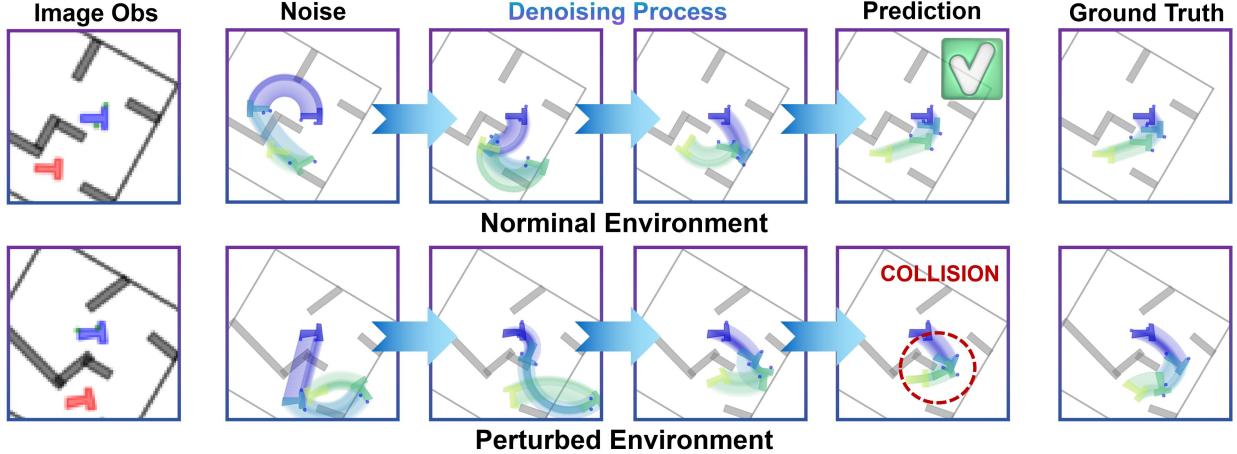


Fig. 18. The denoising process of the baseline method FDIF in both the nominal scenario (**Top**) and perturbed scenario (**Bottom**), where the predicted keyframes appear in collision within the perturbed scenario, which would lead to a higher collision count and frequent replanning in online execution of FDIF.

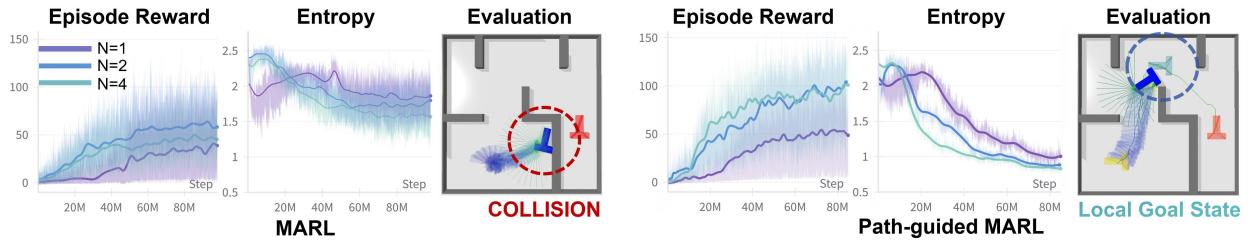


Fig. 19. The training and testing results of MARL-based baselines: the vanilla MARL [49] (**Left**) and the path-guided PG-RL (**Right**).

TABLE IV

COMPARISON OF SINGLE-OBJECT TASKS WITH  $N = 1, 2, 4$  ROBOTS FOR NOMINAL AND PERTURBED ENVIRONMENTS (AVERAGED OVER 30 TESTS)

Env	Method	Success Rate			Execution Time			Planning Time			Control Cost			Collision Count		
		N=1	N=2	N=4	N=1	N=2	N=4	N=1	N=2	N=4	N=1	N=2	N=4	N=1	N=2	N=4
Nominal	HybDIF(ours)	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>35.8</b>	<b>21.5</b>	<b>22.4</b>	<b>1.87</b>	<b>0.88</b>	1.02	<b>25.2</b>	17.7	<b>13.1</b>	<b>0.12</b>	<b>0.03</b>	<b>0.03</b>
	KGHS	0.97	<b>1.00</b>	<b>1.00</b>	38.2	28.3	23.2	4.76	3.67	4.25	26.1	17.3	14.2	0.31	<b>0.03</b>	<b>0.03</b>
	FDIF	0.87	0.94	0.97	46.3	38.4	32.5	3.26	2.10	1.25	31.4	24.5	20.7	0.65	0.31	0.25
	CMTC	0.72	0.84	0.97	95.3	68.7	50.1	10.5	4.20	3.10	80.4	32.6	24.9	0.30	0.04	0.05
	HMS	0.93	<b>1.00</b>	<b>1.00</b>	70.2	40.8	34.0	8.52	2.40	2.00	76.3	28.5	22.4	0.80	0.20	0.07
	PGRL	0.97	<b>1.00</b>	<b>1.00</b>	47.9	25.6	28.3	3.71	1.06	<b>0.98</b>	27.3	<b>16.8</b>	16.2	0.52	0.30	0.27
	MARL	0.53	0.64	0.55	67.2	52.2	59.4	2.04	1.73	2.12	40.2	24.1	28.2	0.72	1.43	1.17
Perturbed	HybDIF(ours)	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>36.4</b>	<b>28.1</b>	<b>22.3</b>	<b>2.04</b>	<b>0.98</b>	<b>1.15</b>	<b>26.3</b>	<b>17.6</b>	<b>12.8</b>	<b>0.22</b>	0.06	<b>0.03</b>
	KGHS	0.97	<b>1.00</b>	<b>1.00</b>	36.7	29.5	23.7	4.82	3.92	4.18	28.1	18.3	15.3	0.32	<b>0.03</b>	<b>0.03</b>
	FDIF	0.72	0.82	0.86	52.1	42.3	37.9	5.90	4.92	2.96	36.5	28.9	25.1	1.21	0.60	0.42
	CMTC	0.65	0.76	0.93	102.4	73.2	55.8	11.7	4.60	3.30	85.7	34.8	25.7	0.32	0.10	0.08
	HMS	0.85	<b>1.00</b>	<b>1.00</b>	82.0	48.5	38.1	9.53	3.05	2.34	87.9	32.6	23.0	0.85	0.26	0.08
	PGRL	0.64	0.75	0.82	64.5	56.4	44.3	2.06	2.12	1.94	38.4	27.6	21.6	1.50	1.23	0.87
	MARL	0.47	0.57	0.52	65.1	62.7	58.1	1.98	1.62	1.19	39.5	24.1	22.1	1.16	1.72	1.34

completes the task first at  $t = 70$ s, while CHO is slower without neural acceleration. Due to the simplified module of cost estimation, SCE is much slower and often exhibits inefficient behaviors, e.g., heavier objects such as the table and the semi-circle are assigned to only two robots. Without the ordering of subtasks, NTD fails to complete the task as the objects often block each other during execution.

Moreover, a quantitative comparison across 50 tasks in different environments is provided in Table V and Fig. 21. MAPF and task decomposition take about 9.1s across all methods, while task assignment and hybrid optimization take 1.82s and 2.64s for NACHO, respectively. NACHO achieves the shortest planning (14.7s) and execution time (78.7s), outperforming other methods by 37.8%. In contrast, SCE is the slowest with an execution time of 94.7s, highlighting the importance of considering robot and object heterogeneity. NTD has the

lowest success rate (44%) and high execution time (86.2s), proving that simply following the MAPF path is infeasible.

7) *Geometric Pushing Puzzle for Ants*: A recent study in [50] poses a hard geometric puzzle for hundreds of ants to *lift, push or pull* a large dumbbell-shaped object through two narrow passages in confined space, as shown in Fig. 22. The ants demonstrate amazing coordination and collaboration to accomplish this task, despite their limited field of view and local communication. For (maybe unfair) comparisons, a similar scenario is replicated, for which 1 and 6 robots are deployed to see different strategies. Note that without the ability to pull, the robots can still accomplish the task via the proposed scheme. It takes 103s and 23 modes for 1 robot, and 52s and 9 modes for 6 robots. Nonetheless, the observations in [50] have inspired our future work towards distributed coordination schemes with only local communication.

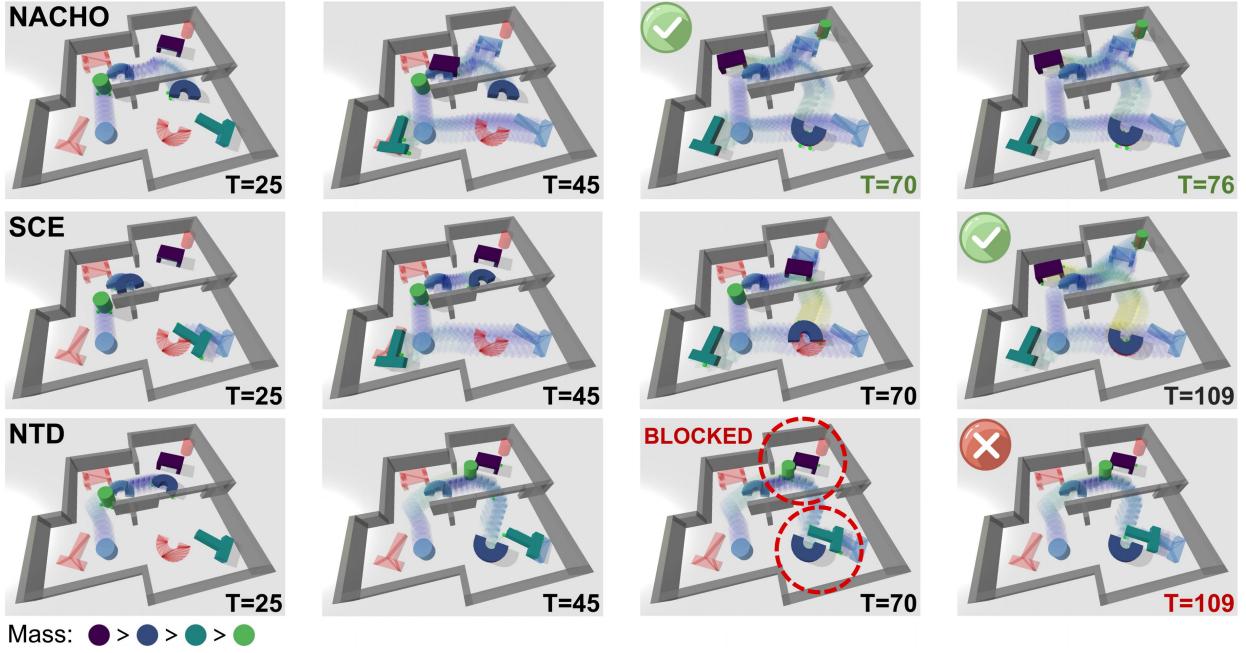


Fig. 20. Comparison of different baseline methods for the multi-object pushing tasks, where 4 objects have different shapes and masses.

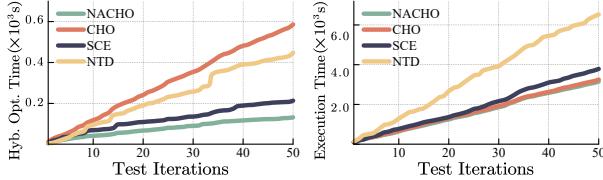


Fig. 21. Accumulated planning time (**Left**) and accumulated execution time (**Right**) over 50 multi-object pushing tasks.

TABLE V

COMPARISON WITH 3 BASELINES FOR THE MULTI-OBJECT PUSHING TASK

Algorithm	SR <sup>1</sup>	MDT <sup>2</sup>	TAT <sup>3</sup>	HOT <sup>4</sup>	ET <sup>5</sup>
NACHO(ours)	<b>1.00</b>	10.3	1.78	<b>2.64</b>	<b>78.7</b>
CHO	<b>1.00</b>	10.2	1.78	11.7	81.3
SCE	<b>1.00</b>	10.3	1.83	4.27	94.7
NTD	0.45	9.05	1.65	8.26	89.7

<sup>1</sup> Success rate. <sup>2</sup> Time for task decomposition.

<sup>3</sup> Time for task assignment. <sup>4</sup> Time for hybrid optimization.

<sup>5</sup> Execution time.

## B. Hardware Experiments

1) *System Description:* As shown in Fig. 23, the experiments are conducted in a  $5m \times 5m$  lab with the OptiTrack motion capture system for global positioning. Four identical Mecanum-wheel robots are deployed, each measuring  $0.2m \times 0.3m$  with a maximum pushing force of 10N. Each robot is equipped with a NVIDIA Jetson Nano running ROS for onboard communication and control. The centralized planning and control algorithm runs on a laptop (Intel Core i7-1280P), which wirelessly transmits the velocity commands  $\mathbf{u}_n$  in (12) to the robots, and executed by their onboard velocity controllers. Objects are made of cardboard, each weighing approximately 1kg, with a ground friction coefficient of 0.5 and a side friction coefficient of 0.2. These parameters were *not* meticulously calibrated, e.g., using force sensors, but are



Fig. 22. Hard geometric pushing puzzle proposed in [50] from the left side to the right side within the confined space. **Top:** the final object trajectory as pushed by hundreds of ants, abstracted from the recorded video in [50]; **Down:** the object trajectory via the proposed method with 1 robot (within 10s and 23 modes) and 6 robots (within 52s and 9 modes).

sufficient for the algorithm to complete the target tasks, as the kinematic-level control is relatively insensitive to those force-related parameters. With above setup, two scenarios were tested: (I) a T-shaped and an L-shaped object placed on opposite sides of a narrow passage formed by bar-shaped obstacles, with four robots swapping their positions under strict task ordering constraints; (II) an L-shaped object, a long rectangular object, and a T-shaped object arranged in a triangular layout, where four robots rotate the objects counter-clockwise around a center obstacle.

2) *Results:* As shown in Fig. 23, the proposed method successfully completes the pushing tasks in both scenarios. In Scenario-I, MAPF generates two collision-free trajectories in 8.2s, decomposed into 5 subtasks in 0.5s. Robots are divided into two subgroups in 0.2s for task assignment: one subgroup pushes the T-shaped object upwards, while the other pushes the L-shaped object right. By  $t = 40s$ , the T-shaped object reaches

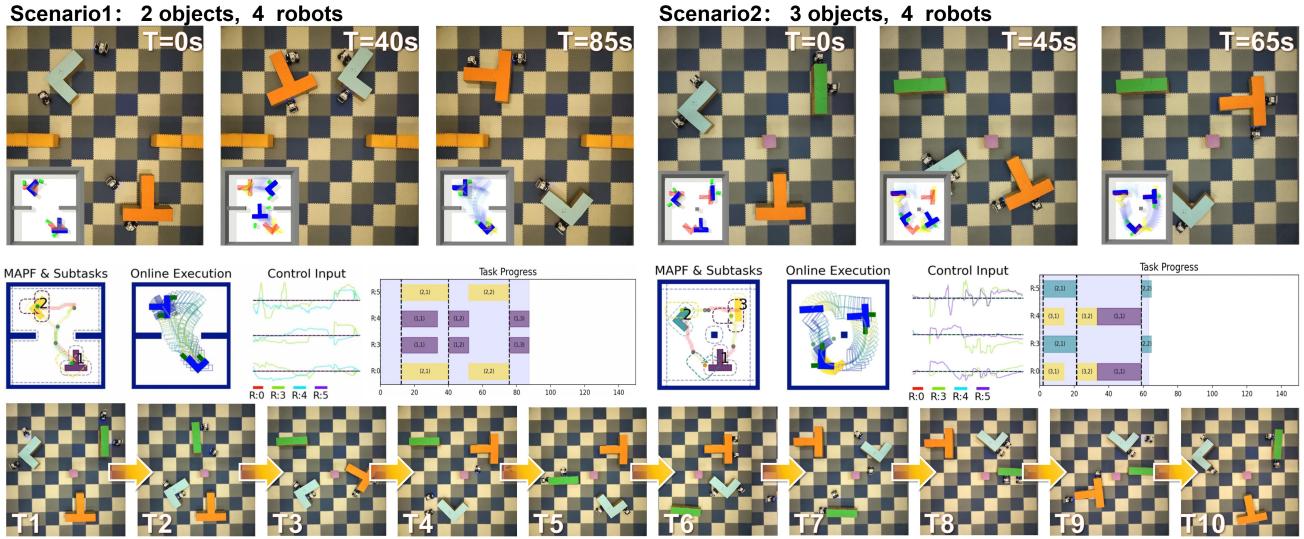


Fig. 23. Snapshots of the PushingBot system during hardware experiments, including the pushing subtasks, the hybrid plan, control inputs and Gantt graph of subtasks. **Left:** four robots are deployed for swapping two objects in Scenario-I; **Right:** four robots are deployed to rotate three objects in Scenario-II.

its target, and the L-shaped object begins its second subtask. All objects reach their goals by  $t = 85\text{s}$  with 7 modes. In Scenario-II, MAPF generates three timed trajectories in 1.2s, decomposed into 5 subtasks in 0.6s. Robots are divided into two subgroups in 0.2s. In particular, the L-shaped object subtask (2, 1) is assigned to one subgroup, and the rectangular object subtask (3, 1) is assigned to the other. By  $t = 45\text{s}$ , the rectangular object has reached its target position, clearing space for the T-shaped object. The L-shaped and rectangular objects are pushed to their targets by  $t = 45\text{s}$ , and the remaining tasks are completed by  $t = 65\text{s}$  with 6 modes. The real-time progress and velocity control inputs are recorded in Fig. 23. Note that the actual execution time fluctuates due to the poor motion of Mecanum wheels in certain directions around  $45^\circ$ , causing drifting and slipping. However, the partial ordering of subtasks and the online adaptation scheme ensure the system consistently completes the tasks. More details can be found in the attached videos.

## VI. CONCLUSION

This work proposes a combinatorial-hybrid optimization scheme for synthesizing collaborative pushing strategies for multiple robots and multiple objects. The scheme consists of two interleaved layers: the decomposition and assignment of pushing subtasks; and the hybrid optimization of pushing modes and forces for each subtask. Moreover, a diffusion-based predictor is trained to accelerate the generation of pushing modes during hybrid optimization. It is shown to be scalable and effective for general-shaped objects in cluttered scenes. The completeness and feasibility of our algorithm have been demonstrated both theoretically and experimentally. Future work includes partial knowledge of the objects, distributed coordination schemes, and other collaborative tasks with dynamical and physical constraints among the robots.

## ACKNOWLEDGMENT

The authors would like to thank Zhengyu Yang for his help on the hardware experiments.

## REFERENCES

- [1] S. Goyal, A. Ruina, and J. Papadopoulos, “Limit surface and moment function descriptions of planar sliding,” in *Proc. IEEE Int. Conf. Robot. Autom.*, 1989, pp. 794–795.
- [2] K. M. Lynch, H. Maekawa, and K. Tanie, “Manipulation and active sensing by pushing using tactile feedback,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, vol. 1, pp. 416–421.
- [3] F. R. Hogan and A. Rodriguez, “Reactive planar non-prehensile manipulation with hybrid model predictive control,” *Int. J. Robot. Res.*, vol. 39, no. 7, pp. 755–773, 2020.
- [4] T. Xue, H. Girgin, T. S. Lembono, and S. Calinon, “Guided optimal control for long-term non-prehensile planar manipulation,” in *Proc. IEEE Int. Conf. Robot. Autom.*, 2023, pp. 4999–5005.
- [5] S. Jeon, M. Jung, S. Choi, B. Kim, and J. Hwangbo, “Learning whole-body manipulation for quadrupedal robot,” *IEEE Robot. Autom. Lett.*, vol. 9, no. 1, pp. 699–706, 2023.
- [6] M. Toussaint, “Logic-geometric programming: An optimization-based approach to combined task and motion planning,” in *Proc. Int. Joint Conf. Artif. Intell.*, 2015, pp. 1930–1936.
- [7] M. A. Toussaint, K. R. Allen, K. A. Smith, and J. B. Tenenbaum, “Differentiable physics and stable modes for tool-use and manipulation planning,” in *Proc. Robot. Sci. Syst.*, 2018.
- [8] A. Zeng, S. Song, S. Welker, J. Lee, A. Rodriguez, and T. Funkhouser, “Learning synergies between pushing and grasping with self-supervised deep reinforcement learning,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 4238–4245.
- [9] Y. Xiao, J. Hoffman, T. Xia, and C. Amato, “Learning multi-robot decentralized macro-action-based policies via a centralized q-net,” in *Proc. IEEE Int. Conf. Robot. Autom.*, 2020, pp. 10695–10701.
- [10] P. García, P. Caamaño, R. J. Duro, and F. Bellas, “Scalable task assignment for heterogeneous multi-robot teams,” *Int. J. Adv. Robot. Syst.*, vol. 10, no. 2, p. 105, 2013.
- [11] C. R. Kube, “Task modelling in collective robotics,” *Auton. Robot.*, vol. 4, pp. 53–72, 1997.
- [12] L. Vig and J. A. Adams, “Multi-robot coalition formation,” *IEEE Trans. Robot.*, vol. 22, no. 4, pp. 637–649, 2006.
- [13] Z. Pan, A. Zeng, Y. Li, J. Yu, and K. Hauser, “Algorithms and systems for manipulating multiple objects,” *IEEE Trans. Robot.*, vol. 39, no. 1, pp. 2–20, 2022.
- [14] M. Guo and M. Bürger, “Geometric task networks: Learning efficient and explainable skill coordination for object manipulation,” *IEEE Trans. Robot.*, vol. 38, no. 3, pp. 1723–1734, 2022.
- [15] B. Kim, Z. Wang, L. P. Kaelbling, and T. Lozano-Pérez, “Learning to guide task and motion planning using score-space representation,” *Int. J. Robot. Res.*, vol. 38, no. 7, pp. 793–812, 2019.
- [16] V. N. Hartmann, A. Orthey, D. Driess, O. S. Oguz, and M. Toussaint, “Long-horizon multi-robot rearrangement planning for construction assembly,” *IEEE Trans. Robot.*, 2022.

- [17] Z. Wang, C. R. Garrett, L. P. Kaelbling, and T. Lozano-Pérez, “Learning compositional models of robot skills for task and motion planning,” *Int. J. Robot. Res.*, vol. 40, no. 6-7, pp. 866–894, 2021.
- [18] A. Torreño, E. Onaindia, A. Komenda, and M. Štolba, “Cooperative multi-agent planning: A survey,” *Comput. Surv.*, vol. 50, no. 6, pp. 1–32, 2017.
- [19] M. Gini, “Multi-robot allocation of tasks with temporal and ordering constraints,” in *Proc. AAAI Artif. Intell.*, 2017.
- [20] A. Khamis, A. Hussein, and A. Elmogy, “Multi-robot task allocation: A review of the state-of-the-art,” *Coop. Robot. Sensor Netw.*, pp. 31–51, 2015.
- [21] Z. Wang, C. Liu, and M. Gombolay, “Heterogeneous graph attention networks for scalable multi-robot scheduling with temporospatial constraints,” *Auton. Robots.*, vol. 46, no. 1, pp. 249–268, 2022.
- [22] M. Guo and M. M. Zavlanos, “Multirobot data gathering under buffer constraints and intermittent communication,” *IEEE Trans. Robot.*, vol. 34, no. 4, pp. 1082–1097, 2018.
- [23] S. Choudhury, J. K. Gupta, M. J. Kochenderfer, D. Sadigh, and J. Bohg, “Dynamic multi-robot task allocation under uncertainty and temporal constraints,” *Auton. Robots.*, vol. 46, no. 1, pp. 231–247, 2022.
- [24] E. Tuci, M. H. Alkilabi, and O. Akanyeti, “Cooperative object transport in multi-robot systems: A review of the state-of-the-art,” *Front. Robot. AI*, vol. 5, p. 59, 2018.
- [25] J. Chen, M. Gauci, W. Li, A. Kolling, and R. Groß, “Occlusion-based cooperative transport with a swarm of miniature mobile robots,” *IEEE Trans. Robot.*, vol. 31, no. 2, pp. 307–321, 2015.
- [26] Z. Tang, J. Chen, and M. Guo, “Combinatorial-hybrid optimization for multi-agent systems under collaborative tasks,” in *Proc. IEEE Conf. Decis. Control*, 2023.
- [27] M. Sombolestan and Q. Nguyen, “Hierarchical adaptive control for collaborative manipulation of a rigid object by quadrupedal robots,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2023, pp. 2752–2759.
- [28] M. Rosenfelder, H. Ebel, and P. Eberhard, “Force-based organization and control scheme for the non-prehensile cooperative transportation of objects,” *Robotica*, vol. 42, no. 2, pp. 611–624, 2024.
- [29] H. Ebel, M. Rosenfelder, and P. Eberhard, “Cooperative object transportation with differential-drive mobile robots: Control and experimentation,” *Robot. Auton. Syst.*, vol. 173, p. 104612, 2024.
- [30] Z. Tang, Y. Feng, and M. Guo, “Collaborative planar pushing of polytopic objects with multiple robots in complex scenes,” in *Proc. Robot. Sci. Syst.*, 2024.
- [31] H. Ebel and P. Eberhard, “Cooperative transportation: realizing the promises of robotic networks using a tailored software/hardware architecture,” *at - Automatisierungstechnik*, vol. 70, no. 4, pp. 378–388, 2022.
- [32] P. Florence, C. Lynch, A. Zeng, O. Ramirez, A. Wahid, L. Downs, A. Wong, J. Lee, I. Mordatch, and J. Tompson, “Implicit behavioral cloning,” *Proc. Conf. Robot. Learn.*, 2021.
- [33] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song, “Diffusion policy: Visuomotor policy learning via action diffusion,” in *Proc. Robot. Sci. Syst.*, 2023.
- [34] A. Simeonov, Y. Du, B. Kim, F. Hogan, J. Tenenbaum, P. Agrawal, and A. Rodriguez, “A long horizon planning framework for manipulating rigid pointcloud objects,” in *Proc. PMLR Conf. Robot Learn.*, 2021, pp. 1582–1601.
- [35] D. Driess, J.-S. Ha, and M. Toussaint, “Learning to solve sequential physical reasoning problems from a scene image,” *Int. J. Robot. Res.*, vol. 40, no. 12-14, pp. 1435–1466, 2021.
- [36] A. Mandlekar, D. Xu, R. Martín-Martín, S. Savarese, and L. Fei-Fei, “Learning to generalize across long-horizon tasks from human demonstrations,” *Proc. Robot. Sci. Syst.*, 2020.
- [37] 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 *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2023, pp. 1916–1923.
- [38] 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 *Proc. Int. Symp. Combin. Search*, vol. 10, no. 1, 2019, pp. 151–158.
- [39] J. Ho, A. Jain, and P. Abbeel, “Denoising diffusion probabilistic models,” *Adv. Neural Inf. Process. Syst.*, vol. 33, pp. 6840–6851, 2020.
- [40] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, “Conflict-based search for optimal multi-agent pathfinding,” *Artif. Intell.*, vol. 219, pp. 40–66, 2015.
- [41] T. Terlaky, *Interior point methods of mathematical programming*. Springer Sci. Bus. Media, 2013, vol. 5.
- [42] F. M. Yamada, H. C. Batagelo, J. P. Gois, and H. Takahashi, “Generative approaches for solving tangram puzzles,” *Disc. Artif. Intell.*, vol. 4, no. 1, p. 12, 2024.
- [43] C. Leet, J. Li, and S. Koenig, “Shard systems: Scalable, robust and persistent multi-agent path finding with performance guarantees,” in *Proc. AAAI Artif. Intell.*, vol. 36, no. 9, 2022, pp. 9386–9395.
- [44] E. Coumans and Y. Bai, “Pybullet, a python module for physics simulation for games, robotics and machine learning,” <http://pybullet.org>, 2016–2019.
- [45] G. v. d. Bergen, “A fast and robust gjk implementation for collision detection of convex objects,” *J. Graph. Tools*, vol. 4, no. 2, pp. 7–25, 1999.
- [46] S. Diamond and S. Boyd, “Cvxpy: A python-embedded modeling language for convex optimization,” *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 2909–2913, 2016.
- [47] I. Zeidis and K. Zimmermann, “Dynamics of a four-wheeled mobile robot with mecanum wheels,” p. e201900173, 2019.
- [48] E. Deutsch and K. Hayes Jr, “A heuristic solution to the tangram puzzle,” *Mach. Intell.*, vol. 7, pp. 205–240, 1972.
- [49] C. Yu, A. Velu, E. Vinitsky, J. Gao, Y. Wang, A. Bayen, and Y. Wu, “The surprising effectiveness of ppo in cooperative multi-agent games,” *Adv. Neural Inf. Process. Syst.*, vol. 35, pp. 24611–24624, 2022.
- [50] T. Dreyer, A. Haluts, A. Korman, N. Gov, E. Fonio, and O. Feinerman, “Comparing cooperative geometric puzzle solving in ants versus humans,” *Proc. Natl. Acad. Sci.*, vol. 122, no. 1, p. e2414274121, 2025.
- [51] I. Kao, K. M. Lynch, and J. W. Burdick, “Contact modeling and manipulation,” *Springer Handb. Robot.*, pp. 931–954, 2016.

## APPENDIX

TABLE VI  
NOMENCLATURE TABLE

Term	Definition	Reference
Pushing Mode ( $\xi$ )	Contact points and forces between robots and the target object.	Sec. III-A
MAPF Path ( $\mathcal{S}_m$ )	Collision-free timed trajectory.	Eq. (2)
Subtask ( $\mathcal{S}_m^k$ )	$k$ -th segment of object path.	Eq. (3)
Partial Order ( $\leq$ )	Temporal ordering between the sub-tasks.	Def. 2
Task Plan ( $\tau_i$ )	Timed sequence of subtasks as the local plan for robot $i$ .	Def. 3
Subgroup ( $\mathcal{N}_m^k$ )	Robot coalition for subtask.	Sec. IV-A2
Keyframe ( $\kappa_\ell$ )	Critical system state.	Sec. IV-B1
Hybrid Plan ( $\vartheta$ )	A sequence of keyframes and pushing modes.	Eq. (8)
Arc Segment ( $\varrho_\ell$ )	Trajectory between keyframes.	Sec. IV-B1
Primitive Plan ( $\hat{\vartheta}$ )	Feasible hybrid plan as sequence of keyframes and modes.	Alg. 2
Diffusion Model	Diffusion-based neural network for generating hybrid plans.	Sec. IV-B2
Primitive Lib. ( $\mathcal{X}$ )	Library that contains verified hybrid plans for different subtasks.	Alg. 2
Mode Lib.	Library of verified modes.	Alg. 2

### A. Modeling and Mode Feasibility

1) *Pushing Modes and Coupled Dynamics:* Given an object  $m$  and a set of robots  $\mathcal{N}_m$ , working at a particular pushing mode  $\xi$ , the robots can apply pushing forces in different directions with different magnitude. Denote by  $\mathbf{f}_1 \mathbf{f}_2 \dots \mathbf{f}_{N_m}$ , where  $\mathbf{f}_n \in \mathbb{R}^2$  is the contact force of robot  $R_n$  at contact point  $\mathbf{c}_n$ ,  $\forall n \in \mathcal{N}_m$ . Furthermore, each force  $\mathbf{f}_n$  can be decomposed in the directions of the normal vector  $\mathbf{n}_n$  and the tangent vector  $\boldsymbol{\tau}_n$  w.r.t. the object surface at the contact point  $\mathbf{c}_n$ , i.e.,  $\mathbf{f}_n = \mathbf{f}_n^n + \mathbf{f}_n^t \triangleq f_n^n \mathbf{n}_n + f_n^t \mathbf{n}_n^\perp$ . Due to the Coulomb law of friction see [51], it holds that:

$$0 \leq f_n^n \leq f_{n,\max}; \quad 0 \leq |f_n^t| \leq \mu_m^c f_n^n, \quad (14)$$

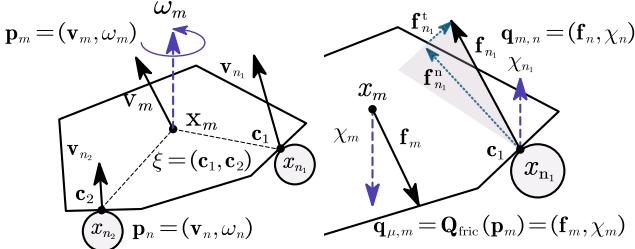


Fig. 24. Illustration of a pushing mode and relevant notations in Sec. A.1.

where  $f_{n,\max} > 0$  is the maximum force each robot can apply; and  $\mu_m^c$  is the coefficient of lateral friction defined earlier. Then, these decomposed forces can be re-arranged by:

$$\mathbf{F}_\xi \triangleq (\mathbf{F}_\xi^n, \mathbf{F}_\xi^t) \triangleq (f_1^n, \dots, f_{N_m}^n, f_1^t, \dots, f_{N_m}^t) \in \mathbb{R}^{2N_m}, \quad (15)$$

and further  $\mathcal{F}_\xi \triangleq \{\mathbf{F}_\xi\}$  denotes the set of all forces within each mode  $\xi \in \Xi_m$ . Furthermore, the combined generalized force  $\mathbf{q}_\xi \triangleq (\mathbf{f}^\star, \chi^\star)$  as also used in [2] is given by:

$$\mathbf{f}^\star \triangleq \sum_{n=1}^{N_m} \mathbf{f}_n; \quad \chi^\star \triangleq \sum_{n=1}^{N_m} (\mathbf{c}_n - \mathbf{x}_n) \times \mathbf{f}_n, \quad (16)$$

where  $\times$  is the cross product and  $\chi^\star$  is the resulting torque from all robots. It can be written in matrix form  $\mathbf{q}_\xi \triangleq \mathbf{J}\mathbf{F}_\xi$ , where  $\mathbf{J} \triangleq \nabla_{\mathbf{F}_\xi} \mathbf{q}_\xi$  is a  $3 \times 2N_m$  Jacobian matrix. Similarly, let  $Q_\xi \triangleq \{\mathbf{q}_\xi\}$  denote the set of all *allowed* combined generalized forces within each mode  $\xi \in \Xi_m$ . The coupled dynamics of the object and robots can be described as follows:

$$\mathbf{M}_m \dot{\mathbf{p}}_m = \mathbf{q}_{\mu_m} + \mathbf{q}_{\xi_m} = Q_\mu^m(\mathbf{p}_m) + \sum_{n \in \mathcal{N}_m} \mathbf{q}_{m,n}; \quad (17a)$$

$$\mathbf{M}_n \dot{\mathbf{p}}_n = Q_{\text{drv}}^n(\mathbf{u}_n, \mathbf{s}_n, \mathbf{p}_n) - \mathbf{q}_{m,n}, \quad (17b)$$

where  $\mathbf{M}_m \triangleq \text{diag}(\mathbf{M}_m, \mathbf{M}_m, \mathbf{I}_m)$ ;  $\mathbf{q}_{m,n}$  is the generalized pushing force in (16) applied by robot  $R_n$  on object  $\Omega_m$ ; and  $\mathbf{q}_{m,\mu} \triangleq Q_\mu^m(\mathbf{p}_m) = (\mathbf{f}_m, \chi_m)$  is the ground friction, determined by the velocity  $\mathbf{p}_m$  and object intrinsics. (17a) models the object's motion under external forces  $\mathbf{q}_\xi$  and  $\mathbf{q}_\mu$  within mode  $\xi$ , while (17b) describes the robot's motion under the control inputs  $\mathbf{u}_n$  and the reaction force from the object. These equations are instrumental for system modeling and subsequent analysis, while the simulation is handled by the physics engine. Denote by  $\xi_m(t) \triangleq (\xi_m(t), \mathbf{q}_{\xi_m}(t), \mathcal{N}_m(t))$  the contact points, pushing forces and participants for object  $m \in \mathcal{M}$  at time  $t \geq 0$ .

2) *Quasi-static Analyses*: The friction force  $Q_\mu(\mathbf{p}_m)$  in (17a) lacks a closed-form expression. As also adopted in [2], [30], the quasi-static analyses assume that the motion of the target is sufficiently slow, such that its acceleration is approximately zero and the inertia forces can be neglected. Consequently, given the desired velocity  $\mathbf{p}_m^*$  of the object, the generalized friction force  $\mathbf{q}_{m,\mu}$  is computed as:

$$\mathbf{q}_{\mu_m} \triangleq \tilde{Q}_\mu^m(\mathbf{p}_m^*) = -\|\mathbf{D}_1 \mathbf{D}_2 \mathbf{p}_m\|_2^{-1} \mathbf{D}_2 \mathbf{p}_m^*, \quad (18)$$

where  $\tilde{Q}_\mu^m(\cdot)$  approximates  $Q_\mu(\cdot)$  in (17a);  $\mathbf{D}_1 \triangleq \text{diag}(f_{\max}, f_{\max}, m_{\max})^{-1}$ ,  $\mathbf{D}_2 \triangleq \text{diag}(1, 1, m_{\max}^2/f_{\max}^2)$ ;  $f_{\max}$  and  $m_{\max}$  are the maximum ground friction and moment of the target object  $\Omega_m$ . Furthermore, assuming no slipping occurs during the pushing process, and the robot continuously applies a net force that exactly counteracts the frictional

force, i.e.,  $\mathbf{q}_{m,\xi} = -\mathbf{q}_{\mu_m}$ , the object can be pushed at a constant body-frame velocity  $\mathbf{p}_m^B$ . Under this condition, the resulting trajectory forms a circular arc, expressed as:  $\Delta s(t) \triangleq \left( \int_{t=0}^t \text{Rot}(\omega t + \psi_0) dt \right) \mathbf{p}_m^B$ , where  $\text{Rot}(\cdot)$  denotes the rotation matrix,  $\omega$  is the angular velocity in  $\mathbf{p}^B$ , and  $\psi_0$  is the initial orientation of the object. Conversely, given any two states,  $s_\ell$  and  $s_{\ell+1}$ , the corresponding arc trajectory  $\varrho_\ell = \widehat{s_\ell s_{\ell+1}}$  can be uniquely determined, along with the body-frame velocity  $\mathbf{p}_\ell^B$  that generates it.

3) *Feasibility of Pushing Modes*: We define the *primary feasibility loss*  $J_F^m$  to evaluate whether the pushing forces  $\mathbf{q}_{m,\xi}$  are sufficient to counteract frictional forces while maintaining the object's motion at the desired body-frame velocity  $\mathbf{p}^B$ :

$$J_F^m(\xi, \mathbf{p}^B) \triangleq \min_{\mathbf{q}_{m,\xi} \in Q_{m,\xi}} \|\mathbf{q}_{m,\xi} + Q_\mu^m(\mathbf{p}^B)\|_1, \quad (19)$$

where  $\|\cdot\|_1$  is the first norm. To account for the robustness of the mode against perturbations in velocity direction, we introduce the *multi-directional feasibility loss*  $J_{MF}^m$ , which aggregates the primary loss over a set of basis velocities  $\mathcal{D}$ :

$$J_{MF}^m(\xi, \mathbf{p}^B) \triangleq \sum_{\mathbf{p}_d \in \mathcal{D}} w_d \cdot J_F^m(\xi, \mathbf{p}_d), \quad (20)$$

where  $\mathcal{D}$  denotes a set of basis velocities that span the generalized velocity space, including the desired direction  $\mathbf{p}^B$ . Each basis direction  $\mathbf{p}_d$  is associated with a weight  $w_d$ , with the primary direction assigned the highest weight. This loss is introduced as a soft measure to account for infeasibility and uncertainty in certain scenarios. A lower  $J_{MF}^m$  indicates greater robustness of the pushing mode  $\xi$ , i.e., it can tolerate perturbations in velocity direction while maintaining feasibility.

**Definition 4.** (I) A mode  $\xi$  is *force-feasible* for a target velocity  $\mathbf{p}^B$  if and only if  $J_F^m(\xi, \mathbf{p}^B) = 0$ ; (II) A mode  $\xi$  is *practically feasible* for a target velocity  $\mathbf{p}^B$  if and only if the robot can stably push the object at velocity  $\mathbf{p}^B$ . ■

Force feasibility is necessary but not sufficient for practical feasibility. For example, robots without force sensing cannot precisely apply desired forces via position control alone. Thus, given a pushing mode  $\xi$  and a target velocity  $\mathbf{p}^B$ , the feasibility check is performed in two stages: (I) Force feasibility: Verify if  $J_F^m(\xi, \mathbf{p}^B) = 0$  holds; (II) Practical feasibility: If force-feasible, verify the mode in simulation by tracking the arc trajectory  $\varrho_p$  generated by  $p$ . If successful, the mode is considered practically feasible. Additionally, for real-world execution, a mode library is maintained to record the mode and observed tracking error. This helps bridge the gap between simulation-based feasibility checks and physical execution.

4) *Mode Sufficiency*: The subgroup of robots  $\mathcal{N}_m$  is said to be *mode-sufficient* for pushing object  $\Omega_m$  if there exists a set of velocities  $P_m^* \triangleq \{\mathbf{p}_j^*\}$  such that: (I)  $P_m^*$  can positively span the  $\mathbb{R}^3$  space; and (II) the robots in  $\mathcal{N}_m$  can push object  $\Omega_m$  in each velocity  $\mathbf{p}_j^* \in P_m^*$ , i.e., there exists a practically feasible mode  $\xi_j$  for each velocity  $\mathbf{p}_j^*$ . The set  $P_m^*$  satisfying the above sufficient conditions is referred to as the set of feasible velocities for the system. Verification of this condition can be done by iteratively expand the set of feasible velocities, i.e., by selecting new velocity direction  $p$  and generate modes

by (7). If a feasible mode is found, this velocity is added to the set  $P_m^*$ . This process is repeated until when the set  $P_m^*$  can span the  $\mathbb{R}^3$  space or the set of velocities is exhausted.

### B. Proof of Lemmas and Theorems

**Proof. of Lemma 1.** Assume that after certain iterations, the largest splitting instance  $t_m^*$  remains unchanged for all objects, and there exists  $m \in \mathcal{M}$  such that  $t_m^* < t_L$ . Let  $m^* \triangleq \operatorname{argmin}_{m \in \mathcal{M}} \{t_m^*\}$ . (I) If  $\forall m', t_m^* < t_{m'}^*$ , the next splitting instance  $t_{m^*}^s$  is determined by (5), which implies that  $t_{m^*}^s \geq t_{m'}^* > t_{m^*}^*$ . Thus  $t_{m'}^c$  will increase, contradicting the assumption. (II) If there exists  $m'$  such that  $t_{m'}^* = t_{m^*}^*$ , which implies that two objects collide at same time  $t_{m^*}^*$ , which contradicts collision-free assumption of the timed paths. Thus,  $t_m^*$  can reach  $t_L$  in finite steps and the algorithm terminates. Moreover, assume that there exists a loop in the ordering i.e.,  $\mathfrak{S}_m^k \leq \dots \leq \mathfrak{S}_m^k$ . If segment  $\mathfrak{S}_m^k \leq \mathfrak{S}_{m'}^{k'}$ , then it holds that  $t_{m'}^{k,c} \leq t_{m'}^{k',c}$ . Following (5),  $\mathfrak{S}_{m'}^{k'}$  can only be created after the segment  $\mathfrak{S}_m^k$  has been created and removed from  $\mathfrak{S}_m$ . Thus,  $\mathfrak{S}_m^k$  must be created before itself, which contradicts the assumption, the partial ordering is strict without loops.  $\square$

**Proof. of Lemma 2.** To begin with, the segments of the same object are followed according to the sequence in  $\bar{\mathfrak{S}}_m$ , i.e., sequentially from  $\mathfrak{S}_m^1$  to  $\mathfrak{S}_m^{K_m}$ . Then, given any two segments  $\mathfrak{S}_{m_1}^{k_1}$  and  $\mathfrak{S}_{m_2}^{k_2}$  of different objects  $m_1$  and  $m_2$ , consider the following two cases: if two segments are not partially ordered, objects  $m_1$  and  $m_2$  can be moved concurrently; otherwise, if  $\mathfrak{S}_{m_1}^{k_1} \leq \mathfrak{S}_{m_2}^{k_2}$ , the second condition requires that the subsequent object  $m_2$  can only be moved after the preceding segment of object  $m_1$  has been traversed. In this way, if each segment is traversed within a bounded time, each object can reach its goal state without collision.  $\square$

**Proof. (Sketch) of Lemma 3.** The key insight is that any arc in collision will be split into two shorter arcs, until all the arcs are collision-free. For these collision-free arcs, the iterative sampling procedure will try to find intermediate keyframes and feasible modes. If the iterative sampling fails, the collision-free arc will be further split. When the arc  $\varrho_\ell$  is sufficiently short, i.e.  $|\varrho_\ell| < \epsilon$ , it can be approximated by a sequence of three arcs. Specifically, the arc  $\varrho_\ell$  can be generated by velocity  $\mathbf{p}_\ell$  with time duration  $t_\ell$ . With the mode sufficiency assumption, any velocity  $\mathbf{p}_\ell$  can be positively decomposed into three primitive velocities  $\mathbf{p}_j^*$ , where  $j \in \{1, 2, 3\}$ , i.e.,  $\mathbf{p}_\ell = \sum_{j=1}^3 \lambda_j^* \mathbf{p}_j^*$ , for  $\lambda_j^* \geq 0$  and  $j \in \{1, 2, 3\}$ . Consider the sequence of arc motion  $(\mathbf{p}_1^*, t_1)(\mathbf{p}_2^*, t_2)(\mathbf{p}_3^*, t_3)$  that starts from  $s_\ell$ , where  $t_j = \lambda_j^* t$  is the duration of velocity  $\mathbf{p}_j^*$ . The Hausdorff distance between the trajectory sequential arc motion and the original arc  $\varrho_\ell$  can be proved to be bounded by  $\mathcal{O}(\epsilon)$ , as detailed in the supplementary material. As each primitive velocity has a feasible mode, this sequence of arc motion can be replaced by the original arc  $\varrho_\ell$  to form a feasible hybrid plan. Consequently, the search depth is bounded by  $|\bar{\mathfrak{S}}_m^k|/\epsilon$  with  $|\bar{\mathfrak{S}}_m^k|$  being the total length of the path segment, yielding that the termination condition is reached in finite steps. Thus, the algorithm is guaranteed to find a feasible solution.  $\square$

**Proof. of Theorem 1.** Under condition C2 in Table I, the collision-free timed paths  $\{\mathfrak{S}_m, \forall m \in \mathcal{M}\}$  can be found by MAPF algorithms. Alg. 1 decomposes the paths  $\{\mathfrak{S}_m, \forall m \in \mathcal{M}\}$  into  $\{\bar{\mathfrak{S}}_m\}$  that satisfies strict partial ordering, with a guarantee on convergence in Lemma 1. Then, the segments are assigned to the robots with a horizon of  $H$ , while ensuring that the partial ordering are respected, and furthermore those segments can be traversed without collisions by Lemma 2. Meanwhile, under the condition C3, each robot subgroup assigned to a subtask must satisfy the mode-sufficient condition, which guarantees that the hybrid strategy for each object is feasible by Lemma 3. Lastly, under the proposed motion controller, each robot can track any given arc motion of the object, and apply the required bounded forces. Thus, all subtasks can be accomplished, which in turns ensures that the overall pushing task for each object can be fulfilled.  $\square$