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 path of minimum width W, we construct a *w–Clearance Connectivity Graph* (WCCG). The graph encodes obstacle–to–obstacle adjacency under a width threshold: edges appear where the clearance between obstacles is below W, and nodes summarize convex parts of obstacles and the bridge points that lie on their boundaries. This discrete representation allows efficient queries about whether a start $\mathbf{s}^{\mathrm{S}}$ and a goal $\mathbf{s}^{\mathrm{G}}$ can be connected without crossing any gap narrower than W.

1) Free-Space View and Clearance Requirement: Let $\mathbf{\Omega}(t) = \{\Omega_1(t), \ldots, \Omega_M(t)\}$ denote movable obstacles at time t, $\mathcal{O}^{\mathtt{fix}}$ the static ones, and $\mathcal{W}$ the workspace. The instantaneous free space is

$$\widehat{\mathcal{W}}(t) \triangleq \mathcal{W} \backslash \big(\mathcal{O}^{\mathtt{fix}} \cup \{\Omega_m(t)\}_{m \in \mathcal{M}} \big). \tag{2}$$

A path is W–feasible iff it maintains clearance at least $W/2$ from the boundary of $\widehat{\mathcal{W}}(t)$. Conceptually, this equals navigating in the eroded set

$$\widehat{\mathcal{W}}^{W}(t) \triangleq \{p \in \widehat{\mathcal{W}}(t) \mid \mathrm{dist}(p, \partial\widehat{\mathcal{W}}(t)) \geqslant W/2\}. \tag{3}$$

In practice, we *do not* compute (3) explicitly; instead, we build a discrete graph that is equivalent for connectivity queries.

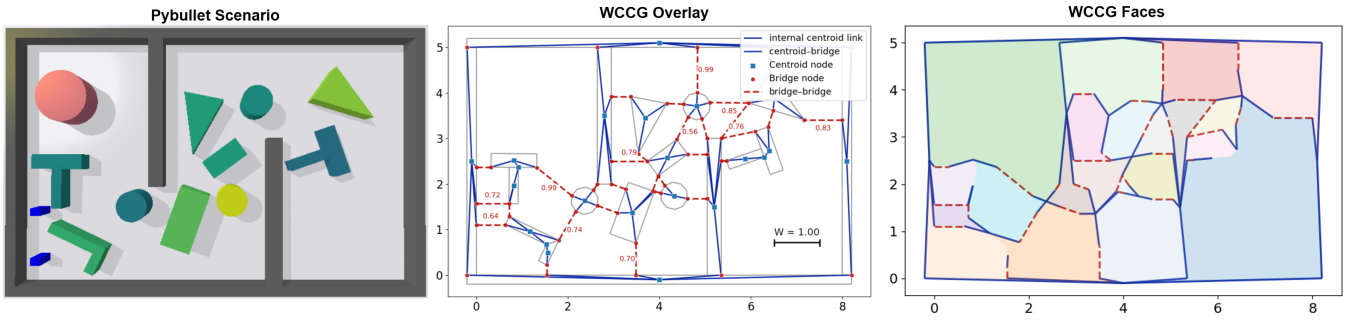

Fig. 1. (a) Top–down PyBullet scene (walls/obstacles in gray). (b) w–Clearance Connectivity Graph (w–CCG): blue solid edges are centroid links; red dashed edges are bridge–bridge links annotated with gap widths; blue squares and red circles denote centroid and bridge nodes, respectively; a scale bar shows W (here $W{=}1.00$). (c) Faces extracted from the w–CCG, displayed as semi-transparent regions.

2) Graph Construction from Polygonal Obstacles: Given polygonal obstacles decomposed into convex parts, the WCCG $\mathcal{G}^W = (\mathcal{V}^W, \mathcal{E}^W)$ is built as follows (see Alg. imp. in `WClearanceGraph`):

(i) Centroid nodes. For each convex subpolygon we add a centroid node; for composite obstacles we also connect adjacent subpolygons by a centroid–midpoint–centroid chain.

(ii) Bridge nodes. For each pair of obstacles we compute all closest-point pairs (p_i, p_j) between their convex parts whose Euclidean distance $d(p_i, p_j)$ is strictly less than W and whose segment is visible (no intersection with other obstacles). Each p_i (resp. p_j) becomes a bridge node attached to its subpolygon's centroid.

(iii) Bridge–bridge edges. Every visible pair (p_i, p_j) with $d(p_i, p_j) < W$ yields a bridge–bridge edge whose attribute stores the gap width. Centroid–bridge edges form the remainder of $\mathcal{E}^W$.

Intuitively, red dashed edges in Fig. 1(b) mark *narrow gaps* ($< W$), while blue solid edges organize the interior connectivity of obstacles.

3) Connectivity Test via BugPlanner: Given start $\mathbf{s}^{\mathrm{S}}$ and goal $\mathbf{s}^{\mathrm{G}}$, we test W-width connectivity on the WCCG using a ray-shoot–and–loop-follow procedure (Alg. 1). From a current point, the planner casts a ray toward the goal, takes the nearest intersected edge, and then walks the *frontier loop* by following the angular order at each vertex. If the loop has odd intersection parity with segment $\mathbf{s}^{\mathrm{S}}\mathbf{s}^{\mathrm{G}}$, it is a blocking frontier; otherwise an *exit* on the loop (chosen by outward normal toward the goal) is taken and the process *re-shoots* from that exit with a tiny bias to avoid degeneracy. The algorithm halts either with a straight, collision-free shot to $\mathbf{s}^{\mathrm{G}}$ or with a certified blocking loop.

Outputs (topological witness). If connectivity holds, the planner returns a frontier–guided *skeleton* Σ (an ordered sequence of WCCG edges: center–bridge–center–$\cdots$) that lives on obstacle/frontier adjacency, not necessarily inside the geometric W–clear tube. Otherwise, it returns (i) a closed frontier ring and (ii) the set of narrow gap edges on that ring (the *critical gaps*), i.e., edges with width $< W$ that lie on every route between the endpoints and are therefore the minimal widening targets.

From skeleton to a W–clear path. Under convex decomposition and the angular-order rule, each center–bridge–center segment of Σ can be slid continuously onto the boundary of the $W/2$–inflated obstacles, forming a boundary-following curve with cross-gap hops. A small inward normal offset of this curve lies entirely in the W–clear free space $\mathcal{F}_W$. With cleared endpoint disks $\mathbb{B}_{W/2}(\mathbf{s}^{\mathrm{S}})$ and $\mathbb{B}_{W/2}(\mathbf{s}^{\mathrm{G}})$, short connectors attach the endpoints, yielding a geometric path of clearance at least W.

Key implementation details. (i) Segment–segment intersections are computed in a vectorized fashion over all candidate edges (AABB culling + linear solve) to obtain the closest hit in one pass. (ii) Loop following uses the precomputed angular ordering of neighbors at each vertex; the side sign is tracked to stay on the same frontier. (iii) The re-shoot step adds a tiny directional bias to bypass $s \approx 0$ degeneracy. (iv) Visualization is incremental: rays, frontier segments, loop edges, and entry/exit points are upda

Theorem 1 (Complete criterion for a W-clear path)**.** *Let $\mathcal{O} \oplus \mathbb{B}_{W/2}$ be the Minkowski sum of all obstacles with a radius-$W/2$ disk, and let $\mathcal{F}_W := \mathbb{R}^2 \backslash (\mathcal{O} \oplus \mathbb{B}_{W/2})$ be the W-clear free space. There exists a collision-free path of clearance at least W from $\mathbf{s}^S$ to $\mathbf{s}^G$* iff

(i) *$\mathbf{s}^S$ and $\mathbf{s}^G$ lie in the same face (path-connected component) of the W-CCG induced by $\mathcal{F}_W$, and*

(ii) *the endpoint disks are clear: $\mathbb{B}_{W/2}(\mathbf{s}^S) \subset \mathcal{F}_W$ and $\mathbb{B}_{W/2}(\mathbf{s}^G) \subset \mathcal{F}_W$.*

Intuitive proof (sketch). Convex–decompose all obstacles and inflate each convex piece by a radius $W/2$; assume the endpoint disks $\mathbb{B}_{W/2}(S)$ and $\mathbb{B}_{W/2}(G)$ are free. **(i) Discrete skeleton from BugPlanner:** Running BugPlanner on the W–CCG produces a "center $\rightarrow$ bridge $\rightarrow$ center $\rightarrow \cdots$" sequence connecting S to G within one face of the graph. **(ii) Slide to the inflated boundary:** By convexity and the angular-order rule, each center–bridge–center segment can be continuously slid onto the boundary of the inflated obstacles, yielding a boundary-following curve with occasional cross-gap transitions between adjacent inflated pieces. **(iii) No premature boundary hits:** If this slide were to encounter a third inflated piece first, that piece would have offered a nearer bridge in the same angular sector; this contradicts BugPlanner's neighbor selection by angular order. Hence the slide stays on the intended boundary and designated bridges.

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* $P\mathbf{s}^{\mathrm{G}}$ *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 $\mathrm{parity}(\mathcal{L}, \mathbf{s}^{\mathrm{S}}\mathbf{s}^{\mathrm{G}})$ *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: Gap Ranking via Frontier Presearch

Input: $\mathbf{s}^{\mathrm{S}}$, $\mathbf{s}^{\mathrm{G}}$, WCCG, λ_{trans}, λ_{push}
Output: Frontier gaps ranked by predicted cost

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

(iv) Offset into free space: A small inward normal offset of the boundary-following curve lies entirely in the W–clear free space $\mathcal{F}_W = \mathbb{R}^2 \backslash (\mathcal{O} \oplus \mathbb{B}_{W/2})$. **(v) Attach endpoints:** Short connectors through the cleared endpoint disks link S and G to the offset curve, preserving the $W/2$ margin everywhere. **(vi) Conclusion:** The resulting route maintains distance at least $W/2$ to every obstacle, hence a W–clear path exists. □

Remark 1 (Practical check and engineering add-on)**.** In practice, the test reduces to two steps: (i) run the W-CCG connectivity between $\mathbf{s}^{\mathrm{S}}$ and $\mathbf{s}^{\mathrm{G}}$; (ii) confirm both endpoint disks $\mathbb{B}_{W/2}$ are obstacle-free. If (i) passes but (ii) fails, we optionally issue a few *away-from-endpoint* pushes to clear the disks; this is an implementation convenience rather than a core part of the method.

Remark 2 (Skeleton → 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* to decide where to clear first. Our ranking is *search- based*: it estimates the end-to-end *cost-to-connect* from the current state to the goal if we first cross a candidate gap and then continue crossing additional frontiers until a W-feasible path emerges.

1) Frontier and Candidates: Given start $\mathbf{s}^{\mathrm{S}}$ and goal $\mathbf{s}^{\mathrm{G}}$, we run a bug-style planner on the WCCG to either (i) certify direct connectivity or (ii) return a counter-clockwise frontier loop $\mathcal{L}$ separating $\mathbf{s}^{\mathrm{S}}$ and $\mathbf{s}^{\mathrm{G}}$ (cf. Sec. III-A.3). Let $\mathcal{G}(\mathcal{L}) = \{g_1, \ldots, g_K\}$ be the set of (visible) bridge–bridge gaps lying on $\mathcal{L}$. These K gaps are the first-hop candidates.

2) Step Cost Model: For a gap $g \in \mathcal{G}(\mathcal{L})$ we define a *step* cost that captures approach effort and the required widening (scaled by mass):

$$J(g \mid \mathcal{L}, \mathbf{s}) = \lambda_{\mathrm{trans}}\, \mathsf{C}_{\mathrm{trans}}(\mathbf{s}, \mathcal{L}, g) + \lambda_{\mathrm{push}}\, \mathsf{C}_{\mathrm{push}}(g). \quad (4)$$

Here (i) $\mathsf{C}_{\mathrm{trans}}$ is the straight-line distance from the current robot center $\mathbf{s}$ to an outside insertion point of g on $\mathcal{L}$; (ii) the widening cost is

$$\mathsf{C}_{\mathrm{push}}(g) = \kappa(g)\left[\max\{0,\, W - w(g)\} + \delta\right], \quad (5)$$

where $w(g)$ is the current gap width, $\delta > 0$ is a small safety margin, and $\kappa(g) = \phi\big(\min\{M_u, M_v\}\big) \geqslant 1$, scales the geometric requirement by a mass-dependent factor using the lighter adjacent obstacle masses M_u, M_v. The nonnegative weights λ_{trans} and λ_{push} trade off the two terms. To guide search we use a consistent one-step heuristic

$$h(g) \;=\; \eta \left\| \mathbf{o}(g) - \mathbf{s}^{\mathrm{G}} \right\|_2, \quad (6)$$

where $\mathbf{o}(g)$ is the outside point associated with g and $\eta > 0$ is a scaling factor.

3) Presearch and Ranking Score: Starting from the initial frontier $\mathcal{L}$, each first-hop candidate $g \in \mathcal{G}(\mathcal{L})$ is evaluated by a greedy beam/A*-like *presearch* that repeatedly: (i) crosses g, (ii) computes the next frontier $\mathcal{L}'$ with the bug planner, and (iii) enqueues the top-B gaps on $\mathcal{L}'$ by the priority $f = g\text{-cost} + J(\cdot) + h(\cdot)$. The presearch terminates either when the goal becomes directly visible (connected) or when the depth/queue budget is exceeded. For a first-hop g, the predicted end-to-end cost is

$$\widehat{\mathsf{Cost}}(g) = J\big(g \mid \mathcal{L}, \mathbf{s}^{\mathrm{S}}\big) + \sum\nolimits_{g' \in \Pi^*(g)} J\big(g' \mid \cdot\big). \quad (7)$$

where $\Pi^*(g)$ is the subsequent gap sequence chosen by the presearch from the new frontier after crossing g. We then rank candidates in ascending order of $\widehat{\mathsf{Cost}}(g)$. Intuitively, the top-ranked gap offers the best predicted connectivity improvement per unit effort, accounting for both the first crossing and the downstream crossings it enables.

4) Notes on Efficiency: We cache frontier loops by a loop signature and memoize transitions $(\mathrm{loop}, g) \mapsto \mathrm{loop}'$ to avoid recomputation. Nearest edge hits for ray shooting are computed in a vectorized manner with AABB culling, yielding near-linear time per shoot in the number of visible edges.

Algorithm 3: SiLS with Deferred Expansion (compact)

In : $\mathbf{s}^{\mathrm{S}}$, $\mathbf{s}^{\mathrm{G}}$, W, initial $\mathtt{Snap}$, batch size B
Out: Executable push plan or $\mathtt{fail}$

Initialize node $\nu_0 = (\mathtt{Snap}_0, \varnothing, g{=}0)$, set $\mathcal{C}(\nu_0) \leftarrow \varnothing$, $j(\nu_0) \leftarrow 0$;
$\mathtt{PQ} \leftarrow \{(\nu_0, f{=}\widehat{\mathsf{Cost}}_{\text{to-go}})\}$
while *PQ not empty* **do**
  Pop ν with minimal f;
  if $\mathcal{C}(\nu) = \varnothing$ **then**
    Build WCCG from $\mathtt{Snap}$; **if** *connected* **then**
      return $\mathtt{Plan}$
    $(\mathcal{L}, \mathtt{conn}) \leftarrow \mathtt{BugPlannerFrontier}(\mathbf{s}^{\mathrm{S}}, \mathbf{s}^{\mathrm{G}})$;
    Rank gaps on $\mathcal{L}$ by presearch; build $\mathcal{C}(\nu)$ as a ranked list of push tasks; set $j(\nu) \leftarrow 0$;
  Take next batch $\mathcal{B} \leftarrow \mathcal{C}(\nu)[\, j(\nu) : j(\nu){+}B \,]$; $j(\nu) \leftarrow j(\nu){+}|\mathcal{B}|$;
  foreach $\tau \in \mathcal{B}$ *in parallel* **do**
    $(\mathtt{ok}, \mathtt{Snap}', t_{\text{push}}) \leftarrow \mathtt{SimPredict}(\tau)$;
    if ok **then**
      $\Delta g \leftarrow \mathsf{C}_{\text{trans}} + t_{\text{push}}$; $g' \leftarrow g + \Delta g$; $h' \leftarrow \widehat{\mathsf{Cost}}_{\text{to-go}}(\mathtt{Snap}')$;
      Push child $\nu' = (\mathtt{Snap}', \mathtt{Plan} \cup \{\tau\}, g')$ with $f' = g' + h'$ into $\mathtt{PQ}$;
  if $j(\nu) < |\mathcal{C}(\nu)|$ **then**
    reinsert ν into $\mathtt{PQ}$ with priority f
return $\mathtt{fail}$
(all nodes exhausted)

C. Simulation-in-the-Loop Search

We integrate dynamics feasibility into planning by embedding a fast, parallel physics predictor inside the search loop. Each expansion proposes concrete push actions, validates them in simulation, and re-estimates clearance-W connectivity via the WCCG presearch. This closes the gap between graph reasoning and executable plans.

1) Node and Priority (with Deferred Expansion): A node stores the current world snapshot and partial plan, $\nu = (\mathtt{Snap}, \mathtt{Plan}, g)$, plus a lazily constructed, ranked candidate list for that snapshot:

$$\mathcal{C}(\nu) = \{\, \text{gap} \rightarrow \text{tasks} \,\}, \quad j(\nu) \in \{0, \ldots, |\mathcal{C}| - 1\}$$

where $j(\nu)$ is a cursor pointing to the next untried batch. We use

$$f(\nu) = g(\nu) + \widehat{\mathsf{Cost}}_{\text{to-go}}(\mathtt{Snap}), \tag{8}$$

with $\widehat{\mathsf{Cost}}_{\text{to-go}}$ from the WCCG frontier presearch (Sec. III-B). The incremental execution cost of one push is

$$\Delta g = \mathsf{C}_{\text{trans}}(\text{robots} \rightarrow \text{target}) + \mathtt{pushing\_time}.$$

2) Simulation-Guided Expansion (Progressive): When a node is popped, we *defer* full branching: (i) if $\mathcal{C}$ is uninitialized, build WCCG, extract the frontier loop, rank gaps by presearch, and generate a small set of push tasks per top gap (ModeTable prior → directions; fallback "away + jitter"); (ii) simulate only the next B untried tasks starting at cursor j (in parallel, with quick-pass and early-stop); (iii) push any successful children; (iv) if untried candidates remain, *reinsert the parent node* with $j \leftarrow j + B$. Thus the PQ never empties unless *all* nodes have exhausted their candidates.

a) Efficiency notes.: (1) **Frontier caching.** BugPlanner caches loop signatures and transitions $(\mathcal{L}, g) \mapsto \mathcal{L}'$, and uses vectorized ray–edge hits with AABB culling. (2) **Parallel pool.** Prediction environments are pre-forked (spawn) and reused; each round broadcasts only the snapshot and reachable contacts; we use $\mathtt{imap\_unordered}$. (3) **Quick-pass & early-stop.** A geometry-only screen can skip physics if the reference rollout clears the gap safely; otherwise the physics loop stops as soon as widening succeeds or progress stalls. (4) **Directional prior.** The ModeTable concentrates sampling on low-loss, well-aligned pushes while preserving exploration via a temperature-controlled softmax. (5) **Deferred expansion.** By batching only the next B tasks and reinserting partially expanded nodes, the PQ remains nonempty until every node's candidates are exhausted; this avoids premature termination when sampling is sparse.

D. Overall Analyses

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

a) 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.

b) 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.

c) 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 presearch, frontier extraction, and ModeTable prior) are integrated as described in Sec. **??**. 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 $\Delta t = 1/120\,\text{s}$ 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.

a) 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.

b) 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 $f = g + \widehat{\text{Cost}}_{\text{to-go}}$ 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:** Depth-first search over W-CCG adjacency with discrete axis-aligned pushes (object $\times$ $\{\leftarrow, \rightarrow, \uparrow, \downarrow\}$).
- **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.

TABLE I

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

Method	Succ. (%)	#PT (s)	#ET (s)	#Sims	#Pushes
DFS-WCCG	–	–	–	–	–
SL-Push (off-line)	–	–	–	–	–
SL-Push (sim)	–	–	–	–	–
Rec-NAMO	–	–	–	–	–
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	–	–	–	–

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`.