Planning for Complex Non-prehensile Manipulation Among Movable Objects by Interleaving Multi-Agent Pathfinding and Physics-Based Simulation

Dhruv Mauria Saxena[1] and Maxim Likhachev[1]

***Abstract*— Real-world manipulation problems in heavy clutter require robots to reason about potential contacts with objects in the environment. We focus on pick-and-place style tasks to retrieve a target object from a shelf where some 'movable' objects must be rearranged in order to solve the task. In particular, our motivation is to allow the robot to reason over and consider non-prehensile rearrangement actions that lead to complex robot-object and object-object interactions where multiple objects might be moved by the robot simultaneously, and objects might tilt, lean on each other, or topple. To support this, we query a physics-based simulator to forward simulate these interaction dynamics which makes action evaluation during planning computationally very expensive. To make the planner tractable, we establish a connection between the domain of Manipulation Among Movable Objects and Multi-Agent Pathfinding that lets us decompose the problem into two phases our M4M algorithm iterates over. First we solve a multi-agent planning problem that reasons about the configurations of movable objects but does not forward simulate a physics model. Next, an arm motion planning problem is solved that uses a physics-based simulator but does not search over possible configurations of movable objects. We run simulated and real-world experiments with the PR2 robot and compare against relevant baseline algorithms. Our results highlight that M4M generates complex 3D interactions, and solves at least twice as many problems as the baselines with competitive performance.**

I. Introduction

Manipulation Among Movable Objects (MAMO) [1] defines a broad class of problems where a robot must complete a manipulation task in the presence of obstructing clutter. In heavily cluttered scenes, there may be no collision-free trajectory that solves the task. This does not make the problem unsolvable since MAMO allows rearrangement of some objects *a priori* designated as 'movable'. In addition, MAMO may associate each object with constraints on how it can be interacted with – it is undesirable to allow robots to carelessly push or throw objects around.

In this paper, we consider MAMO problems for pick-and-place manipulation tasks where the robot needs to retrieve a target object from a cluttered shelf, cabinet, fridge, or a similar structure. Fig. 1 (a) shows an example of such a scene where two movable objects must be rearranged in order to retrieve the desired object, while ensuring they do not topple and no contacts are made with an immovable obstacle.

Solving such MAMO problems requires answers to three difficult questions: *which* objects to move, *where* to move them, and *how* to move them. Thus MAMO problems assign the robot a goal with respect to the overall task and object-of-interest (OoI), without any additional goal specifications for other objects except for satisfying their associated interaction constraints; while MAMO solutions exist in a composite configuration space that includes the configuration of the robot arm and all objects in the scene. The search for a solution is computationally challenging since the size of this space grows exponentially with the number of objects.

[1]The authors are with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA. e-mail: {dsaxena, mlikhach}@andrew.cmu.edu

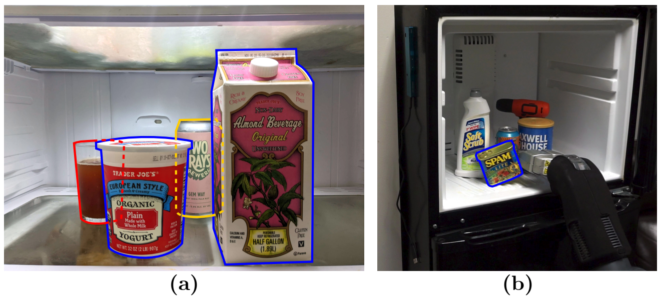

Fig. 1. (a) An example MAMO problem to retrieve the beer can (yellow outline). Access is blocked by the movable box of milk and tub of yogurt (blue outlines). In order to retrieve the can, they must be rearranged out of the way without toppling them, and without anything making contact with the glass of juice (red outline). (b) A complex non-prehensile action that tilts the movable potted meat can (blue outline) to rearrange it.

We are interested in non-prehensile rearrangement actions since they allow robots to manipulate objects that may be too big or too bulky or otherwise ungraspable. In many cases it is more time- and energy-efficient to push an object off to the side than to grasp it, pick it up, move it elsewhere, place it down, and release it before proceeding. Furthermore, we allow the robot to move multiple objects simultaneously with the same push action, and we allow objects to tilt, lean on each other, and slide (an example is shown in Fig. 1 (b)). Planning with these actions requires the ability to predict the effect of robot actions on the configuration of objects, typically through computationally expensive forward simulations of a rigid-body physics simulator.

Our key insight in this work draws a connection between the MAMO domain and Multi-Agent Pathfinding (MAPF) to decompose the problem into two parts. First, we treat the movable objects as artificially actuated agents tasked with avoiding collisions with (i) our robot arm retrieving the OoI, (ii) each other, and (iii) immovable obstacles.

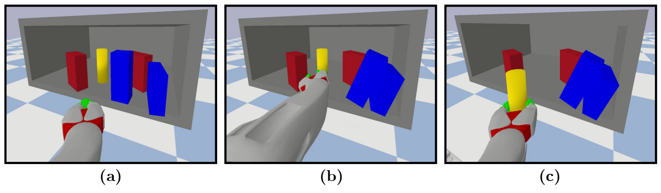

Fig. 2. Sequence of images showing a solution found by our M4M algorithm for a simple MAMO scene. From *left* to *right*: (a) initial scene, (b) rearranged scene after one push action, (c) successful OoI retrieval. Movable objects are blue, immovable obstacles are red, and the OoI is yellow.

A solution to this abstract MAPF problem searches over potential rearrangements of objects *without* the need to query a physics simulator. Next, we use the MAPF solution to compute informed push actions to rearrange movable objects *without* searching over their possible configurations. These actions are forward simulated with a physics model to ensure validity. The decomposition helps us keep track of object configurations in the full $SE(3)$ space and generate informed push actions that lead to realistic multi-body interactions in the 3D workspace as shown in Fig. 1 (b). Fig 2 shows a complex and interesting solution found by our algorithm for one of the simpler scenarios in our test data.

The main contributions of our work in this paper for solving MAMO planning problems are:

- Enable reasoning over and usage of complex non-prehensile interactions that may push multiple objects in tandem and produce object-object interactions like leaning and toppling (Fig. 1 (b)).
- MAPF abstraction for computing suitable rearrangements for MAMO planning problems, without using a simulation-based model.
- An efficient algorithm to solve MAMO problems that iterates between calls to an MAPF solver (to determine *which* objects to move *where*) and a push planner (to verify *how* to move the objects).
- A thorough experimental evaluation of our approach in simulation and in the real-world on a PR2 robot.

We provide details of relevant works from MAMO literature in Section II. Section III formalises the MAMO planning problem. Section IV presents our iterative planning algorithm M4M, including the abstraction from MAMO to MAPF (Section IV-A) and a non-prehensile push planner (Section IV-B). We provide extensive quantitative evaluation against relevant MAMO baselines in simulation in Section V along with real-world results of our algorithm on the PR2 robot. Section VI discusses the benefits, limitations, and future extensions of this work.

II. Related Work

MAMO generalises Navigation Among Movable Obstacles (NAMO) where a mobile robot must navigate from start to goal in a reconfigurable environment [2]–[4]. It is also related to the rearrangement planning problem [5], [6] which explicitly specifies desired goal configurations for movable objects. Wilfong [3] showed that rearrangement planning is PSPACE-hard, and MAMO problems are NP-hard to solve.

Many existing MAMO and *rearrangement planning* solvers make use of prehensile actions [1], [7]–[11]. This simplifies planning since grasped objects behave as rigid bodies attached to the robot, but assumes access to known stable configurations of and grasp poses for objects [1], [7]–[9]. In some cases a "buffer" location to place grasped objects is required [10], [11]. In particular, [7] and [9] utilise the concept of "pebble graphs" [12], [13] from MAPF literature to find prehensile actions for rearrangement planning. Their formulation restricts the motion of the movable objects (pebbles) on a precomputed roadmap of robot arm trajectories via prehensile actions. This limits the possible configurations of objects they consider since motions are limited to poses from where they can be grasped and to those where they can be stably placed. Since we utilise non-prehensile pushes for rearrangement and a physics-based simulator for action validation, our planner explores a richer space of robot-object and object-object interactions in the 3D workspace.

Allowing *non-prehensile interactions* with objects typically requires access to a simulation model to obtain the result of complex interaction dynamics [14]–[19]. Of these approaches, only Selective Simulation [19] considers realistic interactions in the 3D workspace and is one of our comparative baselines in Section V. Others rely on planar robot-object interactions which fail to account for object dynamics in $SE(3)$ where they might tilt, lean, or topple. In Section V, we adapt the MAMO solver from [15] to use our push actions that lead to 3D robot-object interactions and require a physics simulator during planning. Originally their work was limited to interacting with a single object at a time, and used an analytical motion model in $SE(2)$ to propagate the effect of the push on the planar configuration of the object being pushed (tilting and toppling was not considered in [14]–[18]).

Querying *physics-based simulators* for the result of an action is much more expensive than collision checking it. KPIECE [20] is a randomised algorithm for planning with a computationally expensive transition model (querying a physics-based simulator is an example of such a model). KPIECE and RRT [21] are two other baselines we compare against in Section V. In our own prior work on MAMO planning [22], we find a collision-free trajectory to a region near the OoI grasp pose, and simulate goal-directed non-prehensile actions only within this region. The assumption that such a collision-free trajectory exists is easily violated in the cluttered MAMO workspaces we instantiate in our experiments (see Figs. 1 (a), 6, and 7 for example).

III. Problem Statement

Let $\mathcal{X}_\mathcal{R} \subset \mathbb{R}^q$ denote the configuration space of a q degrees-of-freedom robot manipulator $\mathcal{R}$. Let $\mathcal{O} = \{O_1, \dots, O_n\}$ be the set of objects in the scene, and $\mathcal{X}_{O_i} \equiv SE(3)$ be the configuration space of object O_i that includes its 3D position and orientation. The search space for a MAMO planning problem is $\mathcal{X} = \mathcal{X}_\mathcal{R} \times \mathcal{X}_{O_1} \times \cdots \times \mathcal{X}_{O_n}$.

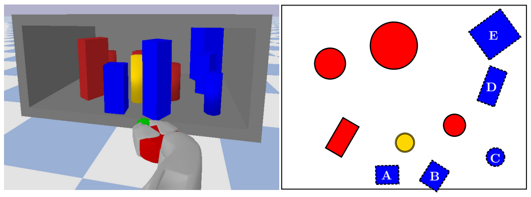

Fig. 3. MAMO workspace (*left*) and its 2D projection labelled with movable object IDs. Movable objects are in blue, immovable obstacles in red, and the object-of-interest to be retrieved in yellow.

We denote movable objects by $\mathcal{O}_M$ and immovable obstacles by $\mathcal{O}_I$ such that $\mathcal{O} = \mathcal{O}_M \cup \mathcal{O}_I$ and $\mathcal{O}_M \cap \mathcal{O}_I = \emptyset$.

Each object is associated with a set of interaction constraints. For example, an 'immovable' obstacle (an object that cannot be interacted with, such as a wall) will contain a constraint function which is satisfied so long as neither the robot nor any other object makes contact with it. In our problems similar functions encode that movable objects cannot fall off the shelf, tilt too far (beyond $25°$), or move with a high instantaneous velocity (above $1\,\mathrm{m\,s^{-1}}$). A state $x \in \mathcal{X}$ is valid if all constraints for all objects are satisfied at that state. Let $\mathcal{X}_V$ be the space of valid states.

A MAMO planning problem can be defined with the tuple $\mathcal{P} = (\mathcal{X}, \mathcal{A}, \mathcal{T}, c, x_S, \mathcal{X}_G)$. $\mathcal{A}$ is the action space of the robot, $\mathcal{T} : \mathcal{X} \times \mathcal{A} \to \mathcal{X}$ is a deterministic transition function, $c : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_{\geq 0}$ is a state transition cost function, $x_S \in \mathcal{X}_V$ is the start state, and $\mathcal{X}_G \subset \mathcal{X}, \mathcal{X}_G \cap \mathcal{X}_V \neq \emptyset$ is the set of goal configurations. The start state x_s includes a "home" robot configuration in $\mathcal{X}_\mathcal{R}$ and the initial poses of all objects. We would like to find the least-cost valid path π^* from start to goal i.e., a path made up of a sequence of valid states. Formally, we can write this as:

$$\begin{aligned}
&\text{find } \pi^* = \underset{\pi=\{x_1,\dots,x_T\}}{\operatorname{argmin}} \sum_{i=1}^{T-1} c(x_i, x_{i+1}) \\
&\text{s.t. } x \in \mathcal{X}_V, \forall\, x \in \pi \qquad \textit{(path of valid states)} \\
&\quad x_1 = x_S, x_T \in \mathcal{X}_G \qquad \textit{(start, goal constraints)} \\
&\quad x_{i+1} = \mathcal{T}(x_i, a_i), a_i \in \mathcal{A}, \forall x_i, x_{i+1} \in \pi \\
&\qquad\qquad \textit{(transition dynamics)}
\end{aligned}$$

In our work we discretise $\mathcal{A}$ to include "simple motion primitives" that independently change each robot joint angle by a fixed amount and dynamically generated "push actions" described in Section IV-B. For transition $x_{i+1} = \mathcal{T}(x_i, a_i)$, action $a_i \in \mathcal{A}$ can affect object configurations between x_i and x_{i+1} only if a_i is a push action or the OoI has been grasped. The cost of robot actions c is proportional to the distance travelled in $\mathcal{X}_\mathcal{R}$. We assume $\mathcal{X}_G$ is defined in two parts – a grasp pose in $SE(3)$ for the OoI and a goal pose in $SE(3)$ where it must end up (while grasped by the robot). Our solution to MAMO problems is a sequence of arm trajectories in the robot configuration space $\mathcal{X}_\mathcal{R} \subset \mathbb{R}^q$ ($q = 7$ for the PR2 robot) that (i) rearrange movable clutter and (ii) retrieve the OoI. Fig. 3 shows an example of the MAMO problems we consider in this paper, along with its 2D projection. Red objects are immovable obstacles $\mathcal{O}_I$, blue objects are initial movable objects $\mathcal{O}_M^{\text{init}}$, and the goal for the robot arm is to extract the yellow OoI from the shelf. There is no collision-free trajectory for the arm to extract the OoI from the shelf. Upon rearrangement of some movable objects (A and B in particular), such a trajectory may be found.

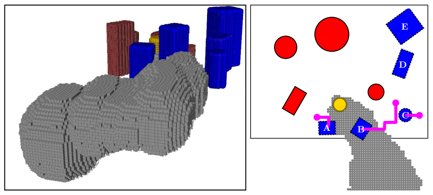

Fig. 4. The negative goal region (NGR) $\mathcal{V}(\hat{\pi}_\mathcal{R})$ in gray for the MAMO problem from Fig. 3. (*left*) 3D volumes of the NGR and all objects at their initial poses (we omit the shelf for ease of visualisation). (*right*) 2D projection of the NGR and the workspace, overlayed with the solution to the abstract MAPF problem from Section IV-A formulated for this scene. Objects A and B need to move outside the NGR, and object C needs to move to allow A to reach its goal. MAPF solution paths are shown in pink.

IV. The M4M Planning Algorithm

We call our algorithm M4M: Multi-Agent Pathfinding for Manipulation Among Movable Objects. M4M is given access to a physics-based simulator (PyBullet [23]) to ensure that no interaction constraints defined in the MAMO problem are violated. We note that a MAMO problem $\mathcal{P}$ to retrieve the OoI with $\mathcal{O}_M \neq \emptyset$ is solvable *iff* the simpler problem $\hat{\mathcal{P}}$ without any movable objects i.e., $\mathcal{O}_M = \emptyset$ can be solved. We denote a solution trajectory to $\hat{\mathcal{P}}$ as $\hat{\pi}_\mathcal{R}$. Let $\mathcal{V}(\hat{\pi}_\mathcal{R})$ denote the volume occupied by the robot arm in the workspace during execution of $\hat{\pi}_\mathcal{R}$. $\mathcal{V}(\hat{\pi}_\mathcal{R})$ specifies a "negative goal region" (NGR) [15] for the movable objects. A NGR is a sufficient volume of the 3D workspace which, if there are no objects inside it, allows the robot arm to retrieve the OoI without other contacts. If all movable objects can be rearranged such that they are outside $\mathcal{V}(\hat{\pi}_\mathcal{R})$, the robot can execute $\hat{\pi}_\mathcal{R}$ to retrieve the OoI. Fig. 4 shows a NGR $\mathcal{V}(\hat{\pi}_\mathcal{R})$ for the problem from Fig. 3.

Algorithm 1 contains the pseudocode for M4M. At a high-level, M4M first computes $\hat{\pi}_\mathcal{R}$ (Line 4) and the NGR $\mathcal{V}(\hat{\pi}_\mathcal{R})$ (Line 5). It then iterates over two steps:

1) Section IV-A: Compute a solution to the abstract Multi-agent Pathfinding (MAPF) problem where each movable object is treated as an agent that needs to escape the NGR without colliding with other agents using Conflict-Based Search (CBS) [24], a complete and optimal MAPF algorithm.
2) Section IV-B: Pick a movable object to be rearranged according to the MAPF plan computed in 1 and find a valid non-prehensile push for it by forward simulating potential pushes using a physics-based simulator.

Algorithm 1 uses `replan` to ensure CBS is only called to solve new MAPF problems. After the first CBS call, `replan` triggers subsequent CBS calls once a valid push has been found i.e., at least one object has been moved. This leads to a different MAPF problem with new object poses.

Algorithm 1 Multi-Agent Pathfinding for Manipulation Among Movable Objects

```
1: procedure M4M($\mathcal{O}_M^{\text{init}}, \mathcal{O}_I$)
2:     $\mathcal{O}_M \leftarrow \mathcal{O}_M^{\text{init}}$                ▷ Rearranged object positions
3:     $\Psi \leftarrow \emptyset$                  ▷ Sequence of arm trajectories
4:     $\hat{\pi}_{\mathcal{R}} \leftarrow$ PLANRETRIEVAL($\mathcal{O}_I$)     ▷ OoI retrieval trajectory
5:     Compute $\mathcal{V}(\hat{\pi}_{\mathcal{R}})$
6:     replan ← true, done ← false
7:     while time remains do
8:         if replan then
9:             $\pi_{\mathcal{R}} \leftarrow$ PLANRETRIEVAL($\mathcal{O}_I \cup \mathcal{O}_M$)
10:            if $\pi_{\mathcal{R}}$ exists then
11:                $\Psi \leftarrow \Psi \cup \{\pi_{\mathcal{R}}\}$, done ← true
12:                break
13:            $\{\pi_{o_m}\}_{o_m \in \mathcal{O}_M} \leftarrow$ CBS($\mathcal{O}_M, \mathcal{O}_I, \mathcal{V}(\hat{\pi}_{\mathcal{R}})$)
14:            replan ← false
15:        for $o_m \in \mathcal{O}_M$ do
16:            if $\pi_{o_m} = \emptyset$ then
17:                continue
18:            $\psi \leftarrow$ PLANPUSH($o_m, \pi_{o_m}, \mathcal{O}_M, \mathcal{O}_I$)
19:            (valid, $o'_m$) ← SIMULATEPUSH($\psi$)
20:            if valid then
21:                $\Psi \leftarrow \Psi \cup \{\psi\}$, replan ← true
22:                UPDATEPOSE($\mathcal{O}_M, o'_m$)
23:                break
24:    if ¬done then
25:        return $\emptyset$
26:    return $\Psi$
```

Until a valid push is found, we sample and simulate pushes for all objects that move in the MAPF solution.

The PLANRETRIEVAL function takes as input a set of objects to be considered as immovable obstacles for the robot and runs Multi-Heuristic A* [25] to find an arm trajectory in $\mathcal{X}_{\mathcal{R}}$ to retrieve the OoI.

CBS is called in Line 13 with the latest known movable object poses in $SE(3)$ to obtain a set of paths that ensure they all satisfy the NGR $\mathcal{V}(\hat{\pi}_{\mathcal{R}})$. This searches over all possible rearrangements of the scene from the current state, without ever querying a physics simulator, by assuming that movable objects are artificially actuated agents (Section IV-A).

We then loop over all objects that need to be rearranged (from Line 15) and try and find a valid push for them (Section IV-B). If a valid push is found (Line 20), it is added to the final sequence of arm trajectories to be executed Ψ, and the pose of that object is updated for future iterations.

M4M terminates either when the allocated planning budget expires, or we successfully find a trajectory to retrieve the OoI in the presence of all objects ($\mathcal{O}_I \cup \mathcal{O}_M$) as obstacles in Line 9. Although this trajectory $\pi_{\mathcal{R}}$ may be different from $\hat{\pi}_{\mathcal{R}}$ (Line 4), it will still retrieve the OoI successfully since it is guaranteed to not make contact with any object (immovable or movable). The sequence of trajectories Ψ can then be executed in order to rearrange the movable objects (if required) and finally ending in successful OoI retrieval.

A. MAPF Abstraction for Manipulation

A fundamental challenge to solving MAMO problems requires determining *which* objects need to be rearranged and *where* they should be moved. The key idea in this paper uses an existing MAPF solver to search over potential rearrangements of the scene which lead to successful OoI retrieval. Importantly, the MAPF solver does not require access to a physics simulator for this purpose – it only relies on 3D collision checking. Our MAPF abstraction includes all movable objects $o_m \in \mathcal{O}_M$ as agents. We check for collisions between agents in space and time in their full $SE(3)$ configuration space. All agents have a discrete action space corresponding to a four-connected grid on the $(x, y)-$plane of the shelf. We assume each action takes unit time and either the agent remains in place, or the $x-$ or $y-$coordinate of the agent pose changes by $1\,\text{cm}$.

Agent start configurations are determined by their latest pose in $SE(3)$ prior to the MAPF call (Algorithm 1, Line 13). Each agent o_m in the MAPF problem has a set of possible goals that include all states where the agent satisfies the NGR by being "outside" it.

We call CBS to obtain a solution, shown in Fig. 4, to this MAPF abstraction. The solution is a set of paths for movable objects $\{\pi_{o_m}\}_{o_m \in \mathcal{O}_M}$ whose final states $\pi_{o_m}^{end}$ satisfy the NGR, and suggests a rearrangement strategy in terms of *which* objects to move and *where*. If we can rearrange all $o_m \in \mathcal{O}_M$ to their respective $\pi_{o_m}^{end}$ poses, we know that the trajectory $\hat{\pi}_{\mathcal{R}}$ will successfully retrieve the OoI, thereby solving the MAMO problem.

B. Generating Non-Prehensile Push Actions

Given a path π_{o_m} for $o_m \in \mathcal{O}_M$ from the MAPF solution, PLANPUSH (Algorithm 1, Line 18) determines *how* an object may be rearranged (Fig. 5). We would like to move the object to $\pi_{o_m}^{end}$, which is known to satisfy the NGR. To compute a push trajectory, we first shortcut π_{o_m} (taking into account collisions with immovable obstacles $\mathcal{O}_I$) into a series of straight line segments defined by points $\{x^1 = \pi_{o_m}^{start}, \ldots, x^n = \pi_{o_m}^{end}\}$. We also compute the point of intersection x^{aabb} of the ray from x^1 along the direction $\overrightarrow{(x^2, x^1)}$ with the axis-aligned bounding box of o_m.

PLANPUSH computes a collision-free path between successive pushes by planning in $\mathcal{X}_{\mathcal{R}}$ with all objects $\mathcal{O}_I \cup \mathcal{O}_M$ as obstacles to a point x_{push}^0 sampled around x^{aabb}[1]. If this path is found, PLANPUSH similarly samples points x_{push}^i around each x^i in the shortcut path. It runs inverse kinematics (IK) in sequence for each segment of the push action between points $\left(x_{\text{push}}^{i-1}, x_{\text{push}}^i\right), i = \{1, \ldots, n\}$. If all IK calls succeed, we return the full push trajectory by concatenating π_0 with all push action segments.

This push action, informed by the MAPF solution about *which* object to move *where*, is forward simulated with a physics model to verify whether it satisfies all interaction constraints for all objects. If so, it is queued into the sequence of rearrangements that will be executed as part of the MAMO solution returned by M4M (Algorithm 1, Line 20).

[1]We sample (x, y) coordinates for x_{push}^0 from $\mathcal{N}(x^{aabb}, \sigma I)$, $\sigma = 2.5\,\text{cm}$. The $z-$coordinate is fixed at $3\,\text{cm}$ above the shelf for the entire push action.

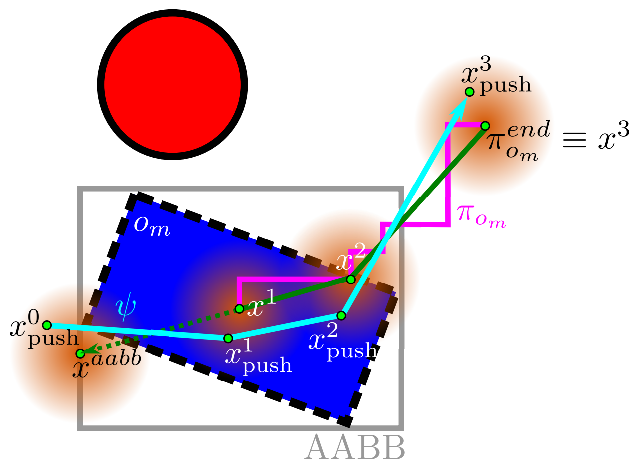

Fig. 5. 2D illustration of our push planner. Given a movable object o_m (blue) and its MAPF solution path π_{o_m} (pink), we shortcut π_{o_m} while accounting for immovable obstacles $\mathcal{O}_I$ (red) to get the green path of straight line segments. After computing x^{aabb} by intersecting the $\overrightarrow{(x^2, x^1)}$ ray with the axis-aligned bounding box for o_m, the push action (cyan) is computed via inverse kinematics between sampled points $x^i_{\text{push}} \sim \mathcal{N}(x^i, \sigma I)$, $i = \{0, \dots, n\}$, $x^0 := x^{aabb}$.

V. Experimental Results

A. Simulation Experiments

We run our simulation experiments in MAMO workspaces of three difficulty levels shown in Fig. 6. Each workspace has one OoI (yellow), four immovable obstacles (red), and different numbers of movable objects (blue). Objects are cylinders and cuboids with random sizes, initial poses, masses, and coefficients of friction. We assume perfect knowledge of the initial workspace state and all object parameters. We set a planning timeout of 120 s for 100 randomly generated MAMO problems at each level. Our analysis includes two versions of our algorithm – M4M refers to Algorithm 1, and $\widehat{\text{M4M}}$ refers to a version which only calls CBS once (after Line 5) and does not iterate between calling CBS and finding a valid push in simulation.

Baselines: We compare the performance of M4M against three types of baselines for solving MAMO problems with non-prehensile interactions. The first are standard implementations of sampling-based algorithms KPIECE [20] and RRT [21] from OMPL [26] that search the entire MAMO state space $\mathcal{X}$ by randomly sampling robot motions.

The second baseline, Selective Simulation [19] (SelSim), is a search-based algorithm that interleaves a 'planning' phase and a 'tracking' phase. The former queries the physics-based simulator for interactions with a set of 'relevant' movable objects identified so far. The latter executes the solution found by the planning phase in the presence of all objects in simulation and, if any interaction constraints are violated, it adds the 'relevant' object to the set. It only uses the simple motion primitives described in Section III.

Our final baseline is the work from Dogar et al. [15] (Dogar) which introduced the idea of a negative goal region (NGR) we use in M4M. Dogar recursively searches for a

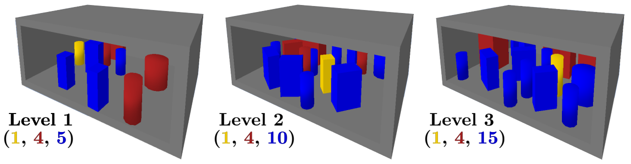

Fig. 6. MAMO problems of differing complexity. From *left* to *right*, Levels 1, 2, and 3 have 5, 10, and 15 movable objects respectively. Each Level has 1 OoI and 4 immovable obstacles.

solution backwards in time, similar to [1]. It first finds an OoI retrieval trajectory ignoring all movable objects. The NGR induced by this trajectory helps identify a set of objects to be rearranged, and the OoI is added as an obstacle. If an object is successfully rearranged, the NGR and set of objects still to be rearranged are updated with the trajectory found, and the rearranged object is added as an obstacle at its initial pose. This process continues until no further objects need to be rearranged. Our implementation of Dogar finds the same OoI retrieval trajectory as M4M, and uses the same push actions (Section IV-B) to try and rearrange objects. Notably, Dogar only has information about *which* objects to move but not *where* to move them. Our implementation finds the closest cell outside the latest NGR for an object and samples points around this location to try to move the object towards.

Results: Table I shows the result of our experiments where we present the *min/median/max* values for total planning time and simulation time of successful runs only. Experiments were run on a 4 GHz Intel i7-4790K CPU with 28 GB 1600 MHz DDR3 RAM.

Both versions of M4M solve the most problems across all difficulty levels. For Levels 1, 2, and 3, the M4M solution successfully executed 0.8, 1.9, and 3.1 push actions on average. The difference in performance between M4M and $\widehat{\text{M4M}}$ highlights the benefit of the iterative nature of M4M. Since MAPF paths are usually not precisely replicated in simulation via pushes, querying the solver repeatedly with an updated workspace configuration leads to more informed future paths for objects, instead of trying to forcibly push them to the first goal configuration suggested by MAPF.

All baseline algorithms from Table I suffer due to poor exploration over the space of rearrangements. Our approach benefits from the MAPF abstraction to produce guidance on *where* to move each object to free up the NGR. The stochastic sampling of push actions used by our push planner leads to complex, multi-body non-prehensile interactions that satisfy interaction constraints in the final solution. In contrast Dogar naively samples pushes to be simulated, and necessarily tries to ensure there is no overlap between the NGR and movable objects, even if a slightly different collision-free path can be found to retrieve the OoI (Algorithm 1, Line 9). This strategy suffers when sampled points are near immovable obstacles, and limits the possible rearrangements considered since movable objects that are rearranged successfully are treated as immovable obstacles. Dogar also never executes a potential trajectory until there is no overlap between the NGR and movable objects, unlike SelSim

TABLE I

SIMULATION STUDY FOR MAMO PLANNING IN CLUTTERED SCENES - SUCCESS RATES AND *min/median/max* PLANNING AND SIMULATION TIMES

Metrics	Level	Planning Algorithms					
		M4M	$\widehat{\text{M4M}}$	DOGAR [15]	SELSIM [19]	KPIECE [20]	RRT [21]
Success Rate (%)	1	92	79	40	33	48	55
	2	73	54	20	21	33	40
	3	62	36	6	16	17	26
Total Planning Time (s)	1	1.0 / 2.6 / 102.5	1.0 / 2.4 / 103.8	0.1 / 0.9 / 115.3	0.004 / 0.02 / 0.03	7.4 / 23.4 / 117.8	7.1 / 15.8 / 101.4
	2	1.2 / 6.6 / 115.4	1.3 / 2.6 / 100.3	0.3 / 0.5 / 113.5	0.002 / 0.008 / 0.2	9.3 / 28.2 / 112.0	8.6 / 27.6 / 104.9
	3	1.3 / 7.2 / 116.1	1.6 / 2.4 / 72.6	0.2 / 0.4 / 55.0	0.004 / 0.01 / 0.03	10.6 / 32.0 / 98.5	10.3 / 26.7 / 113.4
Simulation Time (s)	1	0 / 0 / 58.6	0 / 0 / 20.1	0 / 0 / 42.0	27.3 / 35.0 / 43.6	0 / 10.6 / 99.0	0 / 4.4 / 87.2
	2	0 / 0.4 / 75.9	0 / 0 / 37.0	0 / 0 / 20.9	36.7 / 44.1 / 58.3	0 / 16.1 / 95.4	0 / 16.7 / 83.7
	3	0 / 0.4 / 55.1	0 / 0 / 24.3	0 / 0 / 20.0	47.3 / 55.7 / 76.0	0 / 18.3 / 79.3	0 / 15.3 / 101.2

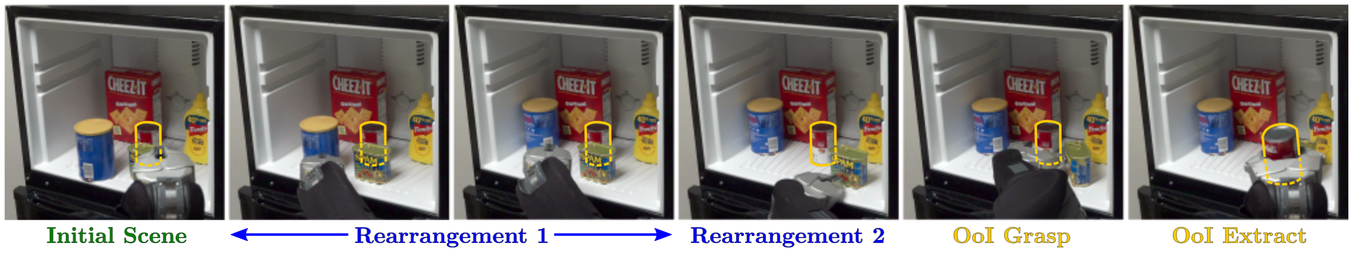

Fig. 7. A MAMO solution generated by M4M. The tomato soup can (yellow outline) is the OoI, all other objects are movable.

which simulates all trajectories found during planning. In fact, all SELSIM successes in Table I correspond to scenes where the very first planned trajectory succeeds in OoI retrieval in simulation. This is only true when there is minimal overlap between the NGR and movable objects. When any movable object needs to be rearranged, SELSIM suffers from its poor action space – the simple motion primitives are ineffective at causing meaningful robot-object interactions in the workspace. KPIECE and RRT benefit significantly from goal biasing in simpler scenes where either little to no robot-object interactions are required or the objects that need to be moved have nice physical properties (large supporting footprint, low center-of-mass, low coefficient of friction).

B. *Real-World Performance on the PR2*

We ran M4M on a PR2 robot where we used a refrigerator compartment as our MAMO workspace (Fig. 7). We placed five objects from the YCB Object Dataset [27] in the refrigerator. Four of these were movable and the tomato soup can was the object-of-interest. Objects were localised using a search-based algorithm [28] run on a NVidia Titan X GPU. We gave M4M a total planning timeout of 120 s.

Out of 16 perturbations of the initial scene from Fig. 7, 12 runs successfully retrieved the OoI. Across the successful runs the planner took 56.41 ± 27.29 s to compute a plan of which 49.26 ± 24.21 s was spent simulating pushes. Failures were due to interaction constraints being violated during execution by the PR2. Since M4M returns a solution that does not violate constraints in simulation, failures are due to modelling errors between the simulator and the real-world. Specifically, accurately computing coefficients of friction is difficult and can lead to differing contact mechanics in simulation than the real-world. Fig. 7 shows the solution to a MAMO problem being executed by the PR2. It moves the coffee can out of the way, pushes the potted meat can slightly aside, and finally the OoI (tomato soup can) is extracted while also nudging the potted meat can.

VI. CONCLUSION AND DISCUSSION

This paper presents M4M: Multi-Agent Pathfinding for Manipulation Among Movable Objects, an algorithm to plan for manipulation in heavy clutter that considers complex interactions such as rearranging multiple objects simultaneously, and tilting, leaning and sliding objects. M4M uses a MAPF abstraction to the MAMO problem to find suitable rearrangements, and a non-prehensile push planner to realise these rearrangements by utilising complex multi-body interactions. It dramatically outperforms alternative approaches that do not reason about such interactions efficiently.

M4M greedily commits valid pushes found to its sequence of rearrangement trajectories. This greedy behaviour makes M4M incomplete, given that it has no ability to backtrack from this decision. In the future we hope to address this incompleteness of M4M by developing an algorithm that considers (i) all feasible pushes for an object that needs to be rearranged to a specific location, (ii) all orderings of all feasible push actions to realise a particular rearrangement for a set of objects, and (iii) all possible rearrangements for a set of objects. Additionally, the MAPF solver used in M4M should be modified to use a cost function which has information about robot kinematics and pushing dynamics so as to compute and thus simulate better push actions.

References

[1] M. Stilman, J. Schamburek, J. Kuffner, and T. Asfour, "Manipulation planning among movable obstacles," in *2007 IEEE International Conference on Robotics and Automation, ICRA*. IEEE, 2007.

[2] R. Alami, J.-P. Laumond, and T. Simeon, "Two manipulation planning algorithms," in *WAFR Proceedings of the workshop on Algorithmic foundations of robotics* , K. Goldberg, D. Halperin, J.-C. Latombe, and R. Wilson, Eds. A. K. Peters, Ltd. Natick, MA, USA, 1994, pp. 109–125.

[3] G. T. Wilfong, "Motion planning in the presence of movable obstacles," *Ann. Math. Artif. Intell.*, 1991.

[4] M. Stilman and J. J. Kuffner, "Navigation among movable obstacles: Real-time reasoning in complex environments," *Int. J. Humanoid Robotics*, vol. 2, no. 4, pp. 479–503, 2005.

[5] O. Ben-Shahar and E. Rivlin, "Practical pushing planning for rearrangement tasks," *IEEE Trans. Robotics Autom.*, vol. 14, no. 4, pp. 549–565, 1998.

[6] J. Ota, "Rearrangement planning of multiple movable objects by a mobile robot," *Adv. Robotics*, vol. 23, no. 1-2, pp. 1–18, 2009.

[7] A. Krontiris, R. Shome, A. Dobson, A. Kimmel, and K. E. Bekris, "Rearranging similar objects with a manipulator using pebble graphs," in *14th IEEE-RAS International Conference on Humanoid Robots, Humanoids 2014, Madrid, Spain, November 18-20, 2014*. IEEE, 2014, pp. 1081–1087.

[8] A. Krontiris and K. E. Bekris, "Dealing with difficult instances of object rearrangement," in *Robotics: Science and Systems XI, Sapienza University of Rome, Rome, Italy, July 13-17, 2015*, L. E. Kavraki, D. Hsu, and J. Buchli, Eds., 2015.

[9] R. Shome and K. E. Bekris, "Synchronized multi-arm rearrangement guided by mode graphs with capacity constraints," in *Algorithmic Foundations of Robotics XIV, Proceedings of the Fourteenth Workshop on the Algorithmic Foundations of Robotics, WAFR 2021, Oulu, Finland, June 21-23, 2021*, ser. Springer Proceedings in Advanced Robotics, S. M. LaValle, M. Lin, T. Ojala, D. A. Shell, and J. Yu, Eds., vol. 17. Springer, 2021, pp. 243–260.

[10] R. Wang, K. Gao, D. Nakhimovich, J. Yu, and K. E. Bekris, "Uniform object rearrangement: From complete monotone primitives to efficient non-monotone informed search," in *International Conference on Robotics and Automation (ICRA) 2021*, 2021.

[11] J. Lee, Y. Cho, C. Nam, J. Park, and C. Kim, "Efficient obstacle rearrangement for object manipulation tasks in cluttered environments," in *International Conference on Robotics and Automation, ICRA 2019, Montreal, QC, Canada, May 20-24, 2019*. IEEE, 2019, pp. 183–189.

[12] D. Kornhauser, G. L. Miller, and P. Spirakis, "Coordinating pebble motion on graphs, the diameter of permutation groups, and applications." Florida: IEEE, October 1984, pp. 241–250.

[13] K. Solovey and D. Halperin, "k-color multi-robot motion planning," in *Algorithmic Foundations of Robotics X*, E. Frazzoli, T. Lozano-Perez, N. Roy, and D. Rus, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 191–207.

[14] J. P. van den Berg, M. Stilman, J. Kuffner, M. C. Lin, and D. Manocha, "Path planning among movable obstacles: A probabilistically complete approach," in *Algorithmic Foundation of Robotics VIII, Selected Contributions of the Eight International Workshop on the Algorithmic Foundations of Robotics, WAFR 2008, Guanajuato, Mexico, December 7-9, 2008*, ser. Springer Tracts in Advanced Robotics, H. Choset, M. Morales, and T. D. Murphey, Eds., vol. 57. Springer, 2008, pp. 599–614.

[15] M. R. Dogar and S. S. Srinivasa, "A planning framework for non-prehensile manipulation under clutter and uncertainty," *Auton. Robots*, vol. 33, no. 3, pp. 217–236, 2012.

[16] J. E. King, M. Cognetti, and S. S. Srinivasa, "Rearrangement planning using object-centric and robot-centric action spaces," in *2016 IEEE International Conference on Robotics and Automation, ICRA*. IEEE, 2016.

[17] B. Huang, S. D. Han, J. Yu, and A. Boularias, "Visual foresight trees for object retrieval from clutter with nonprehensile rearrangement," *IEEE Robotics Autom. Lett.*, vol. 7, no. 1, pp. 231–238, 2022.

[18] E. Vieira, D. Nakhimovich, K. Gao, R. Wang, J. Yu, and K. E. Bekris, "Persistent homology for effective non-prehensile manipulation," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2022.

[19] M. Suhail Saleem and M. Likhachev, "Planning with selective physics-based simulation for manipulation among movable objects," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 6752–6758.

[20] I. A. Sucan and L. E. Kavraki, "A sampling-based tree planner for systems with complex dynamics," *IEEE Trans. Robotics*, 2012.

[21] S. M. LaValle and J. James J. Kuffner, "Randomized kinodynamic planning," *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[22] D. M. Saxena, M. S. Saleem, and M. Likhachev, "Manipulation planning among movable obstacles using physics-based adaptive motion primitives," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 6570–6576.

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

[24] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Conflict-based search for optimal multi-agent pathfinding," *Artif. Intell.*, vol. 219, pp. 40–66, 2015.

[25] S. Aine, S. Swaminathan, V. Narayanan, V. Hwang, and M. Likhachev, "Multi-heuristic A," *Int. J. Robotics Res.*, vol. 35, no. 1-3, pp. 224–243, 2016.

[26] I. A. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, December 2012, https://ompl.kavrakilab.org.

[27] B. Çalli, A. Singh, J. Bruce, A. Walsman, K. Konolige, S. S. Srinivasa, P. Abbeel, and A. M. Dollar, "Yale-cmu-berkeley dataset for robotic manipulation research," *Int. J. Robotics Res.*, vol. 36, no. 3, pp. 261–268, 2017.

[28] A. Agarwal, Y. Han, and M. Likhachev, "Perch 2.0 : Fast and accurate gpu-based perception via search for object pose estimation," in *IROS*, 2020.