Imagine All Objects Are Robots: A Multi-Agent Pathfinding Perspective on Manipulation Among Movable Objects

Dhruv Mauria Saxena[1], **Maxim Likhachev**[1]
[1]Robotics Institute, Carnegie Mellon University
5000 Forbes Ave
Pittsburgh, PA 15213, USA.
{dsaxena, mlikhach}@andrew.cmu.edu

Abstract

We consider the problem of planning for pick-and-place manipulation in heavy clutter where it might be necessary to interact with and rearrange movable objects via a sequence of non-prehensile pushes in order to grasp and extract a desired object. This planning problem is computationally very challenging for several reasons. First, it requires searching over a search-space that includes the configuration of movable objects. Second, it requires prediction of the effects of all the non-prehensile interactions with objects considered by the planner, which involves forward simulating a computationally expensive physics-based model. In this paper, we make an observation that the problem of planning for Manipulation Among Movable Objects is closely related to the Multi-Agent Pathfinding problem if we treat all movable objects as actuated robots. Using this insight, we construct a planning algorithm that iterates between (i) solving a multi-agent planning problem that reasons about the configuration of movable objects but does not forward simulate a physics model, and (ii) solving an arm motion planning problem that uses a physics-based simulator but does not search over the possible configurations of movable objects. We present the M4M algorithm, briefly analyse it from a theoretical perspective, and evaluate its performance experimentally in both simulations and on a physical PR2 robot.

Introduction

Manipulation Among Movable Objects (MAMO) (Stilman et al. 2007) 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 associates each object with constraints on how it can be interacted with – it is undesirable to allow robots to carelessly push or throw objects around. MAMO solutions rely on the ability of the robot to rearrange movable clutter such that the original manipulation task can be completed.

In this paper, we consider pick-and-place tasks where the robot needs to grasp and extract a desired object from a cluttered shelf. In order to retrieve the desired tomato soup can from the example MAMO problem in Fig. 1, the robot must first relocate the potted meat and coffee cans placed in front. Thus the robot is assigned a goal with respect to the overall task and object-of-interest (OoI), without any additional goal specifications for the movable objects except for satisfying their associated interaction constraints. 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 solutions must be found 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. This search also 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.

Figure 1: The tomato soup can (outlined in yellow) is the object-of-interest (OoI) to be retrieved. Access to the OoI is blocked by the potted meat can and coffee can in front of it. They must be rearranged out of the way in order to retrieve the OoI and solve the MAMO problem.

Our key insight in this paper exploits a connection between the MAMO domain and Multi-Agent Pathfinding (MAPF). We imagine the movable objects in the environ-

ment as actuated agents tasked with avoiding collisions with each other and our robot arm. A solution to a MAPF problem with all movable objects as 'agents' lets the robot arm successfully retrieve the OoI while the movable objects rearrange themselves. Although this solves the MAMO problem in the abstract space where movable objects are actuated, it prescribes a rearrangement strategy for the robot to 'realise' in the real-world where it is the only actuated entity. Our M4M algorithm iterates between two aspects of the MAMO problem – first we find a suitable rearrangement of the scene without any physics simulations, and then we try to realise this rearrangement via non-prehensile actions that are simulated for interaction constraint verification.

We use non-prehensile actions for rearrangement 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 to reach another one than to grasp it, pick it up, move it elsewhere, place it down, and release it before going back. In contrast to prehensile actions, non-prehensile actions do not require access to known grasp poses for object pickup or stable configurations for object placement. However, they do require simulating a physics model to obtain the result of robot-object interactions which can be time-consuming.

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

- 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.

Related Work

Manipulation Among Movable Objects

MAMO generalises Navigation Among Movable Obstacles (NAMO) where a mobile robot must navigate from start to goal in a reconfigurable environment (Alami, Laumond, and Simeon 1994; Wilfong 1991; Stilman and Kuffner 2005). It is also related to the rearrangement planning problem (Ben-Shahar and Rivlin 1998; Ota 2009) which explicitly specifies desired goal configurations for movable objects. Wilfong (Wilfong 1991) 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 (Stilman et al. 2007; Krontiris et al. 2014; Krontiris and Bekris 2015; Shome and Bekris 2021; Wang et al. 2021; Lee et al. 2019). 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 (Stilman et al. 2007; Krontiris et al. 2014; Krontiris and Bekris 2015; Shome and Bekris 2021). In some cases a "buffer" location to place grasped objects is required (Wang et al. 2021; Lee et al. 2019). In particular, (Krontiris et al. 2014) and (Shome and Bekris 2021) utilise the concept of "pebble graphs" (Kornhauser, Miller, and Spirakis 1984; Solovey and Halperin 2013) 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 (van den Berg et al. 2008; Dogar and Srinivasa 2012; King, Cognetti, and Srinivasa 2016; Huang et al. 2022; Vieira et al. 2022; Suhail Saleem and Likhachev 2020). Of these approaches, only Selective Simulation (Suhail Saleem and Likhachev 2020) considers realistic interactions in the 3D workspace and is one of our comparative baselines. Others rely on planar robot-object interactions which fail to account for object dynamics in $SE(3)$ where they might tilt, lean, or topple. For another baseline, we adapt the MAMO solver from (Dogar and Srinivasa 2012) 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 (van den Berg et al. 2008; Dogar and Srinivasa 2012; King, Cognetti, and Srinivasa 2016; Huang et al. 2022; Vieira et al. 2022)).

Querying *physics-based simulators* for the result of an action is much more expensive than collision checking it. KPIECE (Sucan and Kavraki 2012) 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 (LaValle and James J. Kuffner 2001) are two other baselines we compare against. In our own prior work on MAMO planning (Saxena, Saleem, and Likhachev 2021), 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, 6, and 7 for example).

Multi-Agent Pathfinding

Multi-agent pathfinding is a family of planning problems that tries to find paths for a team of robots from a set of start locations to a set of goal locations. One class of algorithms that solve MAPF problems assigns priorities to the robots and solves a sequence of single-agent planning problems based on this prioritisation (Erdmann and Lozano-Pérez 1987; Silver 2005). This *prioritised planning* scheme trades off algorithmic incompleteness and solution suboptimality

for practical efficiency. Turpin et al. (Turpin et al. 2014) present a resolution complete prioritised MAPF solver for the specific case when robots are interchangeable.

Conflict-based search (CBS) (Sharon et al. 2012a, 2015) and M* (Wagner and Choset 2011) are complete and optimal MAPF solvers that use different techniques to provide strong theoretical guarantees. CBS searches a tree of all possible solutions in a best-first manner by resolving robot conflicts into constraints on robot motion. For two robots that collide at any instant, either one can be in that location in the final solution but not both. CBS enumerates all such possibilities for all potential conflicts until a solution with no conflict is found. M* resolves robot conflicts by combining two conflicting robots and treating them as one agent until the conflict between them is resolved. This idea has also been adopted in the CBS family of algorithms (Sharon et al. 2012b).

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 $\mathcal{O} = \{O_1, \ldots, O_n\}$ 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 $\mathcal{X} = \mathcal{X}_\mathcal{R} \times \mathcal{X}_{O_1} \times \cdots \times \mathcal{X}_{O_n}$. 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 the movable objects and immovable obstacles. The goal for a MAMO planning problem is to find the least-cost valid path π from start to goal, i.e. a path made up of a sequence of valid states. A path $\pi = \{x_1, \ldots, x_T\}$ has cost $C(\pi) = \sum_{i=1}^{T-1} c(x_i, x_{i+1})$. Formally, we can write this as an optimisation problem:

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

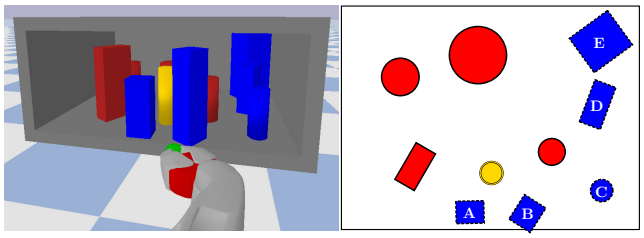

Figure 2: MAMO workspace (*left*) and its 2D projection labelled with movable object IDs. Movable objects are in blue, immovable obstacles in red, and the object-of-interest to be retrieved in yellow.

In our work we discretise the action space $\mathcal{A}$ for the robot to include *simple motion primitives* that independently change each joint angle by a fixed amount and dynamically generate *push actions* based on information about where to move an object. This implies that 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 the goal set $\mathcal{X}_G$ 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 $\mathcal{P} = (\mathcal{X}, \mathcal{A}, x_S, \mathcal{X}_G, \mathcal{T}, c)$ 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. 2 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 of the movable objects (objects A and B in particular), such a trajectory may be found. Our algorithm formulates an abstract MAPF problem to identify *which* objects need to move (A and B in Fig. 2), and *where* they should be moved (shown in Fig. 3).

Classical Multi-Agent Pathfinding

Classical MAPF planning problems seek to find non-conflicting paths for a set of agents $\{r_1, \ldots, r_n\}$ on a discrete graph $G = (V, E)$ in discrete time. Each robot r_i has a designated start state $s_i \in V$ and a desired goal state $g_i \in V$. Robot r_i has access to an action space $\mathcal{A}_i$, which includes an action to *wait* at the current state. An edge $(v, v') \in E$ implies that some action $a^j \in \mathcal{A}_i$ takes robot r_i from vertex v to v'. All actions are assumed to take the same amount of time such that traversing an edge $(v, v') \in E$ takes one unit of time. A *single-agent solution path* for r_i is a sequence of states $\pi_i = \{v_0 = s_i, \ldots, v_T = g_i\}$ where the subscripts denote time indices. Two single-agent solution paths π_i and π_j are *conflict-free* if robots r_i and r_j never collide as they traverse their respective paths. The solution to a MAPF problem with n robots $\{r_1, \ldots, r_n\}$ is a set of n *mutually* conflict-free paths, i.e. π_i and π_j must be conflict-free for

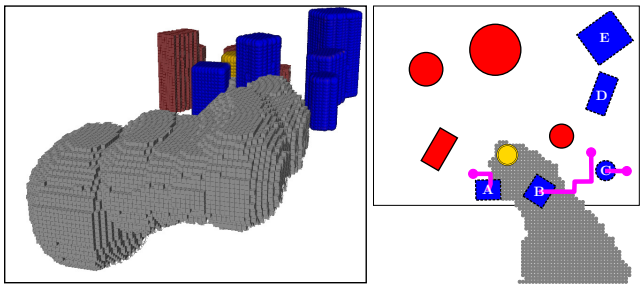

Figure 3: The negative goal region (NGR) $\mathcal{V}(\hat{\pi}_{\mathcal{R}})$ in gray for the MAMO problem from Fig. 2. (*left*) 3D volumes of the NGR and all objects at their initial poses (we omit the shelf for ease of visualisation). (*right*) 2D projection of the NGR and the workspace, overlayed with the solution to the abstract MAPF problem 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.

any $1 \leq i, j \leq n, i \neq j$.

Although a thorough review of MAPF literature is beyond the scope of this work, we would like to highlight that the MAPF problem is NP-hard to solve optimally (Yu and LaValle 2013). While MAPF typically focuses on finding paths on discrete graphs, multi-robot motion planning tries to compute dynamically feasible trajectories for robots in continuous space (Dayan et al. 2021; Shome et al. 2020). The agents in our MAPF formulation (movable objects) are artificially actuated so we are not concerned with any object dynamics. This allows us to solve the simpler discrete MAPF problem and the solution directly informs us of desirable final poses for the objects for the MAMO problem. We use Conflict-Based Search (CBS) (Sharon et al. 2015), a complete and optimal MAPF algorithm, to solve our abstract MAPF problem.

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 (Coumans and Bai 2016–2019)) 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) (Dogar and Srinivasa 2012) 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. 3 shows a NGR $\mathcal{V}(\hat{\pi}_{\mathcal{R}})$ for the problem from Fig. 2.

Algorithm 1 contains the pseudocode for M4M. At a high-level, M4M first computes $\hat{\pi}_{\mathcal{R}}$ (Line 3) and the NGR $\mathcal{V}(\hat{\pi}_{\mathcal{R}})$ (Line 4). It then iterates over two steps:

1. 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) (Sharon et al. 2015), a complete and optimal MAPF algorithm.
2. 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. 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* (Aine et al. 2016) 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.

We then loop over all objects that need to be rearranged (from Line 16) and try and find a valid push for them. If a valid push is found (Line 22), 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 8. Although this trajectory $\pi_{\mathcal{R}}$ may be different from $\hat{\pi}_{\mathcal{R}}$ (Line 3), 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.

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 existing MAPF solvers to compute a potential rearrangement of the scene which leads to successful OoI retrieval. Fig. 4 shows a series of images of movable objects executing the solution to the abstract MAPF problem we formulate in this section. In this abstract space where the movable objects are artificially actuated, they can rearrange themselves out of the way of the robot arm as it retrieves the target object.

Our MAPF abstraction includes all movable objects $o_m \in \mathcal{O}_M$ as agents. The high-level of CBS searches a tree of all

Algorithm 1: Multi-Agent Pathfinding for Manipulation Among Movable Objects

Input: Initial movable objects $\mathcal{O}_M^{\text{init}}$, Immovable obstacles $\mathcal{O}_I$
Output: Sequence of arm trajectories Ψ

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

possible solutions in a best-first manner by resolving agent conflicts into constraints on agent motion. For two agents that collide along their paths, either one can be in the location where and when they collide in the final solution but not both. We check for collisions between agents in space and time in their full $SE(3)$ configuration space. The low-level of CBS runs single-agent searches for each agent on a discrete graph $G_{\text{CBS}} = (V, E)$ where vertices $V \subset SE(3)$ are object poses. All agents have a discrete action space corresponding to a four-connected grid on the $(x, y)-$plane of the shelf on which they are placed. Edges $(v, v') \in E$ take unit time to traverse and either $v \equiv v'$, or the $x-$ or $y-$coordinate of the agent pose changes by $1\,\text{cm}$ between v and v'.

Agent goals in the MAPF abstraction are specified with respect to $\hat{\pi}_{\mathcal{R}}$ and $\mathcal{V}(\hat{\pi}_{\mathcal{R}})$ – the solution of the simpler MAMO problem $\hat{\mathcal{P}}$ that does not include any movable objects and its corresponding "negative goal region" (NGR). Let $\mathcal{B}(o_m, x)$ denote the volume occupied by movable object $o_m \in \mathcal{O}_M$ at pose $x \in SE(3)$. o_m satisfies the NGR if it is "outside" it, i.e. in a state x such that $\mathcal{V}(\hat{\pi}_{\mathcal{R}}) \cap \mathcal{B}(o_m, x) = \emptyset$. To be precise, each agent o_m in the MAPF problem has a *set* of possible goals $\{g : g \in V \wedge \mathcal{V}(\hat{\pi}_{\mathcal{R}}) \cap \mathcal{B}(o_m, g) = \emptyset\}$.

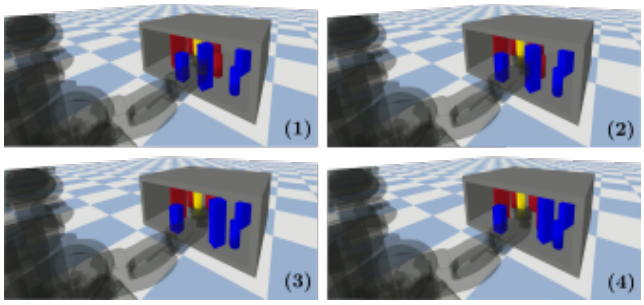

Figure 4: A sequence of images $(1) - (4)$ showing artificially actuated movable objects executing the solution to the abstract MAPF problem.

A solution to this MAPF abstraction, shown in Fig. 3, prescribes a rearrangement strategy in terms of *which* objects to move and *where*. 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. 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.

Generating Non-Prehensile Push Actions

Given a path π_{o_m} for $o_m \in \mathcal{O}_M$ from the MAPF solution, PLANPUSH (Algorithm 2 and Algorithm 1, Line 20) 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 $\{x^1 = \pi_{o_m}^{start}, \ldots, x^n = \pi_{o_m}^{end}\}$ and 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 (Algorithm 2, Lines 1- 3). PLANPUSH then 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}. 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. If this path is found (Line 7), 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 $\left(x_{\text{push}}^{i-1}, x_{\text{push}}^i\right), i = \{1, \ldots, n\}$ (loop from Line 9). 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 22).

Theoretical Discussion

The solution returned by M4M lies on a discrete graph $G_0 = (V_0, E_0)$ whose vertices V_0 are MAMO states in the

Algorithm 2: Non-Prehensile Push Planner PLANPUSH

Input: Object to be pushed o_m, MAPF solution path π_{o_m} for object o_m, Current movable object poses $\mathcal{O}_M$, Immovable obstacles $\mathcal{O}_I$

Output: Push trajectory $\psi \in \mathcal{X}_\mathcal{R}$ for object o_m

1: $\{x^1, \ldots, x^n\} \leftarrow$ SHORTCUTPATH(π_{o_m})
2: $aabb \leftarrow$ COMPUTEAABB(o_m)
3: $x^{aabb} \leftarrow$ RAYAABBINTERSECTION $\left(aabb, \overrightarrow{(x^2, x^1)}\right)$
4: $x^0_{\text{push}} \sim \mathcal{N}(x^{aabb}, \sigma)$
5: ADDOBSTACLES$(\mathcal{O}_I \cup \mathcal{O}_M)$
6: $\pi_0 \leftarrow$ PLANAPPROACH(x^0_{push})
7: **if** π_0 exists **then**
8: REMOVEOBSTACLES$(\mathcal{O}_M)$
9: **for** $i = \{1, \ldots, n\}$ **do**
10: $\pi_i \leftarrow$ INVERSEKINEMATICS $\left(\pi_0^{end}, \overrightarrow{(x^{i-1}_{\text{push}}, x^i_{\text{push}})}\right)$
11: **if** π_i exists **then**
12: $\pi_0 \leftarrow \pi_0 \cup \pi_i$
13: **else**
14: **return** $\emptyset$
15: **end if**
16: **end for**
17: **end if**
18: **return** π_0

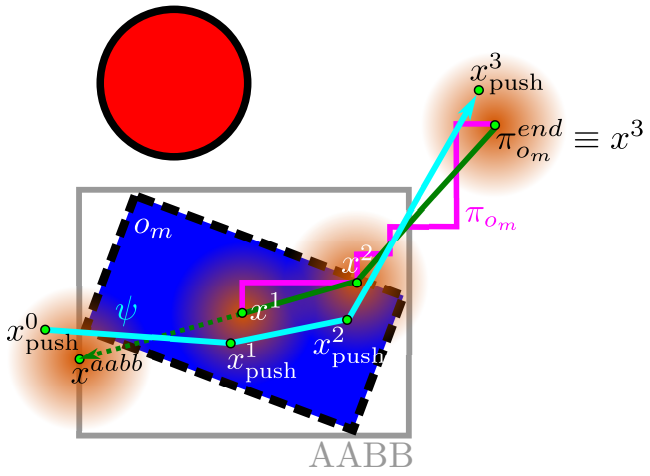

Figure 5: 2D illustration of our push planner. Given a movable object o_m (blue) and its MAPF solution path π_{o_m} (pink), we shortcut π_{o_m} while accounting for immovable obstacles $\mathcal{O}_I$ (red) to get the green path of straight line segments. After computing x^{aabb} by intersecting the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

composite configuration space $\mathcal{X} = \mathcal{X}_\mathcal{R} \times \mathcal{X}_{O_1} \times \cdots \times \mathcal{X}_{O_n}$ and edges E_0 correspond to actions taken by the robot arm. The iterative nature of M4M greedily commits to valid push actions found during the search for a MAMO solution. This notion of planning "in the now" (Kaelbling and Lozano-Pérez 2011) drastically reduces search efforts since we do not consider (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. The trade-off associated with the computational savings of being greedy makes M4M *incomplete* with respect to G_0.

Two modifications to M4M that introduce notions of *backtracking* and *feedback* would result in a resolution complete version with respect to G_0. First, for a particular rearrangement suggested by the MAPF solver, we must consider all feasible pushes for an object and all orderings of these pushes for the set of objects that need to be rearranged. This can be formulated as a search over a discrete graph where nodes are MAMO states and edges represent abstract actions to move an object, changing the greedy nature of M4M to a systematic search inspired by the Minimum Constraint Removal problem (Hauser 2014). Searching this graph is computationally expensive since each abstract action can correspond to multiple push actions, however this graph contains all possible orderings of all feasible pushes to realise a specific rearrangement. If this search fails to find a solution, we must query the MAPF solver for a different solution since we failed to realise the previous one. This feedback loop would force the algorithm to systematically search over all possible rearrangements of the scene since the MAPF solver, itself being complete with respect to G_{CBS}, will find any possible rearrangement. Developing an *efficient* version of such a resolution complete algorithm is ongoing work.

Experimental Results

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 4) 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 (Sucan and Kavraki 2012) and RRT (LaValle and James J. Kuffner 2001) from OMPL (Şucan, Moll, and Kavraki 2012) that search the entire MAMO state space $\mathcal{X}$ by randomly sampling robot motions.

The second baseline, Selective Simulation (Suhail Saleem and Likhachev 2020) (SELSIM), is a search-based algorithm that interleaves a 'planning' phase and a 'tracking' phase.

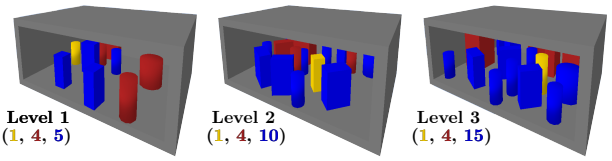

Figure 6: MAMO problems of differing complexity. From *left* to *right*, Levels 1, 2, and 3 have 5, 10, and 15 movable objects respectively. Each Level has 1 OoI and 4 immovable obstacles.

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 simple motion primitives that independently change robot arm joint angles.

Our final baseline is the work from Dogar et al. (Dogar and Srinivasa 2012) (DOGAR) which introduced the idea of a negative goal region (NGR) we use in M4M. DOGAR recursively searches for a solution backwards in time, similar to (Stilman et al. 2007). 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 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 1 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 1 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 8). 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 which simulates all trajectories found during planning. In fact, all SELSIM successes in Table 1 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).

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 (Çalli et al. 2017) 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 (Agarwal, Han, and Likhachev 2020) 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.

Comparison of MAPF Solvers

M4M uses a complete MAPF solver (CBS) to ensure it does not miss any potential rearrangement of a scene (with respect to the graph G_{CBS} of the low-level CBS searches). Another class of MAPF solvers assigns priorities to agents and solves a sequence of single-agent planning problems based on this prioritisation (Erdmann and Lozano-Pérez 1987). This *prioritised planning* (PP) scheme trades off algorithmic incompleteness and solution suboptimality for practical

Table 1: Simulation Study for MAMO Planning in Cluttered Scenes - success rates and *min/median/max* times

Metrics	Level	Planning Algorithms					
		M4M	$\widehat{\text{M4M}}$	DOGAR	SELSIM	KPIECE	RRT
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

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

efficiency. For all problems solved by M4M in Table 1, we compare the performance of CBS against PP in terms of success rates for an initial solution and planning times.

CBS succeeds in finding an MAPF solution 100% of the time. Given the same 30 s timeout as CBS, PP failed to find solutions for 2 Level 1 problems, 9 Level 2 problems, and 9 Level 3 problems. The bottleneck for MAPF is collision checking between objects in $SE(3)$. CBS only collision checks solution trajectories returned by the low-level searches of the corresponding agents. In addition to being incomplete, PP turns out to also be much slower than CBS. This is because PP collision checks every state expanded by the low-level search against corresponding states of higher priority agents, which is slow when many agents collide with each other along their solution paths. We compare the ratio of planning times for CBS to those for PP across the three levels. The median value of this ratio $T_{\text{CBS}}/T_{\text{PP}}$ for Level 1 is 1.22 (PP is at least 22% faster in half the problems). For Levels 2 and 3, this value is 0.89 (PP is at least 11% slower in half the problems) and 0.63 (PP is at least 37% slower in half the problems). The ability to solve all problems and in less time when the MAMO problem is more complicated highlights the benefit of using CBS over PP.

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. These MAMO problems include interaction constraints that define how the robot is allowed to interact with objects. M4M decouples the search over all rearrangements of movable objects from the need to query a physics-based simulator. 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. The MAPF formulation searches over object configurations without a simulator, and upon returning a solution, M4M computes non-prehensile push actions to realise the suggested rearrangement within the simulator. 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. Using a model-based push planner, even for simple straight-line pushes like those used by M4M, will greatly reduce the time M4M currently spends stochastically sampling and simulating valid pushes.

References

Agarwal, A.; Han, Y.; and Likhachev, M. 2020. PERCH 2.0 : Fast and Accurate GPU-based Perception via Search for Object Pose Estimation. In *IROS*.

Aine, S.; Swaminathan, S.; Narayanan, V.; Hwang, V.; and Likhachev, M. 2016. Multi-Heuristic A. *Int. J. Robotics Res.*, 35(1-3): 224–243.

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

Ben-Shahar, O.; and Rivlin, E. 1998. Practical pushing planning for rearrangement tasks. *IEEE Trans. Robotics Autom.*, 14(4): 549–565.

Çalli, B.; Singh, A.; Bruce, J.; Walsman, A.; Konolige, K.; Srinivasa, S. S.; Abbeel, P.; and Dollar, A. M. 2017. Yale-CMU-Berkeley dataset for robotic manipulation research. *Int. J. Robotics Res.*, 36(3): 261–268.

Coumans, E.; and Bai, Y. 2016–2019. PyBullet, a Python module for physics simulation for games, robotics and machine learning. http://pybullet.org.

Dayan, D.; Solovey, K.; Pavone, M.; and Halperin, D. 2021. Near-Optimal Multi-Robot Motion Planning with Finite Sampling. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 9190–9196.

Dogar, M. R.; and Srinivasa, S. S. 2012. A Planning Framework for Non-Prehensile Manipulation under Clutter and Uncertainty. *Auton. Robots*, 33(3): 217–236.

Erdmann, M. A.; and Lozano-Pérez, T. 1987. On Multiple Moving Objects. *Algorithmica*, 2: 477–521.

Hauser, K. K. 2014. The minimum constraint removal problem with three robotics applications. *Int. J. Robotics Res.*, 33(1): 5–17.

Huang, B.; Han, S. D.; Yu, J.; and Boularias, A. 2022. Visual Foresight Trees for Object Retrieval From Clutter With Nonprehensile Rearrangement. *IEEE Robotics Autom. Lett.*, 7(1): 231–238.

Kaelbling, L. P.; and Lozano-Pérez, T. 2011. Hierarchical task and motion planning in the now. In *IEEE International Conference on Robotics and Automation, ICRA 2011, Shanghai, China, 9-13 May 2011*, 1470–1477. IEEE.

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

Kornhauser, D.; Miller, G. L.; and Spirakis, P. 1984. Coordinating Pebble Motion on Graphs, The Diameter of Permutation Groups, and Applications. 241–250. Florida: IEEE.

Krontiris, A.; and Bekris, K. E. 2015. Dealing with Difficult Instances of Object Rearrangement. In Kavraki, L. E.; Hsu, D.; and Buchli, J., eds., *Robotics: Science and Systems XI, Sapienza University of Rome, Rome, Italy, July 13-17, 2015*.

Krontiris, A.; Shome, R.; Dobson, A.; Kimmel, A.; and Bekris, K. E. 2014. 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*, 1081–1087. IEEE.

LaValle, S. M.; and James J. Kuffner, J. 2001. Randomized Kinodynamic Planning. *The International Journal of Robotics Research*, 20(5): 378–400.

Lee, J.; Cho, Y.; Nam, C.; Park, J.; and Kim, C. 2019. 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*, 183–189. IEEE.

Ota, J. 2009. Rearrangement Planning of Multiple Movable Objects by a Mobile Robot. *Adv. Robotics*, 23(1-2): 1–18.

Saxena, D. M.; Saleem, M. S.; and Likhachev, M. 2021. Manipulation Planning Among Movable Obstacles Using Physics-Based Adaptive Motion Primitives. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 6570–6576.

Sharon, G.; Stern, R.; Felner, A.; and Sturtevant, N. R. 2012a. Conflict-Based Search for Optimal Multi-Agent Path Finding. In Borrajo, D.; Felner, A.; Korf, R. E.; Likhachev, M.; López, C. L.; Ruml, W.; and Sturtevant, N. R., eds., *Proceedings of the Fifth Annual Symposium on Combinatorial Search, SOCS 2012, Niagara Falls, Ontario, Canada, July 19-21, 2012*. AAAI Press.

Sharon, G.; Stern, R.; Felner, A.; and Sturtevant, N. R. 2012b. Meta-Agent Conflict-Based Search For Optimal Multi-Agent Path Finding. In Borrajo, D.; Felner, A.; Korf, R. E.; Likhachev, M.; López, C. L.; Ruml, W.; and Sturtevant, N. R., eds., *Proceedings of the Fifth Annual Symposium on Combinatorial Search, SOCS 2012, Niagara Falls, Ontario, Canada, July 19-21, 2012*. AAAI Press.

Sharon, G.; Stern, R.; Felner, A.; and Sturtevant, N. R. 2015. Conflict-based search for optimal multi-agent pathfinding. *Artif. Intell.*, 219: 40–66.

Shome, R.; and Bekris, K. E. 2021. Synchronized Multi-arm Rearrangement Guided by Mode Graphs with Capacity Constraints. In LaValle, S. M.; Lin, M.; Ojala, T.; Shell, D. A.; and Yu, J., eds., *Algorithmic Foundations of Robotics XIV, Proceedings of the Fourteenth Workshop on the Algorithmic Foundations of Robotics, WAFR 2021, Oulu, Finland, June 21-23, 2021*, volume 17 of *Springer Proceedings in Advanced Robotics*, 243–260. Springer.

Shome, R.; Solovey, K.; Dobson, A.; Halperin, D.; and Bekris, K. E. 2020. drrt*: Scalable and informed asymptotically-optimal multi-robot motion planning. *Autonomous Robots*, 44(3): 443–467.

Silver, D. 2005. Cooperative Pathfinding. In Young, R. M.; and Laird, J. E., eds., *Proceedings of the First Artificial Intelligence and Interactive Digital Entertainment Conference, June 1-5, 2005, Marina del Rey, California, USA*, 117–122. AAAI Press.

Solovey, K.; and Halperin, D. 2013. k-Color Multi-robot Motion Planning. In Frazzoli, E.; Lozano-Perez, T.; Roy,

N.; and Rus, D., eds., *Algorithmic Foundations of Robotics X*, 191–207. Berlin, Heidelberg: Springer Berlin Heidelberg.

Stilman, M.; and Kuffner, J. J. 2005. Navigation among Movable Obstacles: Real-Time Reasoning in Complex Environments. *Int. J. Humanoid Robotics*, 2(4): 479–503.

Stilman, M.; Schamburek, J.; Kuffner, J.; and Asfour, T. 2007. Manipulation Planning Among Movable Obstacles. In *2007 IEEE International Conference on Robotics and Automation, ICRA*. IEEE.

Sucan, I. A.; and Kavraki, L. E. 2012. A Sampling-Based Tree Planner for Systems With Complex Dynamics. *IEEE Trans. Robotics*.

Şucan, I. A.; Moll, M.; and Kavraki, L. E. 2012. The Open Motion Planning Library. *IEEE Robotics & Automation Magazine*, 19(4): 72–82. https://ompl.kavrakilab.org.

Suhail Saleem, M.; and Likhachev, M. 2020. Planning with Selective Physics-based Simulation for Manipulation Among Movable Objects. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 6752–6758.

Turpin, M.; Mohta, K.; Michael, N.; and Kumar, V. 2014. Goal assignment and trajectory planning for large teams of interchangeable robots. *Auton. Robots*, 37(4): 401–415.

van den Berg, J. P.; Stilman, M.; Kuffner, J.; Lin, M. C.; and Manocha, D. 2008. Path Planning among Movable Obstacles: A Probabilistically Complete Approach. In Choset, H.; Morales, M.; and Murphey, T. D., eds., *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*, volume 57 of *Springer Tracts in Advanced Robotics*, 599–614. Springer.

Vieira, E.; Nakhimovich, D.; Gao, K.; Wang, R.; Yu, J.; and Bekris, K. E. 2022. Persistent Homology for Effective Non-Prehensile Manipulation. In *IEEE International Conference on Robotics and Automation (ICRA)*.

Wagner, G.; and Choset, H. 2011. M*: A complete multirobot path planning algorithm with performance bounds. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2011, San Francisco, CA, USA, September 25-30, 2011*, 3260–3267. IEEE.

Wang, R.; Gao, K.; Nakhimovich, D.; Yu, J.; and Bekris, K. E. 2021. Uniform Object Rearrangement: From Complete Monotone Primitives to Efficient Non-Monotone Informed Search. In *International Conference on Robotics and Automation (ICRA) 2021*.

Wilfong, G. T. 1991. Motion Planning in the Presence of Movable Obstacles. *Ann. Math. Artif. Intell.*

Yu, J.; and LaValle, S. M. 2013. Structure and Intractability of Optimal Multi-Robot Path Planning on Graphs. In desJardins, M.; and Littman, M. L., eds., *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence, July 14-18, 2013, Bellevue, Washington, USA*. AAAI Press.