PushAround: Collaborative Path Clearing via Physics-Informed Hybrid Search

***Abstract*— In unstructured environments, the passage of large external vehicles is frequently blocked by movable obstacles, which motivates the deployment of mobile robot teams to proactively clear traversable corridors. Existing approaches to Navigation Among Movable Obstacles (NAMO) primarily plan sequences of obstacle manipulations but typically neglect physical feasibility, overlooking essential factors such as robot dimensions, mass, applied forces, and the coupled dynamics of pushing. This limitation often results in strategies that cannot be executed reliably in practice. A physics-informed framework for collaborative multi-robot pushing called PushAround is introduced, enabling the active construction of physically feasible paths. The framework jointly searches the sequence of obstacles to be displaced, the transit motions of robots between obstacles, and the contact points and forces required for effective execution. Core components include a W-Clearance Connectivity Graph (WCCG) for reasoning about vehicle passage under width constraints, a gap-ranking strategy for prioritizing obstacle-clearing actions, and a simulation-in-the-loop search to validate push feasibility. Extensive simulations and hardware experiments demonstrate robust success, scalability, and efficiency compared to baseline NAMO methods.**

I. Introduction

In many real-world scenarios, the path of a large vehicle or transport unit is obstructed by movable obstacles such as pallets, boxes, or equipment, which motivates the use of a fleet of mobile robots to actively clear traversable corridors and enable safe passage. This capability is especially critical in unstructured or cluttered workspaces, including warehouses, disaster sites, and crowded urban environments, where traditional path planning approaches assume static obstacles and therefore fail to provide feasible solutions when direct paths are blocked [1]. Research on Navigation Among Movable Obstacles (NAMO) has proposed strategies for pushing, pulling, or rotating objects to create new routes [2], but these methods often abstract away physical feasibility, ignoring key factors such as robot dimensions, obstacle size and mass, contact geometry, applied forces, and the coupled dynamics of pushing. As a result, the generated plans may be theoretically valid but physically unrealizable in practice. This problem is particularly challenging because a single large robot is often unable to independently clear a corridor; smaller robots are constrained by their limited size and reach, making some obstacle boundaries inaccessible; and pushing actions introduce strong physical coupling, where a single push may trigger chained motions of multiple obstacles, further complicating prediction and planning.

A. Related Work

Navigation Among Movable Obstacles (NAMO) has been studied as a core extension of motion planning in cluttered environments. Early approaches planned sequences of obstacle displacements through pushing, pulling, or rotating actions [2]. Heuristic search methods improved scalability by prioritizing which obstacles to move [3], and graph- or sampling-based algorithms enabled planning in larger workspaces [4]. Multi-agent variants have also been explored [5]. Despite these advances, most NAMO approaches neglect physical feasibility, typically treating robots as point agents with unlimited power and ignoring dimensions, object mass, contact geometry, and force constraints. As a result, many generated plans remain physically unrealizable [6].

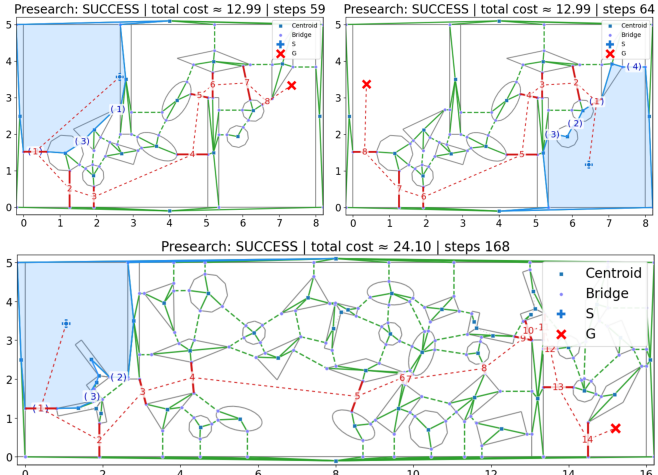

Fig. 1. **Frontier presearch and gap prioritization.** Each panel overlays the WCCG together with the currently selected face (light blue). Numbers in **red** mark the *gap-crossing order* returned by the presearch, and numbers in **blue** indicate the *local ranking* of first-hop gaps on the current face boundary. **Top left/right:** the same scene with start/goal swapped, results in symmetric gap orders with same cost (≈ 12.99). **Bottom:** a longer map with ~ 30 obstacles, showing a 14-gap sequence with cost ≈ 40.36.

Collaborative pushing has also been investigated in the context of multi-robot manipulation. Prior work has addressed synchronization of forces [7], stable contact maintenance [8], and cooperative transport of objects in structured settings [9]. These approaches demonstrate effective coordination but generally focus on a single object type and assume strict collision avoidance with surrounding obstacles. Such assumptions exclude the possibility of exploiting inter-object interactions, and the challenges of coupled dynamics and chain reactions remain largely unaddressed [10].

Physics-informed planning has recently emerged to incorporate realistic dynamics into motion planning. Some methods employ physics engines to validate candidate manipulations [11] or simulate contact dynamics during planning [12]. While these works highlight the benefits of embedding physics, they are mostly limited to single-object manipulation

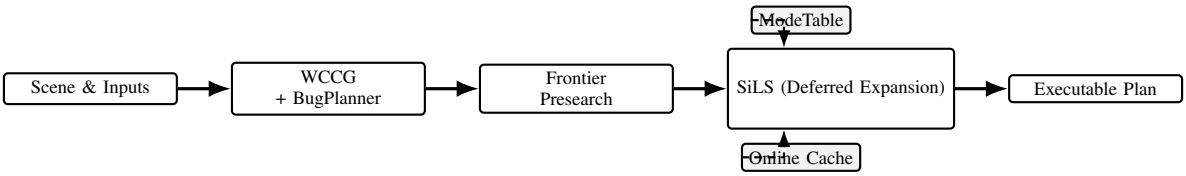

Fig. 2. TODO: Overall framework.

or grasp planning. In addition, many approaches decouple abstract planning from low-level physics checks [7], which reduces efficiency, and simulation-in-the-loop techniques face scalability challenges in multi-robot, cluttered scenarios [10]. A unified framework that couples collaborative planning with physics-based feasibility for robust path clearing has yet to be established.

B. Our Method

This work introduces a physics-informed framework called PushAround for collaborative multi-robot pushing that actively constructs traversable corridors for large vehicles in cluttered environments. The approach integrates three key components: a W-Clearance Connectivity Graph (WCCG) to represent workspace connectivity under vehicle width constraints and to identify blocking frontier gaps; a gap-ranking strategy that estimates the cost of clearing each gap and prioritizes obstacles for coordinated pushing; and a simulation-in-the-loop path construction scheme that incrementally expands a configuration-space search tree and validates each pushing mode through parallel physical simulation. This combination ensures that planned actions account for robot dimensions, contact geometry, applied forces, and chained obstacle motions, enabling physically feasible execution. The framework is evaluated through extensive 2D and large-scale simulations as well as hardware experiments, demonstrating robustness under various dense obstacle configurations, and varying team sizes.

The contributions of this work are twofold: (I) it presents the first unified framework that couples multi-robot collaborative pushing with simulation-in-the-loop planning to construct physically feasible paths in cluttered environments; (II) it demonstrates significant improvements in feasibility, efficiency, and scalability compared to existing NAMO and collaborative pushing approaches.

II. Problem Description

A. Workspace and Robots

The workspace is a bounded planar region $\mathcal{W} \subset \mathbb{R}^2$ that contains two types of obstacles. A set $\mathcal{O}$ represents immovable structures, while a set of M movable rigid polygons is defined as $\mathbf{\Omega} \triangleq \{\Omega_1, \cdots, \Omega_M\} \subset \mathcal{W}$. Each movable obstacle Ω_m is characterized by its mass M_m, inertia I_m, frictional parameters (either identified or estimated), and state $\mathbf{s}_m(t) \triangleq (\mathbf{x}_m(t), \psi_m(t))$, where $\mathbf{x}_m(t) \in \mathbb{R}^2$ is the planar position of its centroid and $\psi_m(t) \in \mathbb{R}$ its orientation angle. The region occupied by Ω_m at time t is denoted $\Omega_m(t)$. Moreover, a small team of N robots, indexed as $\mathcal{R} \triangleq \{R_1, \cdots, R_N\}$, operates as a cooperative unit. Each robot R_i is modeled as a rigid body with state $\mathbf{s}_{R_i}(t) \triangleq (\mathbf{x}_{R_i}(t), \psi_{R_i}(t))$, where $\mathbf{x}_{R_i}(t) \in \mathbb{R}^2$ is its position and $\psi_{R_i}(t) \in \mathbb{R}$ its orientation. The footprint of R_i is denoted $R_i(t)$. The instantaneous free space is given by:

$$\widehat{\mathcal{W}}(t) \triangleq \mathcal{W} \backslash \Big(\mathcal{O} \cup \{\Omega_m(t)\}_{m=1}^{M} \cup \{R_i(t)\}_{i=1}^{N} \Big), \quad (1)$$

where $\widehat{\mathcal{W}}(t)$ excludes regions occupied by immovable obstacles, movable obstacles, and robots.

B. External Vehicle and Clearance Goal

An external vehicle $\mathtt{V}$ of radius $W/2 > 0$ must navigate within the workspace from a start configuration $\mathbf{s}_{\mathtt{V}}^{\mathtt{S}}$ to a goal configuration $\mathbf{s}_{\mathtt{V}}^{\mathtt{G}}$. Since movable obstacles may obstruct the way, a direct passage is not always feasible. Feasibility at time t is captured by the W-clearance condition

$$\exists\, \mathcal{P}_{\mathtt{V}}^{W} \subset \widehat{\mathcal{W}}(t) : \ \mathbf{s}_{\mathtt{V}}^{\mathtt{S}} \rightsquigarrow \mathbf{s}_{\mathtt{V}}^{\mathtt{G}}, \ \mathtt{clr}(\mathcal{P}_{\mathtt{V}}^{W}) \geqslant W, \quad (2)$$

where $\mathcal{P}_{\mathtt{V}}^{W}$ is a continuous curve connecting start and goal inside the free space, and $\mathtt{clr}(\mathcal{P}_{\mathtt{V}}^{W})$ denotes its minimum clearance to surrounding obstacles.

C. Collaborative Pushing Modes

Thus, the robots may actively reconfigure $\mathbf{\Omega}$ by pushing obstacles. The interaction with obstacle Ω_m is described by a pushing mode $\boldsymbol{\xi}_m \triangleq (\mathcal{C}_m, \mathbf{u}_m)$, where $\mathcal{C}_m \in (\partial\Omega_m)^N$ is the set of contact points established by the robots, and $\mathbf{u}_m \in \mathbb{R}^{2N}$ encodes the body-frame forces or an equivalent wrench profile. The admissible set of modes is Ξ_m, determined by contact geometry and frictional limits. Furthermore, the system evolution under a pushing mode is captured by the transition operator below:

$$\mathbf{S}(t^+) \triangleq \Phi\big(\mathbf{S}(t),\, m,\, \boldsymbol{\xi}_m\big), \quad (3)$$

where $\mathbf{S}(t)$ stacks all robot and obstacle states. The operator Φ models the joint dynamics of robots and obstacles under contact. In general, Φ is not available in closed form and is instead evaluated through physics simulation.

D. Problem Statement

The goal is to compute a hybrid schedule for the robots that reconfigures the movable obstacles so that the vehicle admits a W-feasible path from start to goal. The overall schedule for the robotic fleet is defined as:

$$\pi \triangleq \big\{(m_k, \boldsymbol{\xi}_k, \Delta t_k)\big\}_{k=1}^{K}, \quad (4)$$

where $\Omega_{m_k} \in \mathbf{\Omega}$ specifies the movable obstacle to manipulate, $\boldsymbol{\xi}_k \in \Xi_{m_k}$ is the pushing mode, and $\Delta t_k > 0$

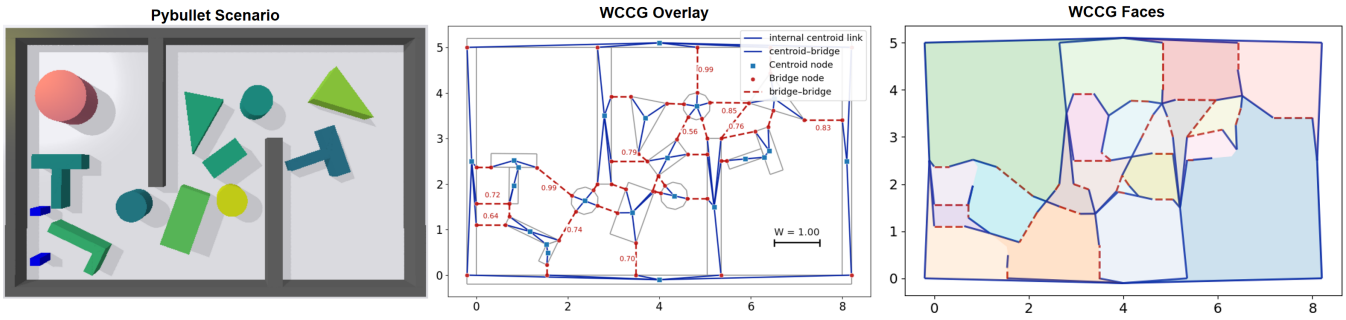

Fig. 3. Illustration of the W–Clearance Connectivity Graph (WCCG). **Left:** Cluttered PyBullet scenario with the immovable walls and movable objects; **Middle:** WCCG overlay with the centroid nodes (blue squares), bridge nodes (red circles), centroid–bridge edges (blue), and bridge–bridge edges (red dashed) annotated by the gap widths; **Right:** Induced faces of the WCCG, where the colors indicate distinct connected regions.

the duration of execution. Thus, the optimization problem balances the execution time and the physical effort of the clearance process, subject to various constraints, i.e.,

$$\min_{\pi} \left\{ T + \alpha \sum_{k=1}^{K} J(m_k, \boldsymbol{\xi}_k, \mathbf{S}(\tau_k)) \right\}, \tag{5}$$

where $T \triangleq \sum_{k=1}^{K} \Delta t_k$ is the total task duration, $\alpha > 0$ is a trade-off parameter, and $J(\cdot)$ is a simulation-based control effort or feasibility cost evaluated at state $\mathbf{S}(\tau_k)$. Note that the above optimization problem is constrained by (1), (2), and (3), which jointly ensure collision-free evolution within the workspace, valid dynamic transitions under pushing, and a terminal W-clearance path for the vehicle.

Remark 1. Problem (5) above uniquely couples obstacle selection, pushing modes, and timing into a single hybrid optimization. This yields a combinatorial search space far more complex than classical NAMO, where prior work often assumes simple object shapes or hand-crafted contact modes [13], [14], [15], [16]. ■

III. Proposed Solution

The unified framework called PushAround is developed to enable collaborative multi-robot pushing with physical feasibility, as described in this section. The framework begins by constructing a W-Clearance Connectivity Graph, introduced in Sec. III-A, which captures the connectivity of the workspace under the clearance requirement and identifies frontier gaps that block passage. Building on this representation, Sec. III-B presents a gap-ranking strategy that assigns costs to candidate gaps and determines which obstacle-clearing actions are most promising. These ranked actions are then evaluated within a simulation-in-the-loop search, described in Sec. III-C, where a configuration-space tree is expanded and candidate pushes are validated by parallel physical simulation. The complete execution flow together with generalization aspects is summarized in Sec. III-D.

A. W–Clearance Connectivity Graph

The feasibility of routing the vehicle depends on whether a corridor of width W exists between the start configuration $\mathbf{s}_{\mathrm{V}}^{\mathrm{S}}$ and the goal configuration $\mathbf{s}_{\mathrm{V}}^{\mathrm{G}}$. To address this question, a *W–Clearance Connectivity Graph* (WCCG) is introduced. The WCCG encodes adjacency relations between obstacles under the clearance threshold W and supports efficient connectivity queries. Unlike grid maps or sampled roadmaps, it is constructed directly from obstacle geometry, avoiding discretization errors and ensuring consistent results. This is particularly important since clearance queries must be invoked repeatedly during planning.

1) Graph Construction: The WCCG is built by decomposing each movable obstacle $\Omega_m \in \boldsymbol{\Omega}$ into convex components. From each component C, a centroid node v_c is created. When two components C_u and C_v have closest points $p_u \in C_u$ and $p_v \in C_v$ with distance smaller than W, bridge nodes are added at p_u and p_v. These bridge nodes are connected by a bridge–bridge edge, annotated with the corresponding gap width $w_{uv} \triangleq \|p_u - p_v\|$, and further linked back to their centroids with centroid–bridge edges. Narrow passages with $w_{uv} < W$ are explicitly marked as potential bottlenecks. The resulting graph is defined below:

$$\mathcal{G}_W \triangleq (\mathcal{V},\ \mathcal{E}_c \cup \mathcal{E}_b), \tag{6}$$

where $\mathcal{V}$ contains all centroid and bridge nodes; $\mathcal{E}_c$ is the set of centroid–bridge edges; and $\mathcal{E}_b$ the set of bridge–bridge edges annotated by widths w_{uv}.

2) Connectivity Criterion: Once $\mathcal{G}_W$ has been constructed, connectivity queries can be performed without explicitly computing a geometric path. A frontier-tracing procedure, similar to the BugPlanner [17], starts from the vehicle start $\mathbf{s}_{\mathrm{V}}^{\mathrm{S}}$, casts a ray toward the goal $\mathbf{s}_{\mathrm{V}}^{\mathrm{G}}$, and explores the encountered loop of frontier edges. If a valid exit is discovered, the process continues until the goal is reached; otherwise a blocking cycle is returned. Successful execution produces a skeleton Σ, which is an ordered sequence of centroid and bridge nodes that certifies $\mathbf{s}_{\mathrm{V}}^{\mathrm{S}}$ and $\mathbf{s}_{\mathrm{V}}^{\mathrm{G}}$ lie in the same connected face. Let the W–clear free space be $\mathcal{F}_W \triangleq \mathbb{R}^2 \backslash (\mathcal{O} \oplus \mathbb{B}_{W/2})$, where $\oplus$ denotes Minkowski addition and $\mathbb{B}_{W/2}$ is a closed disk of radius $W/2$. A complete criterion for existence of a W–clear path $\mathcal{P}_{\mathrm{V}}^{W}$ is defined as:

$$\exists\, \mathcal{P}_{\mathrm{V}}^{W} \subset \mathcal{F}_W :\ \mathbf{s}_{\mathrm{V}}^{\mathrm{S}} \rightsquigarrow \mathbf{s}_{\mathrm{V}}^{\mathrm{G}} \iff \Big\{ \mathbf{s}_{\mathrm{V}}^{\mathrm{S}}, \mathbf{s}_{\mathrm{V}}^{\mathrm{G}} \text{ belong to the same face of } \mathcal{G}_W;\ \mathbb{B}_{W/2}(\mathbf{s}_{\mathrm{V}}^{\mathrm{S}}),\ \mathbb{B}_{W/2}(\mathbf{s}_{\mathrm{V}}^{\mathrm{G}}) \subset \mathcal{F}_W \Big\}, \tag{7}$$

where $\mathbb{B}_{W/2}(\cdot)$ denotes a disk of radius $W/2$ centered at the argument. They align with (2) for the external vehicle.

3) Skeleton to Path: The skeleton Σ as an output of the previous step, is an ordered sequence of centroid and bridge nodes connected by frontier edges in $\mathcal{G}_W$. This skeleton serves as a compact certificate that $\mathbf{s}_{\mathrm{V}}^{\mathrm{S}}$ and $\mathbf{s}_{\mathrm{V}}^{\mathrm{G}}$ lie in the same connected face. Given as input the skeleton Σ, together with $\mathcal{G}_W$, the clearance W, and the endpoint disks $\mathbb{B}_{W/2}(\mathbf{s}_{\mathrm{V}}^{\mathrm{S}})$ and $\mathbb{B}_{W/2}(\mathbf{s}_{\mathrm{V}}^{\mathrm{G}})$, the output is an explicit W–clear path $\mathcal{P}_{\mathrm{V}}^{W}$. This path is constructed by sliding each skeleton segment along the boundary of the inflated obstacles, offsetting slightly inward into $\mathcal{F}_W$, and attaching short connectors inside the endpoint disks. The resulting $\mathcal{P}_{\mathrm{V}}^{W}$ remains in $\mathcal{F}_W$, preserves the homotopy of Σ, and guarantees clearance of at least W.

Remark 2. The proposed WCCG differs from sampling- and grid-based planners in two key aspects: (I) It avoids discretization of $\mathcal{W}$ and is therefore free from resolution-induced errors; (II) It relies purely on geometry, which makes queries highly efficient. These two features are crucial, since connectivity checks and clearance tests are invoked many times within the hybrid planner described in the sequel. ■

B. Ranking of Potential Blocking Gaps

When the condition in (7) fails, the vehicle cannot reach $\mathbf{s}_{\mathrm{V}}^{\mathrm{G}}$ from $\mathbf{s}_{\mathrm{V}}^{\mathrm{S}}$ through a W–clear path $\mathcal{P}_{\mathrm{V}}^{W}$. In this case, the planner must prioritize *blocking gaps* on the reachable frontier of $\mathcal{G}_W$. The goal of this module is to provide an ordered list of such gaps, ranked by their predicted cost to eventually yield a feasible path, which serves as a critical guidance for the hybrid search in the sequel.

1) Frontier Extraction and Candidate Gaps: A BugPlanner query on $\mathcal{G}_W$, given $(\mathbf{s}_{\mathrm{V}}^{\mathrm{S}}, \mathbf{s}_{\mathrm{V}}^{\mathrm{G}}, W)$, either confirms connectivity or returns a counter–clockwise frontier loop $\mathcal{L}$ that separates the two endpoints. The bridge–bridge edges visible on $\mathcal{L}$ form the first–hop candidate set $\Gamma_{\mathcal{L}} \triangleq \{g_1, \cdots, g_K\}$, where each g_k is a reachable candidate gap. These candidates are the inputs to the ranking module, while the output will be an ordered list of the same set sorted by predicted cost.

2) Evaluation for Immediate Cost: Each candidate $g \in \Gamma_{\mathcal{L}}$ is evaluated by combining the cost for the robots to reach the gap and the effort required to widen it. Let $\mathbf{s}_{\mathcal{R}}$ be the current robot positions, and $\mathbf{o}_g$ as the outside insertion point of gap g. The resulting one–hop cost is given by:

$$\mathsf{C}\big(g \mid \mathcal{L}, \mathbf{s}_{\mathcal{R}}\big) = \lambda_{\mathrm{t}}\, \mathsf{C}_{\mathrm{t}}(\mathbf{s}_{\mathcal{R}}, \mathbf{o}_g) + \lambda_{\mathrm{p}}\, \mathsf{C}_{\mathrm{p}}(g), \tag{8}$$

where $\lambda_{\mathrm{t}}, \lambda_{\mathrm{p}} > 0$ are weighting factors for the transition and pushing costs, respectively; function $\mathsf{C}_{\mathrm{t}}(\cdot)$ denotes the collision-free distance between two points; and the function $\mathsf{C}_{\mathrm{p}}(\cdot)$ measures the widening effort, which increases when the gap is narrower than W, or when the adjacent obstacles are heavier as scaled by the mass of adjacent movable obstacles.

3) Long-term Cost w.r.t. Goal: Furthermore, to prioritize gaps closer to the goal, a heuristic is added to the evaluation, i.e., $h(g) = \eta \left\| \mathbf{o}(g) - \mathbf{s}_{\mathrm{V}}^{\mathrm{G}} \right\|$, where $\eta > 0$ is a scaling constant. Lastly, a short A*-style presearch virtually crosses each candidate gap, recomputes the next frontier, and continues

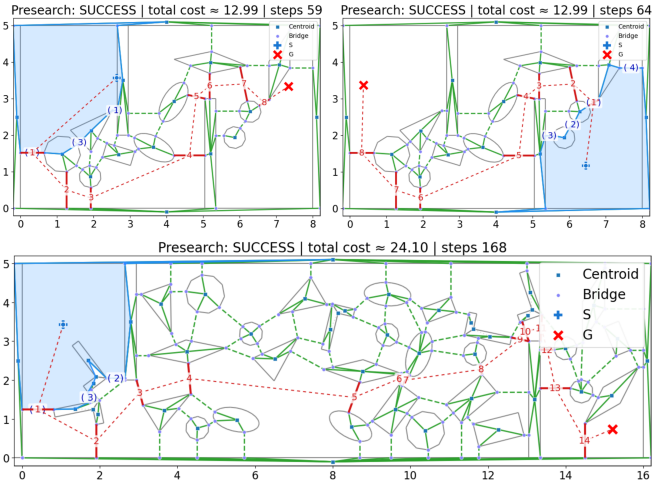

Fig. 4. Illustration for the ranking of potential gaps. Each panel overlays the WCCG together with the currently selected face (light blue). The algorithm (i) extracts the frontier loop from BugPlanner, (ii) enumerates candidate bridge–bridge gaps on the loop, (iii) assigns local first-hop ranks (blue numbers), and (iv) simulates a short presearch to predict the full gap-crossing sequence (red numbers). **Top:** identical environment with start and goal swapped; the resulting gap sequences are symmetric with identical predicted cost at 12.99; **Bottom:** a larger map with about 30 obstacles, where presearch returns a 14-gap sequence of predicted cost at 24.10.

for a limited beam width and depth. The predicted cost-to-connect is given by:

$$\widehat{\mathsf{Cost}}(g) \triangleq \mathsf{C}\big(g \mid \mathcal{L}, \mathbf{s}_{\mathcal{R}}\big) + \sum_{g' \in \Pi^\star(g)} \big\{ \mathsf{C}(g' \mid \cdot) + h(g') \big\}, \tag{9}$$

where $\Pi^\star(g)$ is the sequence of subsequent gaps discovered after virtually crossing g; the dots indicate updated inputs along that rollout; and the scalar score $\widehat{\mathsf{Cost}}(g) > 0$ for each candidate gap. Consequently, the final output of this module is the ranked list of candidate gaps, i.e., $\mathsf{Rank}(\Gamma_{\mathcal{L}}) \triangleq \mathbf{argsort}_{g \in \Gamma_{\mathcal{L}}} \widehat{\mathsf{Cost}}(g)$, which sorts $\Gamma_{\mathcal{L}}$ in ascending predicted cost. This ordered set is passed to the hybrid search module, such that only the promising gaps are expanded first.

Remark 3 (Practical Improvement)**.** Efficiency of the above ranking procedure can be improved by caching frontier loops, storing the transitions $(\mathcal{L}, g) \mapsto \mathcal{L}'$, and accelerating the edge queries with axis-aligned bounding-box culling. With a small beam width and depth (typically around 8), the runtime of ranking remains negligible compared to the cost of simulation-based validation. ■

C. Physics-Informed Hybrid Search

The hybrid search couples high-level decisions about blocking gaps with low-level feasibility of multi-robot pushing. Unlike purely geometric planners, this procedure simultaneously determines a sequence of gaps to clear and physically feasible pushing actions, including directions, contact modes, and forces. Parallel physics simulation is embedded so that many candidate push strategies can be evaluated simultaneously at each expansion, and the resulting successor states are returned to the search. This tight coupling of discrete graph reasoning with continuous pushing dynamics is unique for the considered problem.

Algorithm 1: Physics-Informed Hybrid Search

Input: $\mathbf{s}_0$, α, $\mathsf{EvalSim}(\cdot)$, W–criterion by (7)
Output: $\pi^\star$

/* Initialization */
$\nu_0 \leftarrow (\mathbf{s}_0, \varnothing)$, $\chi(\nu_0) = 0$;
Init $\mathcal{Q}$ by (10);
while *not terminated* **do**
  /* Selection */
  $\nu \leftarrow \mathcal{Q}.\texttt{pop\_min}()$ by (10);
  /* Parallel Expansion */
  foreach $g \in \mathsf{Rank}(\nu)$ **do**
    foreach $(\mathbf{v}, \boldsymbol{\xi}) \in \Xi_g$ **do**
      $\tau = (g, \mathbf{v}, \boldsymbol{\xi})$;
      $(\mathbf{s}', \delta_T, \delta_J) \leftarrow \mathsf{EvalSim}(\nu, \tau)$ by (11);
      if *success* **then**
        $\nu' \leftarrow (\mathbf{s}', \pi \cup \{\tau\})$;
        $\chi(\nu') \leftarrow \chi(\nu) + \delta_{\mathrm{T}} + \alpha\, \delta_{\mathrm{J}}$;
        $\mathcal{Q}.\texttt{push}(\nu')$;
  /* Termination Check */
  if $\mathbf{s}'$ *admits* $\mathcal{P}_V^W$ *by* (7) **then**
    $\pi^\star \leftarrow \pi'$, **break**;
return $\pi^\star$;

1) Tree and Initialization: The search tree $\mathcal{T}$ is composed of nodes $\nu \triangleq (\mathbf{s}, \pi)$, where $\mathbf{s}$ is the current system state including the positions and orientations of all robots and movable obstacles as in (3), and π is the partial pushing strategy realized so far as in (4). Two global functions are maintained: $\mathsf{Rank}(\nu)$ stores a ranked list of candidate gaps at this node together with their exploration status, derived from (9); and $\chi(\nu)$ returns the cumulative execution cost from the root to ν. The root is initialized as $\nu_0 \triangleq (\mathbf{s}_0, \varnothing)$, where $\mathbf{S}^{\mathrm{S}}$ corresponds to the initial system state $\mathbf{s}_0$. At initialization, $\chi(\nu_0) = 0$, and $\mathsf{Rank}(\nu_0)$ is constructed by first computing the frontier loop $\mathcal{L}_\nu$ associated with $\mathbf{s}_0$, then extracting the visible gaps $\Gamma_{\mathcal{L}_\nu}$, and finally ranking them by their predicted cost-to-connect.

2) Node Selection: At each iteration, the node with minimum best-first priority is selected from the queue. The priority function balances the realized execution cost and the estimated effort of remaining gaps:

$$f(\nu) \triangleq \chi(\nu) + \min_{g \in \mathsf{Rank}(\nu)} \widehat{\mathsf{Cost}}(g), \tag{10}$$

where $\chi(\nu)$ is the cumulative execution cost so far, and the second term is the minimum predicted remaining cost among the unexplored gaps of ν. This scoring ensures that nodes are expanded in an order that jointly accounts for physical effort already incurred and the most promising future actions.

3) Node Expansion with Parallel Simulation: When a node ν is selected for expansion, a batch of *pushing strategies* from $\mathsf{Rank}(\nu)$ is evaluated in parallel by simulation. Each pushing strategy is defined as $\tau \triangleq (g, \mathbf{v}, \boldsymbol{\xi})$, where $g \in \boldsymbol{\Omega}$ is the chosen gap; $\mathbf{v} \triangleq (v_x, v_y, \omega) \in \mathbb{R}^3$ is a short-horizon body velocity for the manipulated obstacle; and $\boldsymbol{\xi} \triangleq (\mathcal{C}_m, \mathbf{u}_m) \in (\partial\Omega_m)^N \times \mathbb{R}^{2N}$ is a contact mode specifying robot contact points and pushing forces as in (3). Moreover, each candidate τ is first checked by a geometric quick-pass. If it passes, it is evaluated by the simulator with the current state and pushing strategy:

$$(\mathbf{s}',\, \delta_{\mathrm{T}},\, \delta_{\mathrm{J}}) \triangleq \mathsf{EvalSim}\big(\nu,\, (g, \mathbf{v}, \boldsymbol{\xi})\big), \tag{11}$$

where $\mathbf{s}'$ is the successor state; $\delta_{\mathrm{T}} > 0$ is the time cost; and δ_{J} is the control effort as in (5). Thus, a successful evaluation produces a child $\nu' \triangleq (\mathbf{s}', \pi')$, where π' is obtained by appending π with τ. The incremental realized cost of τ is $\widehat{\mathsf{C}}(\nu, \tau) \triangleq \delta_{\mathrm{T}} + \alpha\, \delta_{\mathrm{J}}$, with $\alpha > 0$ the same trade-off parameter as in (5). The cumulative cost is then updated by $\chi(\nu') \triangleq \chi(\nu) + \widehat{\mathsf{C}}(\nu, \tau)$. Since many pushing strategies are simulated concurrently, numerous successors can be expanded in parallel at each iteration.

4) Termination: Termination occurs when a node ν reaches a state $\mathbf{s}$ for which a W–clear path $\mathcal{P}_{\mathrm{V}}^W$ exists from $\mathbf{s}_{\mathrm{V}}^{\mathrm{S}}$ to $\mathbf{s}_{\mathrm{V}}^{\mathrm{G}}$, as certified by (7). In this case, the schedule π stored in ν constitutes a complete solution to (5), encoding both the sequence of gaps to clear and the physically feasible pushing actions that realize them.

Remark 4 (Parallel Expansion and Mode Reuse)**.** Efficiency of the hybrid search stems primarily from parallel evaluation of pushing strategies, which allows many candidate futures to be simulated at once for each expansion. Deferred expansion ensures that nodes with long candidate lists remain in the priority queue until all tasks are attempted, preventing search starvation. In addition, validated $(\boldsymbol{\xi}, \mathbf{v})$ pairs are cached locally within a subtree and stored in a persistent ModeTable for reuse across similar obstacles, significantly reducing repeated physics calls. These mechanisms yield orders-of-magnitude speedup relative to naive simulation-based search. ■

D. Overall Analyses

1) Online Execution and Adaptation: After the search returns a push sequence $\texttt{Plan} = \{\tau_1, \ldots, \tau_K\}$, execution proceeds *sequentially* with an online hybrid controller that alternates between **transition** (robots move to contact) and **pushing** (robots regulate while the object moves).

Controller overview. Each task τ_k specifies a target obstacle and a world-frame twist $\hat{\mathbf{u}} = (\hat{v}_x, \hat{v}_y, \hat{\omega})$ for a short horizon. We synthesize a reference object trajectory by integrating $\hat{\mathbf{u}}$ and map it to per-robot reference contact states using the active contact mode. Robots are commanded by proportional velocity control on position/yaw errors at their *contact-centric* targets:

$$\mathbf{v}_n = K_p(\hat{\mathbf{p}}_n^{\mathrm{c}} - \mathbf{p}_n^{\mathrm{c}}), \qquad \omega_n = K_r\left(\hat{\psi}_n^{\mathrm{c}} - \psi_n^{\mathrm{c}}\right),$$

with periodic *forward-reference refresh* to remain predictive. During transition, each robot plans a collision-free path (grid/A*) to a small offset of its contact pose; a lightweight conflict check swaps a pair of goal contacts if two transition paths are imminently head-on. During pushing, small contact-frame offsets are adapted online to absorb object yaw drift, keeping contact consistent without re-planning.

Task switching and re-planning. Execution monitors (i) transition feasibility/timeouts and (ii) pushing progress via a short-horizon early-stop test that triggers when the commanded widening succeeds or progress stalls. On task completion, the executor advances to τ_{k+1}. If a failure/stall is detected, execution yields control back to the planner with the current snapshot. Previously explored but unexecuted nodes are *preserved* and reinserted by priority, avoiding priority-queue exhaustion and continuing from the most promising frontier.

Global goal check. At a coarse cadence, the system rebuilds the W-clearance graph on the live snapshot and terminates the episode as soon as a W-clear path exists from start to goal; this prevents unnecessary additional pushes once connectivity is achieved.

Remark 5 (Endpoint disks as a sufficiency booster)**.** The W-CCG connectivity test is necessary but may be conservative at the path endpoints. We therefore include a lightweight *endpoint-clearing* supplement: if connectivity holds but either endpoint's $W/2$ disk is contaminated, the planner proposes a small set of *away-from-endpoint* pushes to clear the disk, after which execution proceeds. This is an implementation convenience rather than a core component of the method.

2) Complexity (Core Conclusions): Per expansion, the dominant cost is parallel prediction of a small batch of push candidates; frontier/W-connectivity and gap presearch are typically subdominant.

$$T_{\text{exp}} \approx \tilde{\mathcal{O}}(M) + \mathcal{O}(BD) + \frac{K}{P}(T_{\text{qp}} + S\,T_{\text{phys}}),$$

where M is #obstacles, B/D are beam width/depth for presearch, K candidates per expansion, P worker processes, S average physics steps (post quick-pass/early-stop), T_{qp} geometry screen time, and T_{phys} per-step physics time.

Implications. (i) ModeTable and beam presearch reduce K and keep $BD \ll M$; (ii) quick-pass/early-stop cut S and the simulator hit-rate; (iii) increasing P yields near-linear speedup until IPC bounds; (iv) deferred reinsertion keeps expansions focused, avoiding wasted restarts.

3) Generalization and limits: **Heterogeneous teams.** The controller already exposes per-robot gains and supports basic heterogeneity during transition (e.g., avoiding goal swaps across mismatched capability classes). Extending the ModeTable prior with robot-specific limits is straightforward. **Sequential vs. concurrent pushes.** The current executor applies one push task at a time; concurrent pushing is a natural extension but requires multi-object mode synthesis and conflict-aware transitions. **Dynamic changes.** Because the executor performs periodic global W-connectivity checks and re-plans on failure, modest perturbations (e.g., small drifts or unmodeled contacts) are handled online without restarting the episode.

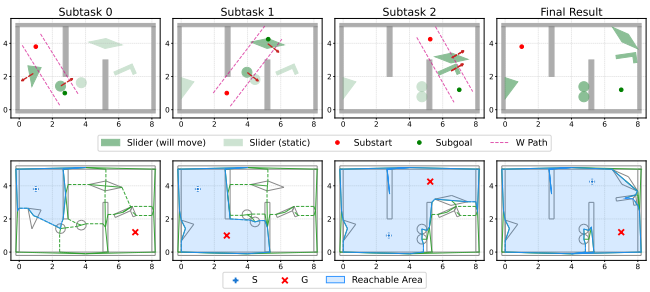

Fig. 5. SL-Push. Compute(offline) or simulate(sim-in-the-loop) the pushing directions for movable obstacles along the W-width straight path from sub-start to sub-goal. Top: Pushing steps. Bottom: Reachable region changes during pushing

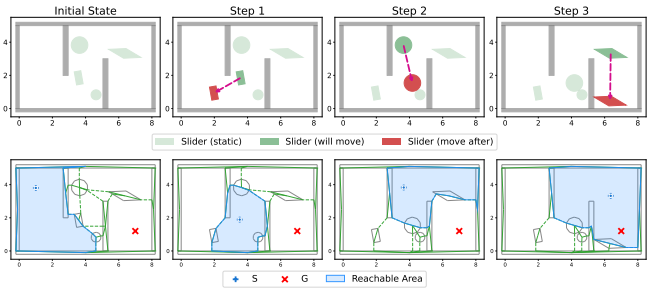

Fig. 6. Rec-NAMO. Rec-NAMO builds path segments sequentially but fails to construct the full W-clear path from start to goal. Top: Pushing steps. Bottom: Reachable region changes during pushing

IV. Numerical Experiments

We evaluate the proposed simulation-in-the-loop NAMO planner (SiLS) in cluttered environments with both movable and immovable obstacles. All components (W-CCG presearch, frontier extraction, and ModeTable prior) are integrated as described in Sec. III. The implementation is in `Python 3` and simulations are run in `PyBullet` [18] on a laptop with an Intel Core i7–1280P CPU. Videos and logs are provided in the supplementary material.

A. Numerical Simulations

1) Setup: The physics step is $\Delta t = 1/120\,\text{s}$ and the control period is $1/40\,\text{s}$. Robots are disk/box pushers with risk radii consistent with the W-clearance definition. Movable obstacles have masses uniformly sampled in $[5, 15]\,\text{kg}$; immovables are modeled with mass 0. A trial succeeds when a W-clear path from start to goal exists and the target reaches its goal disc.

Workspace and scenarios. We use a *nominal scenario* with an $8\times5\,\text{m}$ bounded workspace, two internal bars (bottlenecks), and a mixed pool of movable shapes (curved and polygonal, including rings, ellipses, X/T/L/diamond, arrow-like, rectangles, and cylinders). Two robots start in the lower left; the target goal lies on the right side. Movables are randomly placed with a minimum separation. We test 10–15 objects over multiple seeds.

Geometry and caching. Collision checks use polygon/curve-edge models with convex decomposition

when needed. Ray–edge queries for frontiers are vectorized with AABB culling. Frontier loops are cached by signatures and transitions $(\mathcal{L}, g) \mapsto \mathcal{L}'$ are memoized.

2) Algorithm Configuration: Unless stated otherwise: per-task simulation horizon $= 80$ steps, up to 64 candidate push tasks per expansion, and priority $f = g + \widehat{\mathsf{Cost}}_{\text{to-go}}$ with heuristic factor 10. Gap sampling uses a softmax with temperature 0.05. ModeTable is enabled (auto-baked if missing). A *quick-pass* geometry screen may skip physics if a reference rollout already clears the gap; otherwise a short-horizon simulation with early stop is used. To avoid premature termination, a *deferred-reinsertion* rule keeps high-value but temporarily unexpanded nodes in the queue and revisits them later.

3) Baselines: We compare against three representative families, all using the same W-clearance criterion and contact models:

DFS-WCCG: Simulation-in-the-loop depth-first search that shares our physics predictor and W–CCG for goal checking. Each node is a snapshot; actions are four fixed axis-aligned pushes $!\leftarrow, \rightarrow, \uparrow, \downarrow!$ applied to any movable object (branching $\leqslant 4n$ for n objects).

SL-Push: Straight-line (or waypointed) route; blockers are cleared by (i) off-line minimal normal displacements or (ii) sim-in-the-loop normal pushes from near to far.

Rec-NAMO: Recursive routing/pushing on a cost-weighted visibility graph: Dijkstra for routing, push-decomposition for local clearing; failures prune edges and replan.

4) Metrics: We report success rate, wall-clock time, number of node expansions, number of simulated pushes, cumulative push time, average presearch cost-to-go, quick-pass ratio, early-stop ratio, and worker-pool utilization. Results are averaged over 10 seeds unless noted.

5) Comparison of Main Results: Table I summarizes the performance of four baseline methods and our proposed SiLS method on the nominal scenario. DFS-WCCG exhibits high computational overhead due to the exponential growth of the search space with the number of movable obstacles, resulting in very long planning times and frequent timeouts, which is reflected in its lowest success rate. SL-Push (offline) achieves the fastest planning times, typically under 0.1 s, but the lack of physics-informed simulation leads to physically infeasible solutions. The simulation-in-the-loop type of SL-Push improves solution feasibility; however, it still requires many simulation calls, resulting in increased planning times. Furthermore, the straight-line pushing strategy may generate unnecessarily long action sequences, as the direct path is not always the minimal pushing path. Figure 5 illustrates the step-by-step execution of SL-Push with two manually defined subgoals, showing how the environment gradually becomes connected such that the start and goal are in the same connected face.

Rec-NAMO, while faster than simulation-in-the-loop approaches and effective in classical NAMO tasks, struggles to generate fully connected W-clear paths in our setting. Although its recursive search produces locally connected regions along the pushing sequence, the final map often fails to contain a continuous W-clear path, as demonstrated in Figure 6. This limitation is reflected in its relatively low success rate in Table I.

TABLE I

PERFORMANCE ON THE NOMINAL SCENARIO WITH PUSH-COUNT (MEAN $\pm$ STD).

Method	Succ. (%)	#PT (s)	#ET (s)	#Sims	#Pushes
DFS-WCCG	25	timeout	N/A	-	-
SL-Push (off-line)	62.5	0.02	145.147	0	7
SL-Push (sim)	75	25.09	246.825	16	8
Rec-NAMO	37.5	10.45	179.262	0	7
SiLS (ours)	–	–	–	–	–

Metrics. *Succ.* = success rate; *#PT* = planning time; *#ET* = execution time; *#Sims* = simulations invoked; *#Pushes* = length of executed push sequence.

TABLE II

ABLATIONS ON THE NOMINAL SCENARIO (FILL WITH MEASUREMENTS).

Variant	Succ. (%)	Time (s)	#Sims	Quick-pass (%)
SiLS (full)	–	–	–	–
w/o presearch	–	–	–	–
w/o ModeTable	–	–	–	–
w/o quick-pass/ES	–	–	–	–
w/o reinsertion	–	–	–	–

Our SiLS method attains higher success rates with fewer simulation calls and shorter planning and execution times, thanks to (i) frontier-based gap ranking, (ii) ModeTable-guided push directions, and (iii) quick-pass/early-stop. In addition, deferred reinsertion prevents priority-queue starvation by revisiting previously generated high-value nodes when a batch of actions fails, further improving efficiency and solution quality.

6) Ablations: We remove one component at a time: (i) W-CCG presearch (random gap order), (ii) ModeTable prior (fallback "away + jitter" only), (iii) quick-pass/early-stop (always full physics), (iv) deferred reinsertion (terminate on empty queue). Table II shows consistent drops in success and increased time/#Sims when any component is disabled.

7) Efficiency and Scalability: **Frontier caching** makes BugPlanner near-linear in visible edges. **Parallel prediction** uses a persistent worker pool with per-round broadcast of snapshots and reachable-contact sets to reduce IPC. **Quick-pass** and **early-stop** skip or truncate physics when geometry suffices. **Deferred reinsertion** maintains steady depth growth even when many pushes are rejected.

8) Qualitative Results: Figure **??** shows a typical run: frontiers and ranked gaps, a top-ranked gap sequence, simulated pushes that widen bottlenecks, and the final W-clearance path. Per-step overlays and GIFs are produced by a lightweight snapshot logger.

9) Reproducibility: Random seeds are fixed, JSONL snapshots are logged, and the driver and scenario generator are released. ModeTable entries are auto-generated if absent to ensure repeatability across machines.

V. Conclusion

This work introduces a multi-robot approach for path clearing in unstructured environments, utilizing a hybrid search algorithm to plan and execute the sequence of obstacles, contact points, and forces efficiently. The framework demonstrates real-time adaptability to dynamic scenarios. Future work will address uncertainties such as estimating obstacle positions without external monitoring, handling uncertain obstacle masses, and managing partial robot visibility to improve robustness and performance in real-world, dynamic environments.

References

[1] L. Liu, X. Wang, X. Yang, H. Liu, J. Li, and P. Wang, "Path planning techniques for mobile robots: Review and prospect," *Expert Systems with Applications*, vol. 227, p. 120254, 2023.

[2] M. Stilman and J. J. Kuffner, "Navigation among movable obstacles: Real-time reasoning in complex environments," *International Journal of Humanoid Robotics*, vol. 2, no. 04, pp. 479–503, 2005.

[3] M. Stilman, J.-U. Schamburek, J. Kuffner, and T. Asfour, "Manipulation planning among movable obstacles," in *Proceedings 2007 IEEE international conference on robotics and automation*. IEEE, 2007, pp. 3327–3332.

[4] L. Yao, V. Modugno, A. M. Delfaki, Y. Liu, D. Stoyanov, and D. Kanoulas, "Local path planning among pushable objects based on reinforcement learning," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024, pp. 3062–3068.

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

[6] Z. Ren, B. Suvonov, G. Chen, B. He, Y. Liao, C. Fermuller, and J. Zhang, "Search-based path planning in interactive environments among movable obstacles," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2025, pp. 533–539.

[7] R. Ni and A. H. Qureshi, "Progressive learning for physics-informed neural motion planning," *arXiv preprint arXiv:2306.00616*, 2023.

[8] Y. Liu, R. Ni, and A. H. Qureshi, "Physics-informed neural mapping and motion planning in unknown environments," *IEEE Transactions on Robotics*, 2025.

[9] R. Ni and A. H. Qureshi, "Physics-informed neural motion planning on constraint manifolds," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 12 179–12 185.

[10] Y. Feng, C. Hong, Y. Niu, S. Liu, Y. Yang, and D. Zhao, "Learning multi-agent loco-manipulation for long-horizon quadrupedal pushing," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2025, pp. 14 441–14 448.

[11] Y.-C. Lin, B. Ponton, L. Righetti, and D. Berenson, "Efficient humanoid contact planning using learned centroidal dynamics prediction," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 5280–5286.

[12] Q. Rouxel, S. Ivaldi, and J.-B. Mouret, "Multi-contact whole-body force control for position-controlled robots," *IEEE Robotics and Automation Letters*, vol. 9, no. 6, pp. 5639–5646, 2024.

[13] Y. Wang and C. W. De Silva, "Multi-robot box-pushing: Single-agent q-learning vs. team q-learning," in *2006 IEEE/RSJ international conference on intelligent robots and systems*, 2006, pp. 3694–3699.

[14] S. Goyal, A. Ruina, and J. Papadopoulos, "Limit surface and moment function descriptions of planar sliding," in *IEEE International Conference on Robotics and Automation (ICRA)*, 1989, pp. 794–795.

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

[16] Z. Tang, J. Chen, and M. Guo, "Combinatorial-hybrid optimization for multi-agent systems under collaborative tasks," in *IEEE Conference on Decision and Control (CDC)*, 2023.

[17] K. N. McGuire, G. C. H. E. de Croon, and K. Tuyls, "A comparative study of bug algorithms for robot navigation," *Robotics and Autonomous Systems*, vol. 121, p. 103261, 2019.

[18] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," http://pybullet.org, 2016–2019.