PuSHR: A Multirobot System for Nonprehensile Rearrangement

Sidharth Talia*, Arnav Thareja*, Christoforos Mavrogiannis, Matt Schmittle, Siddhartha S. Srinivasa

***Abstract*— We focus on the problem of rearranging a set of objects with a team of car-like robot pushers built using off-the-shelf components. Maintaining control of pushed objects while avoiding collisions in a tight space demands highly coordinated motion that is challenging to execute on constrained hardware. Centralized replanning approaches become intractable even for small-sized problems whereas decentralized approaches often get stuck in deadlocks. Our key insight is that by carefully assigning pushing tasks to robots, we could reduce the complexity of the rearrangement task, enabling robust performance via scalable decentralized control. Based on this insight, we built PuSHR, a system that optimally assigns pushing tasks and trajectories to robots offline, and performs trajectory tracking via decentralized control online. Through an ablation study in simulation, we demonstrate that PuSHR dominates baselines ranging from purely centralized to fully decentralized in terms of success rate and time efficiency across challenging tasks with up to 4 robots. Hardware experiments demonstrate the transfer of our system to the real world and highlight its robustness to model inaccuracies. Our code can be found at https://github.com/prl-mushr/pushr, and videos from our experiments at https://youtu.be/nyUn9mHoR8Y.**

I. Introduction

Multirobot systems have transformed sectors like fulfillment and warehouse automation. This was made possible in part thanks to efficient, scalable algorithms for multiagent pathfinding (MAPF) [3, 18, 23, 29] and task assignment (TA) [8, 11, 17, 21, 27]. Often, real-world robot deployment of such algorithms leverages extensive workspace and hardware engineering. Robots are typically holonomic, follow predefined paths (e.g., wire-guided), move in very constrained ways (e.g., rectilinearly like Amazon's Kiva robots [38]), and feature advanced gripping mechanisms.

In this paper, we are driven by the challenge of tackling real-world rearrangement tasks on constrained hardware with minimal workspace engineering. Using a team of car-like robots [30], we consider the task of rearranging a set of cubic blocks into a desired planar pattern via pushing actions realized via their front bumper (see Fig. 1). Leveraging pushing, mobile robots can be converted into effective mobile manipulators, capable of completing even complex rearrangement tasks without requiring customized grippers.

Simultaneously accounting for collision avoidance and push-stability in a tight space demands highly coordinated motion. Centralized MAPF algorithms can produce plans for complex rearrangement problems but accounting for model inaccuracies online requires frequent replanning that is impractical to execute on constrained hardware. On the other hand, decentralized controllers [5, 33] provide scalability but lack an understanding of the global task structure which often gets them stuck in deadlocks.

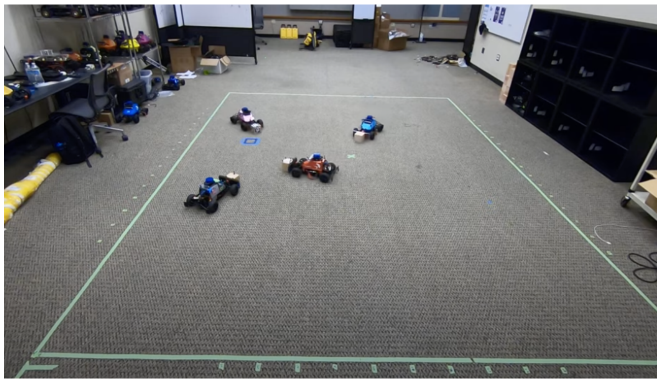


Fig. 1: The PuSHR system guides four MuSHR [30] robots to rearrange four blocks via pushing.

Our key insight is that optimally assigning pushing tasks to robots could simplify multi-robot coordination and enable robust performance via scalable decentralized control. Combined task and motion planning is intractable in our problem domain. Instead of simultaneously iterating over assignments and paths, we first plan an optimal task assignment considering a discretized workspace representation, and using this assignment, we plan collision-free trajectories for all robots in the continuous workspace. Each robot is then tracking its assigned trajectory via decentralized model predictive control accounting for collision avoidance and trajectory deviations.

Through a simulation-based ablation study, we demonstrate that our system (PuSHR) is capable of handling a variety of complex rearrangement scenarios involving up to 4 robots, including asymmetric scenarios with more or fewer robots than objects. In particular, PuSHR is the only system that successfully completes all scenarios, often with the top time efficiency among a series of baselines ranging from fully decentralized to fully centralized control. Through hardware experiments on a team of MuSHR robots [30], we present statistical insights about the ability of PuSHR to handle model inaccuracies and produce robust performance across challenging scenarios with up to 4 robots.

II. Related Work

Our system design brings together multirobot system design, and both nonprehensile and collaborative manipulation.

*Denotes equal contribution.

This work was (partially) funded by the Honda Research Institute USA, the National Science Foundation NRI (#2132848) and CHS (#2007011), DARPA RACER (#HR0011-21-C-0171), the Office of Naval Research (#N00014-17-1-2617-P00004 and #2022-016-01 UW), and Amazon.

Authors are with the Paul G. Allen School of Computer Science & Engineering, University of Washington, Seattle, WA, USA. Email: {sidtalia, athareja, cmavro, schmttle, siddh}@cs.washington.edu.

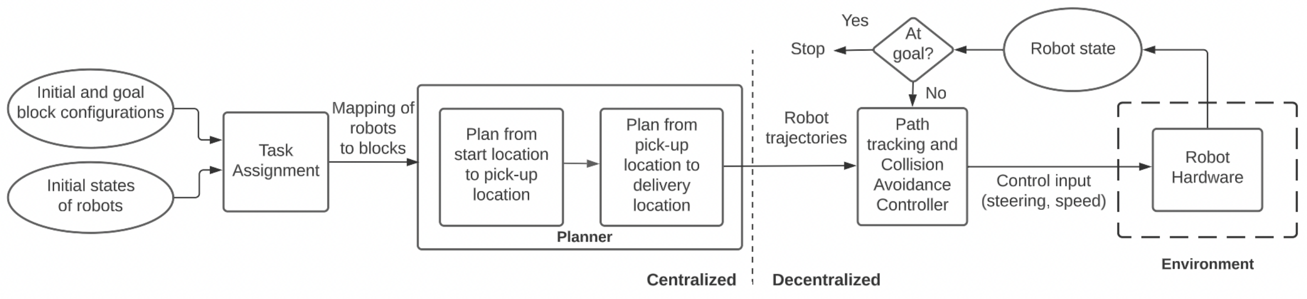

Initial and goal block configurations
Initial states of robots
Task Assignment
Mapping of robots to blocks
Plan from start location to pick-up location
Plan from pick-up location to delivery location
Planner
Centralized
Decentralized
Robot trajectories
Path tracking and Collision Avoidance Controller
At goal?
Yes
Stop
No
Robot state
Control input (steering, speed)
Robot Hardware
Environment

Fig. 2: Overview of the PuSHR system architecture. Our system comprises task assignment, grasp planning, and local collision avoidance control. PuSHR guides a set of robots to push a set of blocks toward a set of goal configurations.

A. Multirobot Systems

Many important advances in MAPF came in recent years, driven by the scalability of multi-robot systems. Algorithms like M^* [35] and Conflict Based Search (CBS) [29] and their variants are behind many successful real-world deployments of multi-robot systems, ranging from fulfillment to warehouse and manufacturing. Such deployments have encouraged further investigation of MAPF variants like multiagent pickup and delivery [3, 21, 23] and the incorporation of kinodynamic constraints [18, 37]. In parallel, TA has also found important applicability in physical domains involving teams of robots and/or humans [8, 17, 27]. In this paper, we integrate recent advances in kinodynamic MAPF [37] and TA [11] into a real-world system.

Besides multi-robot planning, there is recent activity in control for dynamic multiagent domains. Graphics applications have motivated crowd simulators [33], whereas robot deployment in crowded and driving domains has led to proactive controllers for collision avoidance [26, 28]. The insights of such algorithms are often transferred in multi-robot domains where teams of robots navigate in close proximity [15]. In this work, we leverage insights from multiagent collision avoidance [5] into the design of a decentralized local controller that robustly adjusts for deviations from the multiagent plan during deployment.

B. Nonprehensile Manipulation

Nonprehensile manipulation and especially push-based manipulation has been an active area of research [31]. Early work studied mechanics and control for the quasistatic settings [9, 12, 22, 24]. Later work focused on planning using nonprehensile manipulation primitives for complex, cluttered environments [6, 14]. In [10], they consider pushing multiple objects with a single pusher surface. Finally, some work has focused on handling parameter uncertainty by extracting data-driven force-motion models [2, 40], learning to adapt to different objects [19], or learning end-to-end pushing policies [20, 39]. In this paper, we leverage insights on quasistatic pushing [12, 14, 22] to inform the planning and low-level control components of our architecture.

C. Collaborative Manipulation

An active line of work explores strategies of distributing manipulation tasks across a team of robots via implicit signaling encoded in robots' behaviors [25, 32, 36]. Some works developed multi-robot systems capable of assembling furniture [16] or transporting deformable objects [1] via prehensile manipulation. The assumption of rigidity at the contacts [32] often removes the need for force measurements [34]. This can be restrictive as contact parameters are challenging to estimate. In some works robots act as sensors, collaboratively filtering important system parameters to guide the manipulation strategy [4]. Other works focus on alternative manipulation strategies such as multi-robot caging [7] and pushing [25].

Unlike much of the prior work in collaborative manipulation which focuses on the transportation of a single object with holonomic pushers [1, 4, 13, 36], in this paper we consider the rearrangement of *multiple* objects using nonholonomic pushers, with each object being controlled by only one pusher.

III. The Multiagent Nonprehensile Rearrangement Planning Problem

We consider a team of n mobile robots and a set of m rectangular blocks, embedded in a workspace $\mathcal{W} \subseteq \mathbb{R}^2$. We denote the states of the robots as $p^i \in SE(2)$, $i \in \mathcal{N} = \{1, \ldots, n\}$ and the states of the objects as $o^j \in \mathbb{R}^2$, $j \in \mathcal{M} = \{1, \ldots, m\}$ (we ignore objects' orientation). Each robot i follows rear-axle simple-car kinematics $p^i_{k+1} = f(p^i_k, u^i_k)$, where u^i_k represents a control action, drawn from a space of velocities and steering angles $\mathcal{U} = [-v_{max}, v_{max}] \times [-\phi_{max}, \phi_{max}]$ at timestep k. Robots may manipulate objects via pushing realized via flat-surface bumpers attached at their front (see Fig. 3). The goal of the robots is to rearrange the objects from an initial configuration $S = (s^1, \ldots, s^m)$ to a goal configuration $G = (g^1, \ldots, g^m)$ via sequences of push-based manipulation actions. Our goal is to design a system to enable the robots to successfully complete the rearrangement from S to G with an efficient makespan that scales robustly with the number of robots.

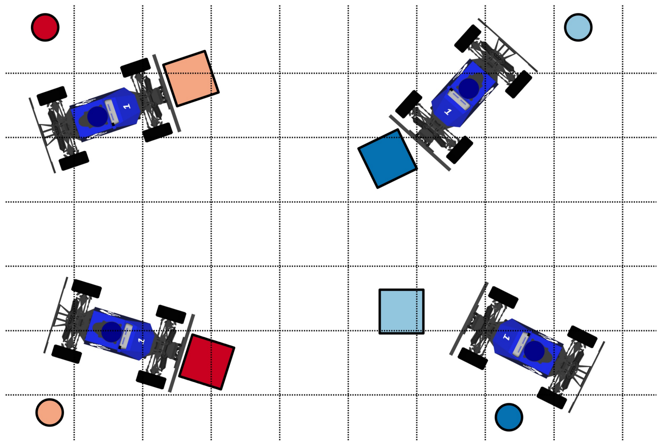


Fig. 3: Problem setup. A team of mobile robots is pushing a set of blocks towards a set of goal locations, indicated as circles of same color.

IV. PuSHR: A Multirobot System for Nonprehensile Rearrangement

We describe a planning architecture for multi-robot, multi-object nonprehensile rearrangement problems.

A. Architecture Overview

As shown in Fig. 2, our architecture consists of three main layers: a) a centralized task-assignment module that assigns objects to robots; b) a centralized global planning module that assigns paths to robots; c) a decentralized control module that enables robots to track their paths while maintaining contact with their assigned objects and avoiding collisions. The process begins with the task assignment system receiving the initial location of the robots and the pickup and delivery location of the blocks. The task assignment system then provides the global planner with the optimized robot-task pairs. The global planner produces a space-time trajectory for each robot. The onboard local planner then produces the controls to follow said trajectory.

B. Task Assignment

Given robots' initial states $P_0 = (p_0^1, \ldots, p_0^n)$ and blocks' initial and final configurations, S and G respectively, the objective of the task assignment module is to generate an efficient assignment of robots to blocks, i.e., $\mathcal{N} \rightarrow \mathcal{M}$. Efficiency refers to the total distance required for all robots to complete their rearrangement tasks. We cast this problem as an instance of multiagent pathfinding (MAPF) on a discrete graph and solve it using the Enhanced Conflict Based Search with Task Assignment (ECBS-TA) by Hönig et al. [11] which we adapt to our problem domain below.

Due to computation considerations, we use a discretized representation of the workspace for task assignment. We partition the workspace $\mathcal{W}$ (see Fig. 3) into a set of discrete regions whose connectivity we describe as a graph $\mathcal{G} = (V, E)$, where vertices $v \in V$ represent regions, and edges $e \in E$ are adjacency relationships between them. Based on this partition, we map the state of a robot/block to the workspace region that contains the majority of its volume. We assume that at each timestep, a robot can only move to an adjacent vertex or wait at its current vertex.

ECBS-TA searches for a collision-free path set $\mathcal{P} = (\mathcal{P}^1, \ldots, \mathcal{P}^n)$ such that path i connects the initial region of robot $i \in \mathcal{N}$ with the initial and final region of block $j \in \mathcal{M}$. The search sequentially refines path assignments by resolving emerging robot collisions, driven by a Euclidean-distance-based heuristic until a collision-free path set $\mathcal{P}$ is found. A byproduct of this process is the assignment of block $j \in \mathcal{M}$ to robot $i \in \mathcal{N}\ \forall i \in \mathcal{N}$. We modify ECBS-TA to handle problems with $n \neq m$ as follows: when $n > m$, the m robots with the lowest cost complete the rearrangement, and when $n < m$, the robots rearrange the n blocks with the lowest cost first, then task assignment is assigns the remaining $m - n$ blocks to robots until all blocks are assigned.

C. Centralized Global Planning

The paths produced by the task assignment module do not account for robot kinematics. Thus, we invoke a global planner to generate smooth, collision-free robot trajectories that respect the task assignment captured in $\mathcal{P}$. For this, we use CL-CBS [37], an algorithm that searches for a collision-free trajectory set $\mathcal{R} = (\mathcal{R}^1, \ldots, \mathcal{R}^n)$, where $\mathcal{R}^i = (r_0^i, \ldots, r_K^i)$ is the trajectory planned for robot $i \in \mathcal{N}$. The search process assembles individual trajectories from a set of kinematically feasible motion primitives and iterates until conflicts are resolved using a conflict-based search (CBS) methodology [29]. These primitives are: stopping in place; moving forward/backward by 1 unit (equal to the speed of the robot multiplied by the timestep); moving forward left/right on a unit-length circular arc; moving backward left/right on a unit-length circular arc.

We invoke CL-CBS twice: first to plan a set of paths that take robots from P_0 to S and wait until all robots reach S, and then to take robots from S to G. This process outputs a time-indexed trajectory set $\mathcal{R} = (\mathcal{R}^1, \ldots, \mathcal{R}^n)$ where $\mathcal{R}^i = (r_0^i, \ldots, r_K^i)$ is the trajectory planned for robot $i \in \mathcal{N}$.

D. MPC for Multiagent Collision Avoidance

Even small inconsistencies in following the timing of the planned trajectories may accumulate and result in collisions when directly following $\mathcal{R}$. For this reason, we deploy a decentralized model predictive controller (MPC) on each robot that besides path tracking accounts for multi-agent collision avoidance. We formulate this problem for robot i as a discrete optimization over a set of control trajectories $\boldsymbol{\mathcal{U}}$. For robot i, at every loop, the MPC outputs a trajectory $\boldsymbol{u}^{i*}$ determined as follows:

$$\begin{aligned} \boldsymbol{u}^{i*} = \arg\min_{\boldsymbol{u}^i \in \boldsymbol{\mathcal{U}}} \sum_{k=0}^{N-1} \Big(& a_{cte}\mathcal{J}_{cte}(p_k^i, r_k^i) + a_{col}\mathcal{J}_{col}(P_k) \\ & + a_{time}\mathcal{J}_{time}(p_k^i, r_k^i) \Big) \\ & s.t.\ p_{k+1}^i = f(p_k^i, u_k^i) \end{aligned} , \quad (1)$$

where $\mathcal{J}_{cte}$ is a cross-track error cost forcing the robot to stay close to its reference path, $\mathcal{J}_{col}$ is a collision avoidance

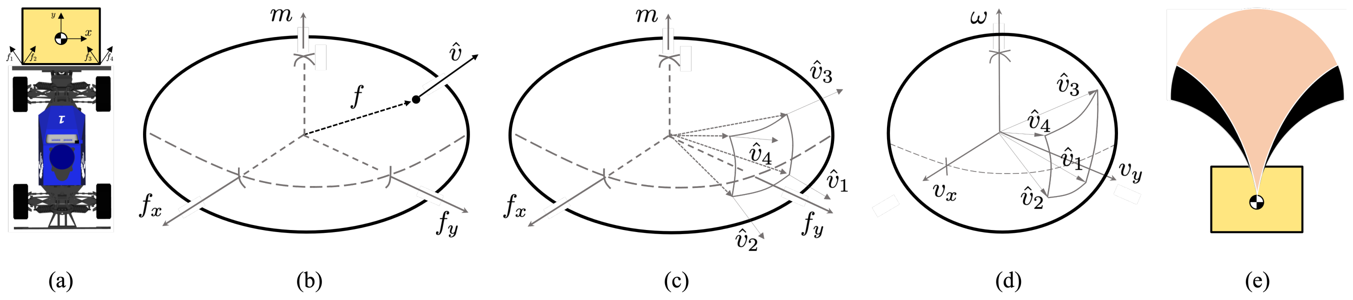


Fig. 4: Deriving the stable set of controls for quasistatic pushing following prior work of [9, 14, 22]. (a) Robot-block pushing system, and static friction boundary forces. (b) Limit surface, mapping force on the object to resulting unit velocity. (c) Boundary friction forces are mapped to boundary unit velocities through the limit surface. (d) The stable set of unit velocities represented on the unit control sphere. (e) Full (black color) and stable (beige) set of controls mapped to block velocities.

cost penalizing proximity between robots, $\mathcal{J}_{time}$ is a timing tracking cost, penalizing deviations from the timing defined by the global plan, and a_{cte}, a_{col},a_{time} are weights. The timing cost is defined as follows:

$$\mathcal{J}_{time}(p_k^i, r_k^i) = \widehat{ate}^i(t) - ate_k^i \qquad (2)$$

where $\widehat{ate}^i$ is the reference along-track error from a waypoint as a function of time resulting from the robot's planned path, and ate_k^i is the along-track error for a given time-step. The collision cost is defined as:

$$\mathcal{J}_{col}(p_k) = \sum_{j \in \mathcal{N} \setminus i} \max(d_{thr} - ||p_i^k - p_j^k||, 0) \qquad (3)$$

where d_{thr} is a threshold beyond which the cost is 0.

E. Stable Pushing

Following Lynch and Mason [22], assuming quasistatic pushing at the block/bumper contact, we derive the stable set of unit velocities $\hat{\mathcal{U}}_{stable}$ that guarantee that the pushed object remains fixed on the bumper. We model the contact between the robot's bumper and a block as a line defined by two-point contacts at the edges of the block (Fig. 4a). Given the block/bumper friction coefficient μ, we calculate the boundary forces f_1, f_2, f_3, f_4 and use them to construct a composite friction cone. Through the limit surface [9, 14] (see Fig. 4b), we map these forces to limit unit velocities $\hat{v}_1, \hat{v}_2, \hat{v}_3, \hat{v}_4$, representing the boundary of the stable set $\hat{\mathcal{U}}_{stable}$ (Fig. 4c), beyond which the block would start sliding. Finally, we map these velocities to limits of robot controls using the robot kinematics (Fig. 4d) and extract a limiting turning radius that the robots must respect to maintain control of the blocks while pushing (Fig. 4e). Assuming a coefficient of friction $\mu \approx 0.6$ (which we experimentally measured upon attaching sandpaper on the robots' bumpers), we found this radius to be $1.6m$, corresponding to a steering-angle limit of $\phi_{max} = 0.17rad$ for a MuSHR robot [30]. This value was also confirmed through experiments for speeds ranging from $0.3 - 0.8m/s$.

F. Implementation

In our ECBS-TA implementation, we set the suboptimality factor $w = 1.3$, which gave high-quality assignments within acceptable times. We modified CL-CBS [37] to account for pushing stability: when planning a pushing trajectory phase, we restrict the maximum steering angle to be less than ϕ_{max}. Our MPC used $d_{thr} = 0.6m$, $a_{cte} = 200$, $a_{time} = 20$, $a_{col} = 15$, which we obtained via parameter sweeps over success rate and minimum distance for scenarios similar to S3.a,b, S4.a,b (see Fig. 5). To remain within the quasistatic regime, we set $v_{max} = 0.4m/s$, $\phi_{max} = 0.314\ rad$ during non-pushing phases, and $\phi_{max} = 0.17\ rad$ during pushing phases. We assume that the agents have perfect knowledge of each other's states and block positions. In the lab, we achieve this using a motion-capture system.

V. Evaluation

We conduct simulated and lab experiments by deploying MuSHR [30] robots in a workspace of size $4 \times 6m^2$.

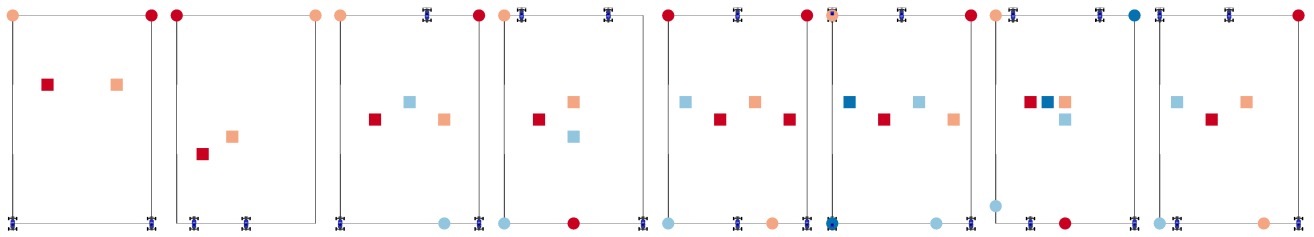
A. Experiment Design

Scenarios: We investigate the performance of PuSHR across 8 scenarios of varying difficulty involving two, three, and four robots (see Fig. 5). The scenarios involve navigating from a set of initial robot configurations to a set of initial block configurations, making contact with the blocks, and pushing them to the blocks' goal configurations. They were designed to give rise to challenging encounters among robots.

Algorithms: To evaluate PuSHR, we perform an ablation study, comparing its performance against a set of variants. Note that in the following descriptions, *manual TA* refers to TA that ranks around the median of all possible assignments ordered with respect to the time cost used by ECBS-TA [11]. See Fig. 7 for an example of how different assignments in scenario S4b rank and map to path planning Makespan.

LC. We implemented a Local Control (LC) baseline using the NonHolonomic Time To Collision (NHTTC) [5], an optimization-based decentralized controller that accounts for robot kinematics and multi-robot collision avoidance. NHTTC evaluates a set of constant-radius trajectories with respect to distance to goal and time to collision. During pushing, we constrain these trajectories to be in the stable set (see Fig. 4e).

TA-LC. This baseline is also LC, executed considering *optimal TA* provided by ECBS-TA [11].



(a) S2.a. (b) S2.b. (c) S3.a. (d) S3.b. (e) S3.c. (f) S4.a. (g) S4.b. (h) S4.c.

Fig. 5: Scenarios used in our evaluation. The cars are shown in their initial configurations. Initial block configurations are shown as squares whereas their goal positions are shown as circles (we ignore orientations).

TABLE I: Success rate over 100 simulated trials for each scenario and algorithm.

Configuration	2 robots - 2 objects		3 robots - 3 objects		3 robots - 4 objects	4 robots - 4 objects		4 robots - 3 objects
Scenario	S2.a	S2.b	S3.a	S3.b	S3.c	S4.a	S4.b	S4.c
LC	0.99	0.00	0.00	0.00	0.00	0.00	0.00	0.00
TA-LC	0.99	0.99	0.89	0.99	0.00	0.98	0.00	0.73
GP	0.99	0.98	0.88	1.00	0.00	0.99	0.00	0.14
GP-CA	1.00	0.98	0.82	0.99	0.00	0.96	0.20	0.92
PuSHR	**1.00**	**0.99**	**0.99**	**1.00**	**0.96**	**0.99**	**1.00**	**0.98**

GP. Using *manual* TA, the Global Planning (GP) baseline invokes CL-CBS [37] to plan trajectories for all robots, and a decentralized MPC to track the trajectories. This MPC is equivalent to the description of eq. (1) with $a_{col} = 0$.

GP-CA. This baseline is identical to GP but uses $a_{col} \neq 0$ in eq. (1) to account for collision avoidance.

PuSHR. PuSHR is identical to GP-CA, but leverages *optimal TA* through ECBS-TA [11] as described in Sec. IV.

Metrics: We evaluate performance with respect to:

Success rate. We consider a trial to be successful if all robots are able to move their assigned blocks within a threshold distance of $0.1m$ from their goals. Missing the block is also considered a failure even if the robots make it to their end location within the specified tolerance.

Makespan. The time taken by the last robot to reach the goal of its assigned block in a successful trial. The clock starts when the plans are published by the global planner.

Minimum distance. The minimum Euclidean distance between any two robots during a successful trial. A large minimum distance is advantageous as it provides an additional buffer to correct plan deviations and avoid collisions.

Hypotheses: We generally expect that the spatial structure representation introduced through TA and GP will improve the system's time efficiency. We also expect that the collision avoidance (CA) module will enable robots to keep a larger buffer between each other. We distill our expectations about system performance into the following hypotheses:

H1. GP achieves higher success rate than LC.

H2. GP-CA achieves higher minimum distance than GP.

H3. TA improves the success rate for LC and GP-CA.

H4. PuSHR is the most successful across all scenarios.

H5. PuSHR maintains block control in the real world.

Experimental Procedure. To extract statistical insights about the *planning* and *tracking* performance of PuSHR, we initially ran the scenarios of Fig. 5 in simulation. For each scenario, we generated 100 different trials, each resulting from a random spatial perturbation (radius $0.05m$) of all robots' initial configurations. These perturbations allow us to understand the robustness of PuSHR while ensuring the consistency of the scenarios. We executed all trials for each scenario with each of the algorithms.

To characterize the robustness of PuSHR to model inaccuracies, we executed some of the more challenging scenarios in an identical lab setup. As blocks, we used a set of wooden cubes with a side of $0.1m$. Our workspace was fully covered by a motion capture system of 12 overhead cameras, providing high-accuracy localization of robots ($\sim 1mm$). Note that we only recorded the initial block configurations, letting our system push the blocks to their goals by leveraging the stable pushing derivation, without closing the loop on block position.

B. Results

In Tables I, II, and III, we report respectively the Success rate, Makespan, and Minimum distance for all algorithms across all scenarios in simulation. In Fig. 8, we report the performance of PuSHR on three of the scenarios (S2.b, S3.a, and S4.a) executed in the lab. Footage from our lab experiments can be found at https://youtu.be/nyUn9mHoR8Y.

H1. GP completes 7/8 scenarios with high success (Table I) whereas LC only completes the easiest scenario (S2.a), thus H1 is confirmed. Lacking a detailed predictive model, LC cannot anticipate future conflicts in such a confined space. This observation is more pronounced for $n \geq 3$ with LC often violating the workspace or getting stuck.

H2. GP-CA keeps greater or equal clearances than GP for similar success (Table III), thus confirming H2. As n increases, planning with GP produces more complex maneuvers that need to be executed in a tight space. Thus, minor deviations

TABLE II: Makespan (s) over 100 simulated trials for each scenario and algorithm.

Configuration	2 robots - 2 objects		3 robots - 3 objects		3 robots - 4 objects	4 robots - 4 objects		4 robots - 3 objects
Scenario	S2.a	S2.b	S3.a	S3.b	S3.c	S4.a	S4.b	S4.c
LC	29.25 $\pm$ 2.83	N/A	N/A	N/A	N/A	N/A	N/A	N/A
TA-LC	30.03 $\pm$ 2.97	25.03 $\pm$ 0.00	**17.02** $\pm$ **0.00**	**17.91** $\pm$ **0.31**	N/A	**17.60** $\pm$ **0.49**	N/A	**14.17** $\pm$ **0.36**
GP	20.02 $\pm$ 0.10	32.75 $\pm$ 0.52	33.84 $\pm$ 0.44	29.06 $\pm$ 0.01	N/A	34.08 $\pm$ 0.02	N/A	32.49 $\pm$ 0.50
GP-CA	20.03 $\pm$ 0.00	32.72 $\pm$ 0.49	33.98 $\pm$ 0.50	30.12 $\pm$ 0.83	N/A	34.34 $\pm$ 0.44	31.20 $\pm$ 0.35	32.76 $\pm$ 0.46
PuSHR	**20.02** $\pm$ **0.00**	**13.02** $\pm$ **0.00**	22.01 $\pm$ 0.17	20.61 $\pm$ 0.50	**51.55** $\pm$ **0.85**	23.36 $\pm$ 0.47	**21.05** $\pm$ **0.01**	18.03 $\pm$ 0.01

TABLE III: Minimum distance (m) over 100 simulated trials for each scenario and algorithm.

Configuration	2 robots - 2 objects		3 robots - 3 objects		3 robots - 4 objects	4 robots - 4 objects		4 robots - 3 objects
Scenario	S2.a	S2.b	S3.a	S3.b	S3.c	S4.a	S4.b	S4.c
LC	0.50 $\pm$ 0.01	N/A	N/A	N/A	N/A	N/A	N/A	N/A
TA-LC	0.50 $\pm$ 0.01	0.69 $\pm$ 0.20	0.44 $\pm$ 0.03	0.57 $\pm$ 0.01	N/A	0.57 $\pm$ 0.01	N/A	0.99 $\pm$ 0.00
GP	0.49 $\pm$ 0.02	0.82 $\pm$ 0.01	0.43 $\pm$ 0.01	0.53 $\pm$ 0.06	N/A	0.37 $\pm$ 0.01	N/A	0.41 $\pm$ 0.01
GP-CA	0.66 $\pm$ 0.01	0.82 $\pm$ 0.01	**0.45** $\pm$ **0.03**	0.70 $\pm$ 0.03	N/A	0.39 $\pm$ 0.01	0.37 $\pm$ 0.02	0.41 $\pm$ 0.01
PuSHR	**0.67** $\pm$ **0.02**	**0.94** $\pm$ **0.02**	0.42 $\pm$ 0.00	**0.72** $\pm$ **0.01**	**0.73** $\pm$ **0.01**	**0.73** $\pm$ **0.01**	**0.39** $\pm$ **0.02**	**1.07** $\pm$ **0.01**

due to tracking errors can bring collisions, and CA can help avoid this by increasing clearances. The distance gains of the GP-CA enable it to solve the challenging S4.b in contrast to GP which failed completely.

H3. PuSHR and TA-LC improve respectively over GP-CA and LC (Table I) confirming H3. We also see that PuSHR and TA-LC achieve improved Makespans (Table II) suggesting that TA enables more efficient coordination among agents. Taking a deeper look, we see that this is indeed the case: Fig. 7 shows a positive correlation between TA quality (ECBS-TA cost) and path planning quality (CL-CBS Makespan), with the lowest ECBS-TA cost coinciding with the lowest CL-CBS Makespan. Note that unlike CL-CBS, ECBS-TA uses a simplified version of the domain (no nonholonomic constraints), leading to an imperfect correlation of the costs; however the trend is clear: better task assignments lead to better plans overall. This pattern is also visible qualitatively in Fig. 6: plans generated upon TA are geometrically simpler. This simplification is helpful in more complex cases such as S3.c, where the non-TA paths are too difficult for the MPC to track properly, causing even GP-CA to fail.

H4. Thanks to TA and collision avoidance, PuSHR scales best as the only one to succeed in all scenarios, always with the top success rate. This confirms H4. Our system is slower than TA-LC in a few scenarios (S3.a, S3.b, S4.a, s4.c) but more successful as the global plan often pauses the trajectories to synchronize robots whereas TA-LC does not, thus finishing faster in simple scenarios. It is important to note that these scenarios are much simpler than e.g., S4.b, involving sparse regions, and do not require much turning (see Fig. 5).

H5. Our system achieved 100% success in the lab trials (Fig. 8), which implies that the robots were consistently able to maintain control of their blocks and push them to their goals. Thus H5 is confirmed. Compared to the simulated trials, we see that in the lab the Makespan increases, an artifact of communication, computation, and actuation overhead. The Minimum distance varies but provides a sufficient buffer to avoid collisions. We observed that the collision avoidance feature of the MPC was instrumental in making sure the robots succeeded even when unmodeled physics or communication delays caused deviations from the planned trajectories.

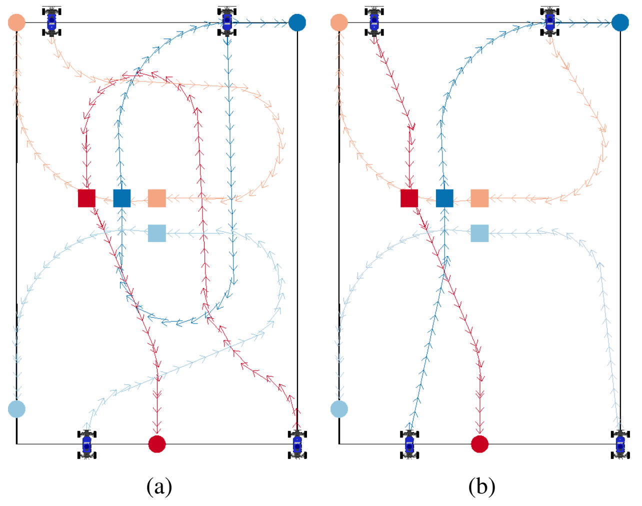


Fig. 6: Task simplification via task assignment (TA). (a) Paths planned by CL-CBS upon manual TA for scenario S4.b (median assignment quality out of 4! combinations). (b) Paths planned by CL-CBS upon *optimal* TA from ECBS-TA for the same scenario. Blocks' initial and goal locations are noted as squares and circles of the same color respectively.

VI. Discussion

Task assignment. Our findings (H3) confirmed the value of task assignment for our domain. We suspected this when we saw that a manual assignment strategy using a distance-based heuristic for task assignment would fail when distances are roughly equal (e.g., Fig. 5g). This motivated us to incorporate ECBS-TA [11] into our system which significantly boosted performance. However, the ECBS-TA implementation makes use of a Euclidian-distance-based heuristic, which is unaware of robots' kinematics or pushing constraints and does not guarantee ideal assignments as the workspace density increases. A direction for future work is to design a heuristic that efficiently incorporates such constraints.

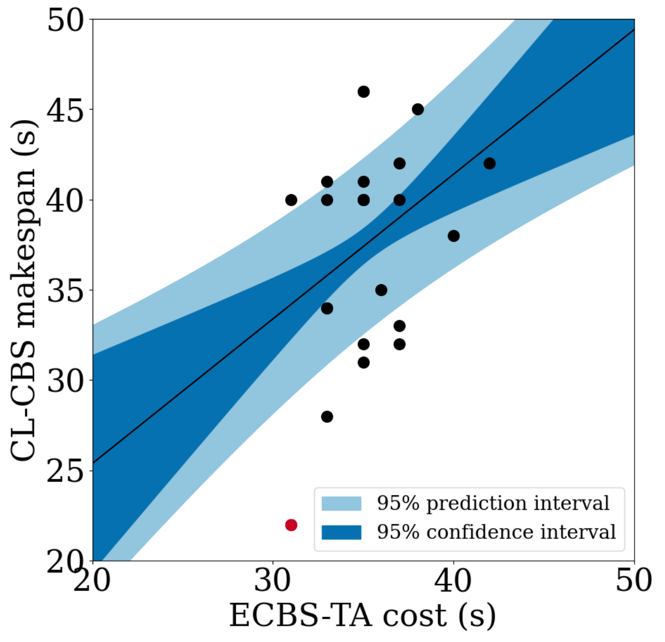


Fig. 7: TA cost (x-axis) against path quality cost (y-axis) for the $4!$ possible task assignments for scenario S4.b. The lowest makespan coincides with the lowest TA cost (red color).

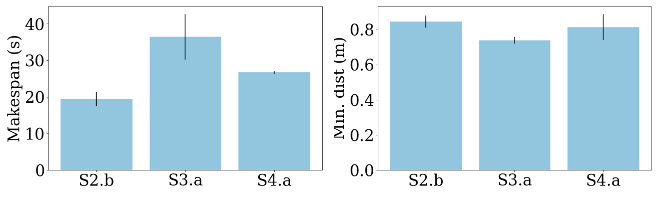


Fig. 8: Makespan and Minimum distance from lab experiments. Bars are means and errors are standard deviations over 10 trials. All trials were successful.

Tractability. As we initially experimented with ECBS-TA and CL-CBS, two CBS-driven algorithms, we explored the idea of jointly extracting task assignments and trajectories through CL-CBS. However, simultaneously iterating over multiagent trajectories and assignments using the CBS paradigm is intractable. In contrast, ECBS-TA finds the optimal task assignment very efficiently by planning in a much lower resolution (a 2-dimensional grid); then CL-CBS produces a single joint plan in the higher-resolution space that leverages the simplification induced by the advantageous assignment. For reference, planning for the S4.a scenario (Fig. 5) takes around $10min$ with the former approach whereas PuSHR takes about 10 seconds. Computations took place on a laptop with an Intel i7-10750H CPU (6 cores @ 2.6 GHz). This enabled us to scale our evaluation and run hardware experiments. However, the problem of efficiently combining task and motion planning in this domain is still interesting and worth investigating.

Quasistatics. Leveraging stable pushing [22], PuSHR completed challenging tasks under constraints imposed by robots' kinematics, their plans, and the workspace boundary without closing the loop for block control. While more fine-grained contact models [2, 20, 31] could help scale the system (e.g., dynamic pushing, irregular objects), our experiments demonstrate the value of quasistatic analytical modeling for a variety of practical problems involving contact.

References

[1] J. Alonso-Mora, R. Knepper, R. Siegwart, and D. Rus. Local motion planning for collaborative multi-robot manipulation of deformable objects. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 5495–5502, 2015.

[2] M. Bauza and A. Rodriguez. A probabilistic data-driven model for planar pushing. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3008–3015, 2017.

[3] Z. Chen, J. Alonso-Mora, X. Bai, D. D. Harabor, and P. J. Stuckey. Integrated task assignment and path planning for capacitated multi-agent pickup and delivery. *IEEE Robotics and Automation Letters*, 6(3):5816–5823, 2021.

[4] P. Culbertson, J.-J. Slotine, and M. Schwager. Decentralized adaptive control for collaborative manipulation of rigid bodies. *IEEE Transactions on Robotics*, 37(6): 1906–1920, 2021.

[5] B. Davis, I. Karamouzas, and S. J. Guy. NH-TTC: A gradient-based framework for generalized anticipatory collision avoidance. In *Proceedings of Robotics: Science and Systems (RSS)*, 2020.

[6] M. R. Dogar and S. S. Srinivasa. A planning framework for non-prehensile manipulation under clutter and uncertainty. *Autonomous Robots*, 33:217–236, 2012.

[7] J. Fink, M. A. Hsieh, and V. Kumar. Multi-robot manipulation via caging in environments with obstacles. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1471–1476, 2008.

[8] B. P. Gerkey and M. J. Matarić. A formal analysis and taxonomy of task allocation in multi-robot systems. *The International Journal of Robotics Research*, 23(9): 939–954, 2004.

[9] S. Goyal, A. Ruina, and J. Papadopoulos. Planar sliding with dry friction part 1. limit surface and moment function. *Wear*, 143(2):307 – 330, 1991.

[10] K. Harada, J. Nishiyama, Y. Murakami, and M. Kaneko. Pushing manipulation for multiple objects. *Journal of dynamic systems, measurement, and control*, 128(2): 422–427, 2006.

[11] W. Hönig, S. Kiesel, A. Tinka, J. W. Durham, and N. Ayanian. Conflict-based search with optimal task assignment. In *Proceedings of the International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, page 757–765, 2018.

[12] R. D. Howe and M. R. Cutkosky. Practical force-motion models for sliding manipulation. *The International Journal of Robotics Research*, 15(6):557–572, 1996.

[13] Y. Hu, L. Wang, J. Liang, and T. Wang. Cooperative box-pushing with multiple autonomous robotic fish in underwater environment. *IET control theory & applications*, 5(17):2015–2022, 2011.

[14] J. King, M. Klingensmith, C. Dellin, M. Dogar, P. Velagapudi, N. Pollard, and S. Srinivasa. Pregrasp manipulation as trajectory optimization. In *Proceedings of Robotics: Science and Systems (RSS)*, 2013.

[15] R. A. Knepper and D. Rus. Pedestrian-inspired sampling-based multi-robot collision avoidance. In *Proceedings of the IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 94–100, 2012.

[16] R. A. Knepper, T. Layton, J. Romanishin, and D. Rus. Ikeabot: An autonomous multi-robot coordinated furniture assembly system. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 855–862, 2013.

[17] G. A. Korsah, A. Stentz, and M. B. Dias. A comprehensive taxonomy for multi-robot task allocation. *The International Journal of Robotics Research*, 32(12): 1495–1512, 2013.

[18] J. Kottinger, S. Almagor, and M. Lahijanian. Conflict-based search for multi-robot motion planning with kinodynamic constraints. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022.

[19] S. Krivic and J. Piater. Pushing corridors for delivering unknown objects with a mobile robot. *Autonomous Robots*, 43(6):1435–1452, 2019.

[20] J. Li, W. S. Lee, and D. Hsu. Push-net: Deep planar pushing for objects with unknown physical properties. In *Proceedings of Robotics: Science and Systems*, 2018.

[21] M. Liu, H. Ma, J. Li, and S. Koenig. Task and path planning for multi-agent pickup and delivery. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2019.

[22] K. M. Lynch and M. T. Mason. Stable pushing: Mechanics, controllability, and planning. *The International Journal of Robotics Research*, 15(6):533–556, 1996.

[23] H. Ma, W. Hönig, T. K. S. Kumar, N. Ayanian, and S. Koenig. Lifelong path planning with kinematic constraints for multi-agent pickup and delivery. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019.

[24] M. T. Mason. Mechanics and planning of manipulator pushing operations. *The International Journal of Robotics Research*, 5(3):53–71, 1986.

[25] M. Mataric, M. Nilsson, and K. Simsarin. Cooperative multi-robot box-pushing. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 3, pages 556–561 vol.3, 1995.

[26] C. I. Mavrogiannis, W. B. Thomason, and R. A. Knepper. Social momentum: A framework for legible navigation in dynamic multi-agent environments. In *Proceedings of the ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, page 361–369, 2018.

[27] G. Qu, D. Brown, and N. Li. Distributed greedy algorithm for multi-agent task assignment problem with submodular utility functions. *Automatica*, 105:206–215, 2019.

[28] J. Roh, C. Mavrogiannis, R. Madan, D. Fox, and S. Srinivasa. Multimodal trajectory prediction via topological invariance for navigation at uncontrolled intersections. In *Proceedings of the Conference on Robot Learning (CoRL)*, 2020.

[29] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant. Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelligence*, 219:40–66, 2015.

[30] S. S. Srinivasa, P. Lancaster, J. Michalove, M. Schmittle, C. Summers, M. Rockett, J. R. Smith, S. Choudhury, C. Mavrogiannis, and F. Sadeghi. MuSHR: A low-cost, open-source robotic racecar for education and research. *CoRR*, abs/1908.08031, 2019.

[31] J. Stüber, C. Zito, and R. Stolkin. Let's push things forward: A survey on robot pushing. *Frontiers in Robotics and AI*, 7, 2020.

[32] A. Tsiamis, C. K. Verginis, C. P. Bechlioulis, and K. J. Kyriakopoulos. Cooperative manipulation exploiting only implicit communication. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 864–869, 2015.

[33] J. van den Berg, S. J. Guy, M. Lin, and D. Manocha. Reciprocal n-body collision avoidance. In *Robotics Research*, pages 3–19. Springer Berlin Heidelberg, 2011.

[34] C. K. Verginis, M. Mastellaro, and D. V. Dimarogonas. Robust cooperative manipulation without force/torque measurements: Control design and experiments. *IEEE Transactions on Control Systems Technology*, 28(3):713–729, 2020.

[35] G. Wagner and H. Choset. M*: A complete multirobot path planning algorithm with performance bounds. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3260–3267, 2011.

[36] Z. Wang and M. Schwager. Force-amplifying n-robot transport system (force-ants) for cooperative planar manipulation without communication. *The International Journal of Robotics Research*, 35(13):1564–1586, 2016.

[37] L. Wen, Y. Liu, and H. Li. CL-MAPF: Multi-agent path finding for car-like robots with kinematic and spatiotemporal constraints. *Robotics and Autonomous Systems*, 150:103997, 2022.

[38] P. R. Wurman, R. D'Andrea, and M. Mountz. Coordinating hundreds of cooperative, autonomous vehicles in warehouses. *AI Magazine*, 29(1), 2008.

[39] W. Yuan, J. A. Stork, D. Kragic, M. Y. Wang, and K. Hang. Rearrangement with nonprehensile manipulation using deep reinforcement learning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 270–277, 2018.

[40] J. Zhou, M. T. Mason, R. Paolini, and D. Bagnell. A convex polynomial model for planar sliding mechanics: theory, application, and experimental validation. *The International Journal of Robotics Research*, 37(2-3): 249–265, 2018.