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 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].

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 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 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. Model of Workspace and Robots

We consider a 2D workspace $\mathcal{W} \subset \mathbb{R}^2$ cluttered with immovable obstacles $\mathcal{O}^{\mathtt{fix}}$ and a set of M movable rigid polygons $\mathbf{\Omega} \triangleq \{\Omega_1, \ldots, \Omega_M\} \subset \mathcal{W}$. Each Ω_m has mass M_m, inertia I_m, frictional parameters (identified or estimated), state $\mathbf{s}_m(t) \triangleq (\mathbf{x}_m(t), \psi_m(t))$, and occupied region $\Omega_m(t)$. A small, fixed team of robots $\mathcal{R}_{\mathrm{grp}} \subseteq \mathcal{R}$ (with $|\mathcal{R}_{\mathrm{grp}}| \in \{2, 3\}$ in our experiments) operates as a single cooperative unit; each robot R has state $\mathbf{s}_R(t) \triangleq (\mathbf{x}_R(t), \psi_R(t))$ and footprint $R(t)$. The instantaneous free space is

$$\widehat{\mathcal{W}}(t) \triangleq \mathcal{W} \backslash \Big(\mathcal{O}^{\mathtt{fix}} \cup \{\Omega_m(t)\}_{m=1}^{M} \cup \{R(t)\}_{R \in \mathcal{R}_{\mathrm{grp}}} \Big).$$

The "large payload" (or agent to be routed) is modeled as a disc of radius $W/2$; a curve is W–feasible if its clearance is at least W everywhere.

B. Collaborative Pushing Modes

Robots interact with a movable obstacle Ω_m through *pushing modes*. Since $\mathcal{R}_{\mathrm{grp}}$ acts as a single unit, a mode for Ω_m is specified as $\boldsymbol{\xi}_m \triangleq (\mathcal{C}_m, \mathbf{u}_m)$, where $\mathcal{C}_m \subset \partial\Omega_m$ are the contact locations realized by the group and $\mathbf{u}_m$ encodes the nominal push action (e.g., body–frame velocity or an equivalent wrench profile). Let Ξ_m denote the admissible mode set determined by contact geometry and frictional limits. Different modes induce different motions of Ω_m through the physics engine.

C. Problem Statement

Given start $\mathbf{s}^{\mathrm{S}}$ and goal $\mathbf{s}^{\mathrm{G}}$, the objective is to compute a *hybrid schedule* $\pi = \{(m_k, \boldsymbol{\xi}_k, \Delta t_k)\}_{k=1}^{K}$ that reconfigures $\{\Omega_m\}$ by sequential pushes of the robot team so that a W–feasible path exists from $\mathbf{s}^{\mathrm{S}}$ to $\mathbf{s}^{\mathrm{G}}$. Let $T \triangleq \sum_{k=1}^{K} \Delta t_k$ be the task duration, and $J(m_k, \boldsymbol{\xi}_k; \mathbf{S}(\tau_k))$ the physics-based effort/feasibility cost of executing mode $\boldsymbol{\xi}_k \in \Xi_{m_k}$ starting at system state $\mathbf{S}(\tau_k)$. We seek a compact single-column form:

$$\begin{aligned} \min_{\{(m_k, \boldsymbol{\xi}_k, \Delta t_k)\}_{k=1}^{K}} \quad & T \;+\; \alpha \sum_{k=1}^{K} J(m_k, \boldsymbol{\xi}_k; \mathbf{S}(\tau_k)) \\ \text{s.t.} \quad & \boldsymbol{\xi}_k \in \Xi_{m_k},\ \Delta t_k > 0,\ \tau_{k+1} = \tau_k + \Delta t_k, \\ & \mathbf{S}(t^+) = \Phi\big(\mathbf{S}(t),\, m_k,\, \boldsymbol{\xi}_k\big),\ t \in [\tau_k, \tau_{k+1}), \\ & R(t) \cap R'(t) = \varnothing,\ \Omega_i(t) \cap \Omega_j(t) = \varnothing,\ \forall t, \\ & R(t) \cap \Omega_m(t) = \varnothing,\ R(t), \Omega_m(t) \subset \widehat{\mathcal{W}}(t),\ \forall t, \\ & \exists\, \mathcal{P}_W \subset \widehat{\mathcal{W}}(T) :\ S \rightsquigarrow G,\ \mathrm{clr}(\mathcal{P}_W) \geqslant W. \end{aligned} \tag{1}$$

Here, Φ is the (simulator-consistent) state transition under the selected pushing mode; $\alpha > 0$ balances duration and effort. The schedule π implicitly encodes *which obstacle is pushed, how, and for how long*; no per-robot assignment variables are introduced, consistent with our use of a fixed small team executing one push at a time.

III. Proposed Solution

This section presents a unified framework for collaborative multi-robot pushing. The approach consists of three main steps. First, a W-Clearance Connectivity Graph (Sec. III-A) is constructed to test the existence of a path of width W and identify frontier gaps when blocked. Second, a gap-ranking strategy (Sec. III-B) estimates the cost of clearing candidate gaps and guides robot coordination. Third, a simulation-in-the-loop search (Sec. III-C) expands a configuration-space tree while validating pushing actions through parallel physical simulation. The overall execution flow and generalizations are summarized in Sec. III-D.

A. W–Clearance Connectivity Graph (WCCG)

To reason about the existence of a traversable corridor of width W, we construct a *W–Clearance Connectivity Graph* (WCCG) that captures obstacle–to–obstacle adjacency under the threshold W and supports fast connectivity queries between start $\mathbf{s}^{\mathrm{S}}$ and goal $\mathbf{s}^{\mathrm{G}}$.

1) Construction (high level).: Polygonal obstacles are convex–decomposed. For each convex part we add a *centroid* node; for each visible closest pair between two parts whose distance is $< W$, we add *bridge* nodes at the closest points and a bridge–bridge edge annotated by the gap width; centroid–bridge edges connect bridges to their parts. Red dashed edges (Fig. 2) mark *narrow gaps*.

2) Connectivity test.: We employ a ray–shoot and loop–follow routine ("BugPlanner") on the WCCG frontiers: from a current point we shoot toward the goal, walk the encountered frontier loop by angular order, and either reach $\mathbf{s}^{\mathrm{G}}$ or certify a blocking loop. If connected, the routine returns a *skeleton* (an ordered center–bridge–center sequence) witnessing that $\mathbf{s}^{\mathrm{S}}$ and $\mathbf{s}^{\mathrm{G}}$ lie in the same face of the induced decomposition.

Theorem 1 (Complete criterion for a W–clear path)**.** *Let $\mathcal{F}_W := \mathbb{R}^2 \backslash (\mathcal{O} \oplus \mathbb{B}_{W/2})$ be the W–clear free space. A collision–free path of clearance at least W exists between $\mathbf{s}^S$ and $\mathbf{s}^G$ iff (i) they lie in the same face of the WCCG induced by $\mathcal{F}_W$, and (ii) both endpoint disks $\mathbb{B}_{W/2}(\mathbf{s}^S)$ and $\mathbb{B}_{W/2}(\mathbf{s}^G)$ are obstacle–free.*

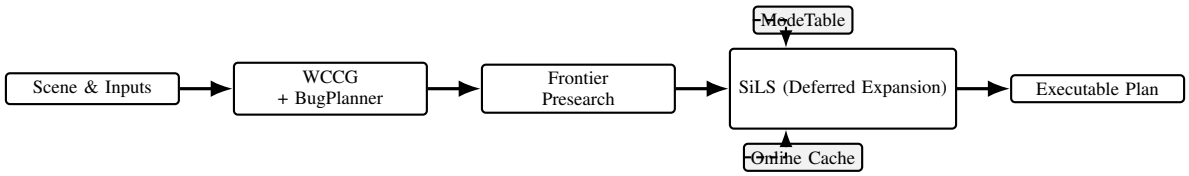

Fig. 1. TODO: Overall framework.

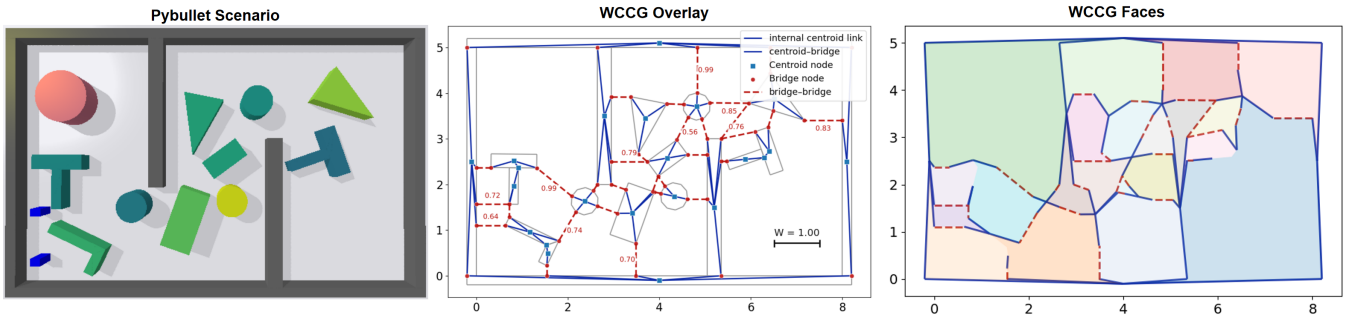

Fig. 2. (Left) top–down scene; (Middle) W–CCG with centroid (blue) and bridge (red) nodes, red dashed edges encode gaps $< W$; (Right) faces induced by W–CCG.

Algorithm 1: BugPlanner for W-width connectivity (skeleton witness)

In : $\mathbf{s}^{\mathrm{S}}$, $\mathbf{s}^{\mathrm{G}}$, WCCG
Out: if connected: skeleton Σ; else: frontier loop $\mathcal{L}$

$P \leftarrow \mathbf{s}^{\mathrm{S}}$, $\Sigma \leftarrow [\,]$
while *true* **do**
  if *segment* $P\mathbf{s}^{G}$ *hits no edge* **then**
    return $\Sigma \cup \{\mathtt{straight}(P, \mathbf{s}^{\mathrm{G}})\}$
  Build loop $\mathcal{L}$ by angle-follow from the hit edge; append traversed edges to Σ
  if $\mathrm{parity}(\mathcal{L}, \mathbf{s}^{S}\mathbf{s}^{G})$ *is odd* **then**
    return $(\varnothing, \mathcal{L})$
  Choose exit e on $\mathcal{L}$ (outward normal $\rightarrow \mathbf{s}^{\mathrm{G}}$); append arc to Σ; $P \leftarrow e$ (tiny bias)

Algorithm 2: Frontier Presearch for Gap Ranking (compact)

In : $\mathbf{s}^{\mathrm{S}}$, $\mathbf{s}^{\mathrm{G}}$, WCCG, λ_{trans}, λ_{push}
Out: First-hop gaps ranked by predicted cost

$(\mathcal{L}, \mathtt{conn}) \leftarrow \mathtt{BugFrontier}(\mathbf{s}^{\mathrm{S}}, \mathbf{s}^{\mathrm{G}})$;
if *conn* **then**
  return $\varnothing$
$\mathcal{C} \leftarrow$ bridge–bridge gaps on $\mathcal{L}$;
for $g \in \mathcal{C}$ **do**
  $J_1 \leftarrow \lambda_{\mathrm{trans}} \mathsf{C}_{\mathrm{trans}} + \lambda_{\mathrm{push}} \kappa(g)[\max(0, W - w(g)) + \delta]$;
  $(\mathtt{succ}, J_{\mathrm{tail}}) \leftarrow \mathtt{PresearchAfter}(g)$;
  $\widehat{\mathrm{Cost}}(g) \leftarrow J_1 + (\mathtt{succ}?J_{\mathrm{tail}} : \infty)$;
return $\mathrm{argsort}_{g \in \mathcal{C}} \widehat{\mathrm{Cost}}(g)$ (asc.)

Proof sketch. BugPlanner yields a frontier *skeleton* within one face. By convex decomposition and angular–order, each center–bridge–center segment can be slid onto the boundary of the $W/2$–inflated obstacles without touching unintended parts; concatenating these slides forms a boundary–following curve with cross–gap hops. A small inward offset lies entirely in $\mathcal{F}_W$; with cleared endpoint disks, short connectors attach the endpoints, giving a geometric W–clear path. The converse is immediate.

Remark 1 (Practical check)**.** In code we simply (a) run WCCG connectivity; (b) verify the two $W/2$ endpoint disks. If (a) passes but (b) fails, a few *away–from–endpoint* pushes may clear the disks; this is an engineering add–on, not core to the method.

Remark 2 (Skeleton $\rightarrow$ path)**.** BugPlanner certifies connectivity via a frontier skeleton. A geometric W–clear path is obtained by sliding Σ onto the $W/2$–inflated boundary, offsetting slightly inward into $\mathcal{F}_W$, and attaching the cleared endpoint disks. This mapping is continuous and preserves homotopy.

B. Gap Ranking Strategy

When no clearance-W path exists, we rank *blocking gaps* on the reachable frontier and expand the most promising one first. The score is obtained by a short A*-style *presearch* that predicts the end-to-end *cost-to-connect* if we cross a candidate gap now and then continue across subsequent frontiers.

1) Frontier and candidates.: Given $(\mathbf{s}^{\mathrm{S}}, \mathbf{s}^{\mathrm{G}})$, a bug-style query on the WCCG either certifies connectivity or returns a counter-clockwise *frontier loop* $\mathcal{L}$ that separates start and goal (Sec. III-A.2). The *first-hop* candidate set $\mathcal{G}(\mathcal{L}) = \{g_1, \ldots, g_K\}$ consists of visible bridge–bridge edges on $\mathcal{L}$.

2) Step cost.: For a gap $g \in \mathcal{G}(\mathcal{L})$, we define the one-hop cost

$$J(g \,|\, \mathcal{L}, \mathbf{s}) \;=\; \lambda_{\mathrm{trans}}\, \mathsf{C}_{\mathrm{trans}}(\mathbf{s} \rightarrow g) \;+\; \lambda_{\mathrm{push}}\, \mathsf{C}_{\mathrm{push}}(g), \quad (2)$$

where $\mathsf{C}_{\mathrm{trans}}$ is the straight-line approach from the current robot center $\mathbf{s}$ to the outside insertion point of g on $\mathcal{L}$, and

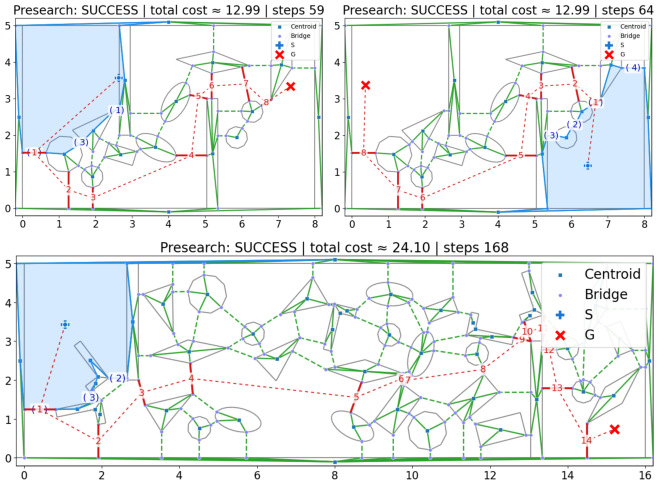

Fig. 3. **Frontier presearch and gap prioritization.** Each panel overlays the WCCG together with the currently selected face (light blue). Numbers in **red** mark the *gap-crossing order* returned by the presearch, and numbers in **blue** indicate the *local ranking* of first-hop gaps on the current 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.

the widening term

$$\mathsf{C}_{\text{push}}(g) \;=\; \kappa(g)\,\Big(\max\{0,\, W - w(g)\} + \delta\Big) \qquad (3)$$

scales the geometric shortfall $W - w(g)$ by a mass factor $\kappa(g) = \phi(\min\{M_u, M_v\}) \geqslant 1$ of the lighter adjacent obstacles (u, v); a simple choice is $\phi(m) = 1 + \alpha/(m + \varepsilon)$ (heavier sliders $\Rightarrow$ larger cost). We add a small margin $\delta > 0$ for robustness. A consistent one-step heuristic is

$$h(g) \;=\; \eta\,\big\|\mathbf{o}(g) - \mathbf{s}^{\mathsf{G}}\big\|_2, \qquad (4)$$

with $\mathbf{o}(g)$ the outside point of g and $\eta > 0$ a scale; this is an admissible proxy for the remaining crossings and empirically stabilizes the presearch.

3) Presearch ranking.: For each first-hop g, we run a short beam/A* that: (i) *virtually* crosses g, (ii) recomputes the next frontier $\mathcal{L}'$, and (iii) enqueues the top-B gaps on $\mathcal{L}'$ by priority $f = g\text{-cost} + J(\cdot) + h(\cdot)$. The process stops when the start and goal become connected (success) or a small depth/queue budget is reached (cut). The predicted end-to-end score of g is

$$\widehat{\mathsf{Cost}}(g) \;=\; J\big(g \mid \mathcal{L}, \mathbf{s}^{\mathsf{S}}\big) \;+\! \sum\nolimits_{g' \in \Pi^*(g)} J\big(g' \mid \cdot\big), \qquad (5)$$

where $\Pi^*(g)$ is the subsequent gap sequence discovered by the presearch after crossing g. We rank in ascending $\widehat{\mathsf{Cost}}(g)$.

4) Efficiency notes.: We cache frontier loops and memoize $(\mathcal{L}, g) \mapsto \mathcal{L}'$ transitions; nearest-edge hits are vectorized with AABB culling, giving near-linear time per ray-shoot in the number of visible edges. The presearch depth/beam B is small (e.g., 6–10), so ranking is typically subdominant to the physics loop.

C. Simulation-in-the-Loop Search

We embed a parallel physics predictor into search. Each expansion proposes ranked push tasks, filters them by a geometry *quick-pass* and a short-horizon simulation with *early-stop*, and refreshes the W-connectivity heuristic via frontier presearch—closing the gap between graph reasoning and executable actions. We refer to this two-stage filter as *the pipeline* below.

Algorithm 3: SiLS with Deferred Expansion (clean version)

In : $\mathbf{s}^{\mathsf{S}}$, $\mathbf{s}^{\mathsf{G}}$, W, initial snapshot, batch size B
Out: Executable push plan or `fail`

State of a node: $\nu = (\texttt{snap}, \texttt{plan}, g, \mathcal{C}, j)$, with candidates $\mathcal{C}$ (ranked) and cursor j.
Init: $\nu_0 \leftarrow (\texttt{snap}_0, \varnothing, 0, \varnothing, 0)$;
$\texttt{PQ} \leftarrow \{(\nu_0, f = \widehat{\mathsf{Cost}}_{\text{to-go}}(\texttt{snap}_0))\}$
while *PQ not empty* **do**
  Pop (ν, f) with minimal f;
  // $\nu = (\texttt{snap}, \texttt{plan}, g, \mathcal{C}, j)$
  if $\mathcal{C} = \varnothing$ **then**
    $(\texttt{connected}) \leftarrow \texttt{BuildWCCG}(\texttt{snap})$;
    if *connected* **then**
      return plan
    $\mathcal{L} \leftarrow \texttt{BugPlannerFrontier}(\mathbf{s}^{\mathsf{S}}, \mathbf{s}^{\mathsf{G}})$;
    $\mathcal{C} \leftarrow \texttt{PresearchRank}(\mathcal{L})$;
    $j \leftarrow 0$
  $\mathcal{B} \leftarrow \texttt{NextBatch}(\mathcal{C}, j, B)$;
  $j \leftarrow j + |\mathcal{B}|$
  foreach $\tau \in \mathcal{B}$ ***in parallel*** **do**
    $(\texttt{ok}, \texttt{snap}', t_{\text{push}}) \leftarrow \texttt{SimPredict}(\tau)$
    if *ok* **then**
      $g' \leftarrow g + \mathsf{C}_{\text{trans}} + t_{\text{push}}$;
      $h' \leftarrow \widehat{\mathsf{Cost}}_{\text{to-go}}(\texttt{snap}')$;
      Push $((\texttt{snap}', \texttt{plan} \cup \{\tau\}, g', \varnothing, 0), f' = g' + h')$ into PQ
  if $j < |\mathcal{C}|$ **then**
    reinsert current ν into PQ with the same f
return `fail`

1) Nodes & Priority (Deferred Expansion): A node $\nu = (\texttt{snapshot}, \texttt{plan}, g)$ keeps a lazily built, ranked candidate list $\mathcal{C}(\nu)$ and a cursor $j(\nu)$ to the next untried batch. We use a best-first score

$$f(\nu) = g(\nu) + \widehat{\mathsf{Cost}}_{\text{to-go}}(\texttt{snapshot}),$$

where g accumulates robot transit plus predicted push time, and $\widehat{\mathsf{Cost}}_{\text{to-go}}$ comes from the frontier presearch (Sec. III-B).

2) Progressive Expansion: When a node ν is popped, we expand it *progressively*.

Initialize on demand. If $\mathcal{C}(\nu)$ is empty, we (a) build the WCCG and extract one frontier loop via BugPlanner, (b) rank its bridge–bridge gaps by the frontier presearch, and (c) instantiate a few push tasks per top gap (ModeTable prior; fallback *away + jitter*). Set $j(\nu) = 0$.

Evaluate a small slice. Take the next B tasks $\mathcal{B} = \mathcal{C}(\nu)[\,j(\nu) : j(\nu)+B\,]$, advance $j(\nu) \leftarrow j(\nu)+|\mathcal{B}|$, and evaluate $\mathcal{B}$ in parallel through *the pipeline*. Each successful task τ yields a child ν' with updated snapshot, cost g', and priority $f' = g' + \widehat{\mathsf{Cost}}_{\text{to-go}}$.

Reinsert if unfinished. If $j(\nu) < |\mathcal{C}(\nu)|$, reinsert ν into the PQ with the *same* priority $f(\nu)$ to preserve best-

first ordering across partially expanded nodes and avoid PQ starvation.

3) Collaborative Pushing in Parallel Prediction: Each candidate task τ specifies a target gap and a short-horizon body twist $\mathbf{v} = (v_x, v_y, \omega)$. We synthesize per-robot contact modes $\boldsymbol{\xi}$ aligned with $\mathbf{v}$, validate $(\boldsymbol{\xi}, \mathbf{v})$ via *the pipeline*, and reuse verified modes within the subtree.

Reachable contact region from WCCG/BugPlanner. We reuse the same machinery with W set to the robot's circumscribed diameter, yielding the *reachable band* on the target boundary (convex pieces) for non–point robots. We uniformly sample contact points on this band and attach local frames $(\mathbf{n}, \tau)$ (see Fig. **??**).

Greedy mode search on sampled contacts. Given the desired body-frame velocity (arc tracker), we greedily assemble a contact assignment across robots using the objective of [5] (balanced normal support, torque leverage, slip margin), restricted to the sampled candidates. This returns a small set of locally optimal modes $\{\boldsymbol{\xi}\}$ for the requested velocity bin.

Online reuse. Verified $(\boldsymbol{\xi}, \mathbf{v})$ pairs and summary stats (tracking error, success rate, push time) are cached for the current expansion and descendants; a lightweight ModeTable acts as a persistent cache across scenes, exploiting the near-invariance of feasible modes for arc-like motions on a given shape. Only a few hundred entries per shape typically cover most velocity directions.

4) Efficiency Notes: Frontier loops are cached (loop signatures; $(\mathcal{L}, g) \mapsto \mathcal{L}'$ memoization), ray casts are vectorized with AABB culling, predictor workers are pre-forked and reused, and deferred expansion keeps the PQ nonempty until all candidates are tried.

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 3 (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.

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

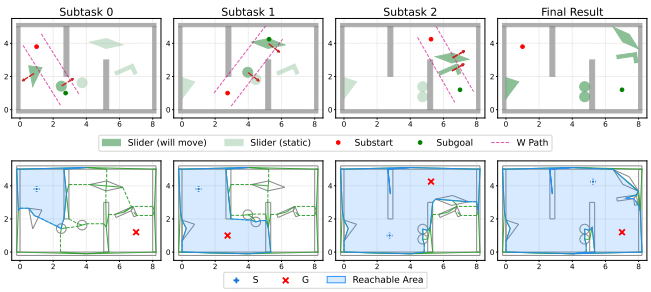

Fig. 4. SL-Push. Compute(offline) or simulate(sim-in-the-loop) the pushing directions for movable obstacles along the W-width straight path from sub-start to sub-goal. Top: Pushing steps. Bottom: Reachable region changes during pushing

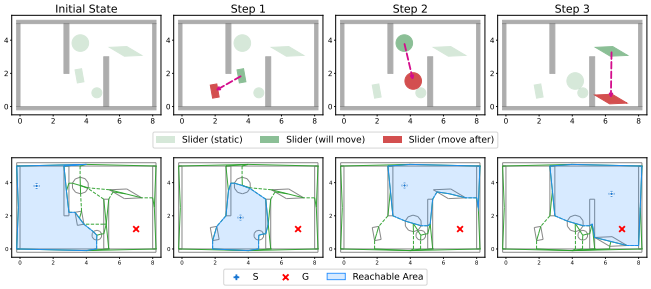

Fig. 5. Rec-NAMO. Rec-NAMO builds path segments sequentially but fails to construct the full W-widthpath from start to goal. Top: Pushing steps. Bottom: Reachable region changes during pushing

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` [**?**] 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 $\Delta t = 1/120\,\mathrm{s}$ and the control period is $1/40\,\mathrm{s}$. Robots are disk/box pushers with risk radii consistent with the W-clearance definition. Movable obstacles have masses uniformly sampled in $[5, 15]\,\mathrm{kg}$; 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 $8\times5\,\mathrm{m}$ 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.

TABLE I

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

Method	Succ. (%)	#PT (s)	#ET (s)	#Sims	#Pushes
DFS-WCCG	25	41.77	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.

2) Algorithm Configuration: Unless stated otherwise: per-task simulation horizon $= 80$ steps, up to 64 candidate push tasks per expansion, and priority $f = g + \widehat{\mathrm{Cost}}_{\text{to-go}}$ 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) Main Results: Table **??** summarizes performance on the nominal scenario. SiLS attains higher success with fewer simulations and lower time due to: (i) frontier-based gap ranking, (ii) ModeTable-guided push directions, and (iii) quick-pass/early-stop. Deferred reinsertion prevents priority-queue starvation by revisiting previously generated high-value nodes when a batch of actions fails.

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.

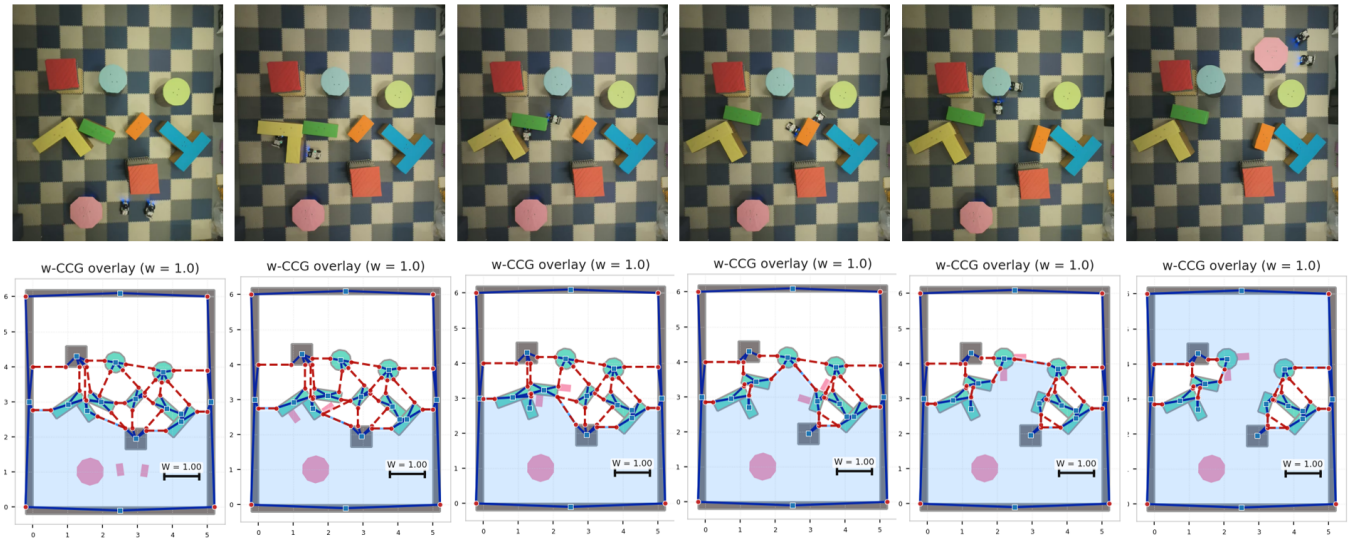

Fig. 6. SL-Push. Compute(offline) or simulate(sim-in-the-loop) the pushing directions for movable obstacles along the W-width straight path from sub-start to sub-goal. Top: Pushing steps. Bottom: Reachable region changes during pushing

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	–	–	–	–

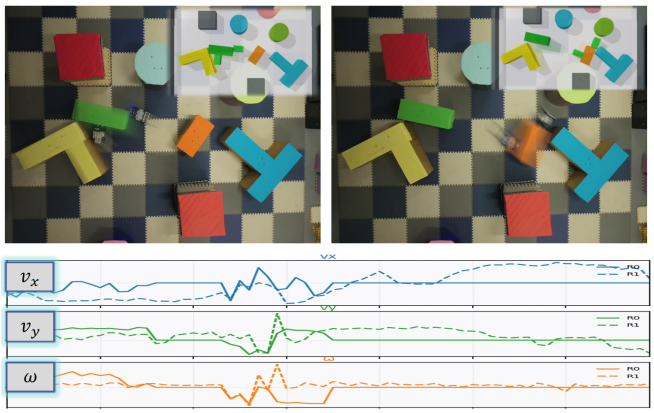

Fig. 7. SL-Push. Compute(offline) or simulate(sim-in-the-loop) the pushing directions for movable obstacles along the W-width straight path from sub-start to sub-goal. Top: Pushing steps. Bottom: Reachable region changes during pushing

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] G. L. O. Solver, `https://developers.google.com/optimization/lp`.