2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

Improved M4M: Faster and Richer Planning for Manipulation Among Movable Objects in Cluttered 3D Workspaces

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

***Abstract*— We are interested in enabling robots to solve difficult pick-and-place manipulation tasks in cluttered and constrained environments. If the robot does not have collision-free access to the object-of-interest (OoI) which it intends to grasp and extract from the workspace, it must reason about *which* movable objects to rearrange, *where* to move them, and *how* it may do so. In recent work [1] we introduced E−M4M, a graph search-based solver for solving such Manipulation tasks Among Movable Objects (MAMO). In this paper we make several improvements to E−M4M – we introduce the use of prehensile or pick-and-place rearrangement actions in addition to pushes; we show that by running it as a depth-first search improves performance; we show how the search can be run "eagerly lazily" to only simulate actions in a physics-based simulator when necessary; finally we relax the assumption that we require perfect knowledge of the physical properties of objects (mass and coefficient of friction in particular). The improved version of E−M4M presented in this paper, I−M4M, is a faster and more versatile MAMO solver with a rich action space. We discuss the impact of the improvements we make in an extensive simulation study and show previously unachievable results on a real-world PR2 robot.**

I. Introduction

In cluttered real-world workspaces, simple pick-and-place tasks for robot manipulators can be quite challenging to solve. Often there is no collision-free trajectory that allows the robot to grasp and extract the desired object from the scene. This requires motion planning algorithms to reason about rearranging some of the *movable* clutter in the scene so as to make the task feasible. Our work focuses on solving these pick-and-place tasks in 3D workspaces where objects may tilt, lean on each other, topple, and slide. "Manipulation Among Movable Objects" (MAMO) [2], [3] defines a class of *hybrid* or *multi-modal* planning problems [4], [5]. The difficulty in solving multi-modal planning problems arises from the need to jointly optimise over discrete decisions (which objects to rearrange) and continuous trajectories (for the robot arm to rearrange objects) in a search space that grows with the number of movable objects in the workspace of the robot. In our work, the configuration space of the robot $\mathcal{X}_{\mathcal{R}} \subset \mathbb{R}^7$ since we use the 7 degree-of-freedom manipulator on a PR2 robot. The configuration space of each movable object O_i is $\mathcal{X}_{O_i} \equiv SE(3)$ since we keep track of their 3D pose (position and orientation). The search space for MAMO solutions in a workspace with n movable objects is the cross-product space $\mathcal{X} = \mathcal{X}_{\mathcal{R}} \times \mathcal{X}_{O_1} \times \cdots \times \mathcal{X}_{O_n}$. As an example, Fig. 1 shows a MAMO problem a robot may be asked to solve. The task can be stated very simply – retrieve the can of beer (object-of-interest or OoI, outlined in yellow) from the refrigerator shelf. The complexity in solving this problem arises from the fact that the robot is only allowed to move the yogurt and almond beverage (movable objects outlined in blue). Furthermore, the solution must not violate object-centric "interaction constraints" specified in the problem. These may encode properties desired in the MAMO solution such as whether the robot is allowed to make contact with certain objects, how far it can tilt them, whether they can topple, and how fast they can be moved at all times along the solution trajectory. In the problems we consider, the robot cannot make movable objects tilt beyond $25°$ along any axis, topple, or make contact with the (fragile) immovable obstacles outlined in red in Fig. 1 (eggs, cup of coffee, and glass bottles).

Fig. 1. An example MAMO problem to retrieve the beer can (object-of-interest or OoI, yellow outline). The yogurt and almond beverage are movable objects (blue outlines). All other objects in the scene are immovable obstacles (red outline).

In prior work [1], [6] we have developed the E−M4M algorithm – a solver for MAMO problems that draws a connection between the MAMO domain and Multi-Agent Pathfinding (MAPF). E−M4M first solves an MAPF abstraction of the MAMO problem to determine which rearrangement actions it should generate as candidates from a particular state. These are then converted into robot trajectories for validation in a physics-based simulator. E−M4M formulates a graph search for MAMO problems and searches over different rearrangements of the scene, different orderings of rearrangements for movable objects in a scene, and different ways to rearrange a movable object. However E−M4M is limited to the use of nonprehensile or pushing actions to rearrange movable objects, uses a learned heuristic

[1]The authors are with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA. e-mail: {dsaxena, mlikhach}@andrew.cmu.edu. This work was in part supported by ARL grant W911NF-18-2-0218 and ONR grant N00014-18-1-2775.

function that does not capture the complexity of difficult-to-solve problem instances[1], and assumes accurate estimates of movable object masses and coefficients of friction in order to minimise the *sim-to-real* gap [7], [8].

In this paper we make several changes to E−M4M and present an **I**mproved version of the algorithm, I−M4M. Each of the modifications are intended to create a more versatile and/or efficient solver for MAMO problems. To be precise, we make the following contributions with the I−M4M algorithm:

1) We introduce the use of prehensile rearrangement actions to allow the robot to deliberately pick up objects and place them down elsewhere. This makes significant progress towards the MAMO solution that may otherwise have required multiple pushes.
2) We implement I−M4M as a depth-first search with respect to valid rearrangement actions found. In doing so we are able to address inaccuracies in the learned heuristic function used by E−M4M that would lead to re-expansions of states from which the algorithm had already made progress.
3) With the inclusion of prehensile rearrangements I−M4M uses a richer action space that increases the branching factor of the search and hence the computational complexity. We propose an "eagerly lazy" version of I−M4M that only validates rearrangement actions when necessary.
4) We parallelise simulations across multiple instances of the physics-based simulator to relax the assumption of accurate estimates of object masses and coefficients of friction that make I−M4M more robust to modeling errors in the real-world.

Section II of this paper contains a review of work related to MAMO. Section III provides the necessary background about the E−M4M algorithm. Section IV contains technical details of the major improvements we incorporate in the I−M4M algorithm. These are thoroughly evaluated in simulation and on real-world runs on the PR2. The results of these experiments are presented in Section V. Section VI provides a discussion about the work in this paper and how it can be extended in future work.

II. Related Work

There are two broad categories of MAMO solvers that can be differentiated by whether they use a discretised action space or whether they search for solutions in the joint configuration space of the robot and all movable objects.

Task and motion planning (TAMP) solvers [9]–[14] use a discrete set of parameterised symbolic actions to compute "task plans" – abstract, high-level action sequences that solve the MAMO problem but need to be refined into continuous trajectories executable by a robot. They differ in whether a complete task plan to the goal state is refined or whether each abstract action is verified by a motion planner when generated. In our work, we present an algorithm that implicitly generates these high-level actions by solving an appropriately constructed MAPF abstraction to MAMO and does not rely on a predefined set of symbolic actions. Other MAMO solvers and algorithms for *rearrangement planning* keep track of object configurations as part of the search graph they construct with a discrete set of rearrangement actions. To avoid an exhaustive search over the entire configuration space $\mathcal{X}$, these algorithms restrict themselves to the use of prehensile or pick-and-place style rearrangement actions that change the configuration of at most one object at a time [3], [15]–[22]. Pick-and-place rearrangement actions can be computed with collision-free motion planning where the robot grasps an object, rigidly attaching the object to its kinematic chain, and relocates it to the desired location. This does not require querying a computationally expensive model (such as a physics-based simulator in our case) to forward simulate the effect of robot actions on the configurations of all other movable objects, something typical of algorithms that use nonprehensile robot-object interactions.

Since nonprehensile robot-object interactions can change the configuration of several movable objects during the same action, usually a physics-based simulator is integrated into the motion planning algorithm. The simulator is used to forward simulate the effect of robot actions on the configuration of objects in the environment. Since querying such a simulator is computationally expensive, existing literature limits nonprehensile interactions to a planar world [23]–[28]. This makes it easier to find valid pushing actions as the only interaction constraint to be satisfied is ensuring objects remain within workspace bounds. Our work does not make the planar world assumption and keeps track of object configurations in $SE(3)$ (3D position and orientation) and satisfies a more challenging but realistic set of interaction constraints.

There is existing work that allows the robot to manipulate objects in the workspace via both prehensile and nonprehensile rearrangement actions [29]–[31]. However they also limit nonprehensile interactions to a planar world and only allow the robot to manipulate a single object at a time. We do not impose these restrictions in our work, nor do we limit each object to be rearranged once.

The second category of algorithms that search the joint configuration space of the robot and all objects are able to rely on sampling algorithms to generate candidate robot actions [23], [30], [32]–[37]. They are aided by additional guidance or biased sampling that provides heuristic guidance to the algorithm about which (if any) object should be manipulated and how. Additionally they may rely on constrained projections of robot motions to ensure that the robot maintains appropriate contacts when evaluating actions.

III. Background

A. Multi-Agent Pathfinding for MAMO

In order to determine *which* movable objects should be rearranged and *where* they should be moved to solve

[1]Problems that are difficult to solve typically require the robot to rearrange many objects, each with multiple actions.

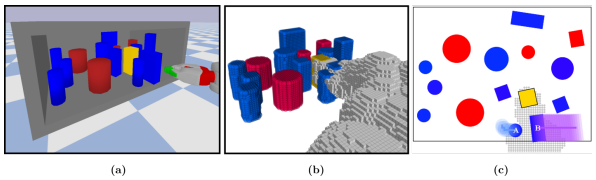

Fig. 2. (a) A MAMO problem with ten movable objects (blue), four immovable obstacles (red), and one OoI (yellow); (b) the initial NGR (in gray), and (c) a 2D projection of the scene with the MAPF solution paths in pink. This MAPF solution suggests that the objects labelled A and B should be rearranged as per the pink paths.

the MAMO problem, [6] proposed a MAPF abstraction to MAMO where the movable objects are artificially actuated agents. They are given the task of moving to configurations such that the robot can retrieve the OoI without making contact with any other objects. The MAPF problem specifies this goal condition for the movable objects as a "negative goal region" (NGR) [29] – a volume of the workspace that when free of movable objects, will allow contact-free OoI retrieval. Fig. 2 (a) shows a simulated MAMO problem with a valid NGR for this problem shown alongside in Fig. 2 (b). A NGR can be easily computed for a MAMO problem by finding a robot arm trajectory in $\mathcal{X}_\mathcal{R}$ to retrieve the OoI in the absence of all movable objects. The NGR is the volume swept by the robot arm along this trajectory. Note that if such a trajectory does not exist the MAMO problem is unsolvable.

Once an NGR has been computed, we solve an MAPF problem with all movable objects assumed to be actuated agents, immovable objects (and the OoI) as obstacles, and a goal condition for all agents specified by the NGR[2]. We use Conflict-Based Search (CBS) [38] as our MAPF solver. For ease of visualisation, Fig. 2 (c) shows a 2D projection of the solution found by CBS for the MAMO problem discussed earlier. It suggests that the robot should rearrange object A to the left side and object B to the right side of the NGR.

B. E−M4M

E−M4M [1] is a discrete graph search algorithm for solving MAMO problems that searches a graph $G = (V, E)$ of vertices $v \in V \subset \mathcal{X}$ where $\mathcal{X} = \mathcal{X}_\mathcal{R} \times \mathcal{X}_{O_1} \times \cdots \times \mathcal{X}_{O_n}$ is the search space for MAMO problems. Each vertex contains a different rearrangement of the scene. Edges or transitions $e = (v, v') \in E$ between states $v, v' \in V$ generated by E−M4M are rearrangement actions $a \in \mathcal{A}$ in the action space of the robot that change the configuration of at least one movable object. A *valid* edge (v, v') between two states implies that we have found a robot arm trajectory that rearranges the objects from configuration v to v' while not violating any interaction constraints.

To determine the set of outgoing edges from a vertex v in G, E−M4M first solves the MAPF problem described in Section III-A. Each object that moves in the MAPF solution for v leads to an outgoing edge to some v' that will be determined once the edge is evaluated. Evaluating an edge requires (i) calling a motion planner to compute a robot arm trajectory in $\mathcal{X}_\mathcal{R}$ and (ii) forward simulating it in a physics-based simulator for interaction constraint verification. Fig. 3 shows the graph constructed by E−M4M to solve the MAMO problem from Fig. 2. We only show the result of valid rearrangement actions found by E−M4M.

Fig. 3 also contains a graphical illustration of the non-prehensile push planner used by E−M4M to convert MAPF paths into pushing trajectories. First, E−M4M computes an appropriate end-effector pose in $x^{aabb} \in SE(3)$ on the basis of the path π_m found in the MAPF solution and plans a trajectory to this pose in $\mathcal{X}_\mathcal{R}$ (while avoiding collisions with *all* objects). If this trajectory is found, it uses inverse kinematics (IK) (along the MAPF path) to compute the pushing action γ_m. If IK succeeds[3], the pushing action is forward simulated in the physics-based simulator for constraint verification.

IV. Improvements to the E−M4M Algorithm

This paper introduces several improvements to the E−M4M that contribute to a faster MAMO solver with a richer action space. We call this improved version of the algorithm Improved-M4M or I−M4M. This section discusses four improvements we introduce in this paper.

A. Addition of Prehensile Rearrangement Actions

Given the structure of the E−M4M graph search, it is straightforward to extend it to obtain a richer MAMO solver with a diverse action space.For objects that are "graspable", i.e. those that will fit inside the end-effector of our robot, we generate two outgoing edges from vertex v to vertices v'_1 and v'_2 in I−M4M corresponding to rearranging an object by a push and pick-and-place action respectively. To validate edges (v, v'_2) corresponding to pick-and-place rearrangement, we first compute a pick-and-place trajectory, and if one is found, validate it in the simulator.

We assume access to known grasp poses for each object. While the MAPF solution contains entire paths for objects to be rearranged as shown in Fig. 2, if we want to rearrange an object with a pick-and-place action we select the final state in its MAPF solution path as the placement pose and discard the rest. The rearrangement action can be broken down into four parts – a trajectory to reach the grasp pose, the grasp maneuver, a trajectory to reach the placement pose, and the placement action. Grasping and placement actions are hardcoded movements of the end-effector relative to the object from the grasp pose and placement pose repectively. The trajectories to the grasp and placement poses are computed in $\mathcal{X}_\mathcal{R}$ by calling a motion planner for the robot arm. For prehensile rearrangements, we impose the restriction that the entire trajectory must be collision-free with *all* objects (movable and immovable). If such a collision-free trajectory is found, we simulate the grasping and placement actions in the simulator to ensure that no interaction constraints are violated. We do not need to simulate any other parts of the trajectory since by construction they are collision-free.

[2]We also enforce the constraint that agents should remain within workspace bounds, i.e. on the shelf, as part of the solution.

[3]IK does not hit joint limits or collides with immovable obstacles.

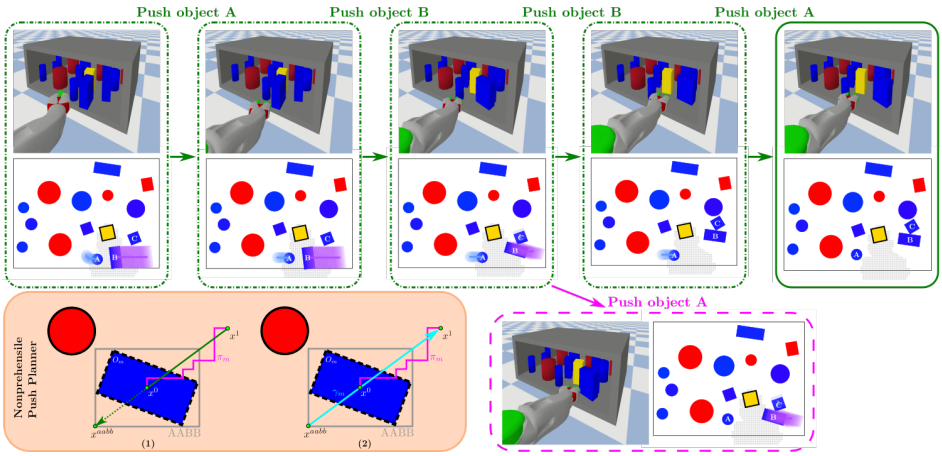

Fig. 3. The graph of valid rearrangement actions constructed by E−M4M while solving the MAMO problem from Fig. 2. Green edges and vertices are part of the solution path, pink ones are not. An inset shows an illustration of the nonprehensile push planner used by E−M4M.

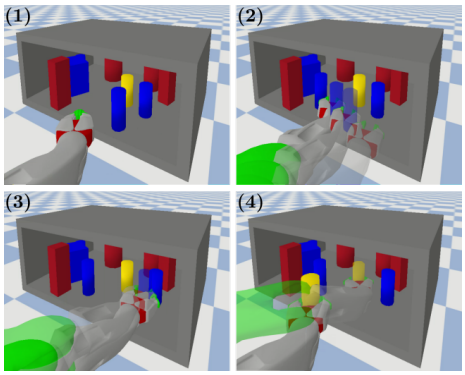

Fig. 4. (1) The initial scene; (2) Pushing the movable cylinder in the front to the left; (3) Prehensile rearrangement of the movable cylinder in the back; (4) Retrieving the OoI from the scene.

Fig. 4 shows a MAMO solution found by I−M4M with one nonprehensile and one prehensile rearrangement.

B. Depth-First I−M4M

E−M4M was proposed as a prioritised best-first search on graph G described in Section III-B where the priority function was learned offline to predict the likelihood a particular state lay on a path to the goal state for the MAMO problem. This was shown to be particularly ineffective in disambiguating between, i.e. accurately prioritising, scenes in which several objects overlap with the negative goal region. This leads to E−M4M (re-)expanding vertices in the graph from which it is difficult to make further progress.

Instead I−M4M is run as a depth-first search to continue the search for solutions from states which we have reached via valid rearrangement actions. This is closely related to the work of Garrett et al. [31] which proposes a hill-climbing solver for TAMP problems, and that of Stilman et al. [3] and Dogar et al. [29] who propose a depth-first solver for MAMO problems that searches backwards from the goal state.

I−M4M orders the priority queue lexicographically based on the pair $(\texttt{actions}, \texttt{reexpands})$ where $\texttt{actions}$ is the number of valid rearrangement actions found on the partial path to a vertex and $\texttt{reexpands}$ is the number of times that vertex has been re-expanded. We preferentially prioritise greater values of $\texttt{actions}$ and lesser values of $\texttt{reexpands}$. Thus if $u \prec v$ implies that u appears in the priority queue *ahead* of v, and consequently u will be expanded *before* v, I−M4M will contain the following ordering of states: $(3,0) \prec (3,1) \prec (2,4) \prec (1,0) \prec (0,2)$.

C. Eagerly Lazy Evaluation of Rearrangement Actions

The virtue of lazy search algorithms is well-understood [39]–[41]. *Laziness* in the property where the evaluation of an edge is postponed until it becomes necessary. For MAMO problems edge-evaluations require computing a robot arm trajectory in $\mathcal{X}_{\mathcal{R}}$ and in some cases simulating it in a physics-based simulator if such a trajectory is found. This is a particularly time-consuming operation executed for every rearrangement action we evaluate during the search (taking on average $3-4$s per action). Lazy I−M4M can thus save considerable computational effort by evaluating actions as and when required.

Conventional lazy search algorithms assume that during the search the existence of an edge (v, v'), and by extension that of the successor state v', cannot be refuted and only its validity is determined upon evaluation. However in our domain the edge is generated as an abstract action based on the MAPF solution that when evaluated will almost surely achieve a successor state $v'' \neq v'$ if valid. This leads to a slightly modified implementation of the Lazy Weighted A* (LwA*) algorithm from [42] that we propose in this paper.

Until an edge (v, v') has been evaluated, we say that v' is an "unevaluated" state and afterwards it is a "fully evaluated" state. LwA* maintains duplicates of all states in the priority queue with different parents. When unevaluated states are expanded, LwA* evaluates the edge from its best parent. When fully evaluated states are expanded[4] LwA* grows the graph by adding unevaluated successor states. To

[4]Unevaluated copies of this state can be ignored/discarded in the future.

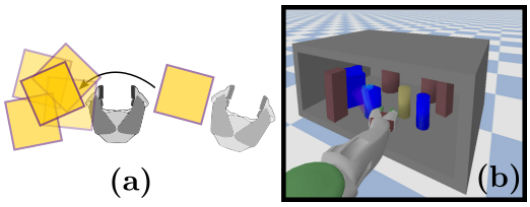

Fig. 5. (a) If the robot end-effect pushes an object whose mass and coefficient of friction is not known precisely, it may end up in one of many different possible final poses. (b) A screenshot from our simulator where multiple copies of an object with different parameters are being pushed. Irrelevant colours have been desaturated for ease of viewing. Different object copies can be seen in blue (opaque), and cyan and magenta (partially transparent)

deal with the fact that evaluating an edge to an unevaluated state can lead to a different fully evaluated state in Lazy I−M4M, our priority queue is allowed to contain duplicates of unevaluated states with different parents but not of fully evaluated states. This is because we cannot discard any unevaluated state on the basis of a different edge to it having been evaluated before and we need only keep track of the best parent/path to fully evaluated states. To implement this we store fully evaluated states in a hash table, and we do not hash unevaluated states.

D. Parallelised Simulations for Robustness to Parameter Uncertainty

E−M4M assumes that prior to planning we can perfectly localise all objects in the scene and we know their respective masses and coefficients of friction. This is difficult to satisfy in the real-world even with accurate localisation algorithms [43], [44] since masses can change over time and cannot be perceived by vision sensors, while coefficients of friction are difficult to compute accurately and change with the surface an object is placed on. If we do not know one or both of these parameters accurately, even with the use of a physics-based simulator, we cannot accurately simulate the result of a pushing action. Fig. 5 (a) shows a graphical 2D illustration of the potential results of pushing an object whose mass and coefficient of friction are not known accurately.

Fortunately, our use of a physics-based simulator for action evaluation makes it easy to make E−M4M (or any modified version of it) robust to these uncertain parameters. Taking inspiration from existing work [45]–[47], we utilise parallelised simulations of the same trajectory in scenes where each object is instantiated with different values of their mass and coefficient of friction. We assume known, bounded uncertainty on each of these parameters and given some budget on the number of parallelised simulators we are allowed to launch, we instantiate multiple copies of each object within each simulator. If the true mass and coefficient of friction for an object are m and μ respectively, for each object copy we sample $\hat{m} = \min\left(2m, \max\left(\frac{m}{2}, \mathcal{N}(m, 0.2m)\right)\right)$ and $\hat{\mu} = \min\left(2\mu, \max\left(\frac{\mu}{2}, \mathcal{N}(\mu, 0.5\mu)\right)\right)$. $\mathcal{N}(x, y)$ represents a sample from a 1D Gaussian distribution with mean x and standard deviation y.

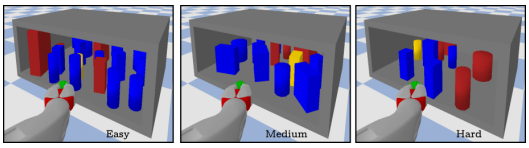

Fig. 6. Example Easy, Medium, and Hard scenes.

An action is said to be valid or not based on some user-defined threshold $\delta \in (0, 1]$ for success. Given N parallel simulators and M copies of each object, we say that an action is valid if $\max(1, \lfloor \delta N M \rfloor)$ samples are valid in simulation. Fig. 5 (b) shows a screenshot of a push being evaluated in our physics-based simulator with multiple object copies.

V. Experimental Results

A. Simulation Study

We ran all algorithms on 75 randomly generated MAMO problems categorised into three difficulty levels. Problems are Easy, Medium, or Hard depending on whether there are one, two, or more than two movable objects overlapping with the initial NGR. Each problem has 1 OoI, 4 immovable obstacles, and 5, 10, or 15 movable objects. Objects are randomly generated as cuboids or cylinders and their dimensions, mass, coefficient of friction, and initial configuration are all randomly initialised. Fig. 6 shows sample MAMO workspaces of each difficulty level. We test the performance of all algorithms on 25 Easy, 25 Medium, and 25 Hard problems with a 5 min planning timeout and use PyBullet [48] as our physics-based simulator.

Table I contains the number of problems solved by all algorithms, and for solved problems only the minimum/median/maximum total planning time and time spent simulating actions. We compare performance of E−M4M against: (i) I−M4M – depth-first search with pushing actions only, (ii) I−M4M-PnP – I−M4M with pick-and-place actions, (iii) Lazy I−M4M – eagerly lazy version with pushes only, and (iv) Lazy I−M4M-PnP – eagerly lazy version with pick-and-place actions added.

In a separate experiment we compare the performance of Robust I−M4M that uses parallelised simulations against $\widehat{\text{I−M4M}}$ – a version of I−M4M that instantiates a single copy of each object with mass and coefficient of friction sampled as described in Section IV-D. For this experiment a problem is successfully solved if the solution trajectory found by the planner does not violate any interaction constraints when executed in simulation with the true parameters for all objects. We only used pushing actions for this experiment since they are acutely affected by parameter uncertainty. Robust I−M4M used $N = 6$ parallel simulators each with $M = 2$ object copies and with $\delta = 0.9$ an action must be valid for 10 samples to be added to the graph.

All I−M4M versions solve more problems than E−M4M. Basic I−M4M solves problems faster and spends less time in simulation than E−M4M. This is because the planner tries to exploit the "goal-directed" rearrangement actions suggested by our MAPF solver that make deliberate progress towards

TABLE I

SIMULATION STUDY FOR MAMO PLANNING IN CLUTTERED SCENES

Metrics	Difficulty	Planning Algorithms						
		E-M4M	I-M4M	I-M4M-PnP	Lazy I-M4M	Lazy I-M4M-PnP	Robust I-M4M	$\widehat{\text{I-M4M}}$
Solved Problems	Easy	22	23	23	25	24	23	18
	Medium	20	24	22	25	23	23	17
	Hard	17	22	21	23	22	22	11
Total Planning Time (s)	Easy	0.5 / 18.6 / 79.3	0.5 / 12.1 / 46.3	15.8 / 27.3 / 279.0	0.4 / 9.9 / 53.6	6.2 / 11.6 / 266.7	8.1 / 11.3 / 284.9	7.4 / 14.3 / 177.1
	Medium	0.5 / 41.4 / 195.7	0.4 / 21.1 / 84.4	0.6 / 47.2 / 221.0	0.5 / 20.8 / 179.9	0.4 / 24.7 / 132.1	1.1 / 60.3 / 216.7	1.9 / 54.3 / 128.8
	Hard	9.8 / 55.3 / 188.5	13.8 / 35.6 / 212.0	9.6 / 61.2 / 206.2	9.3 / 29.7 / 144.6	12.0 / 40.9 / 187.7	19.2 / 83.6 / 295.9	22.9 / 104.7 / 257.4
Simulation Time (s)	Easy	0 / 6.4 / 58.0	0 / 5.9 / 28.0	0 / 15.8 / 196.5	0 / 5.9 / 40.5	0 / 6.5 / 119.7	0 / 6.2 / 253.8	0 / 5.4 / 77.3
	Medium	0 / 23.5 / 82.8	0 / 13.4 / 46.7	0 / 30.1 / 71.6	0 / 10.6 / 70.8	0 / 11.2 / 58.3	0 / 29.8 / 127.2	0 / 20.8 / 64.7
	Hard	11.3 / 34.9 / 154.9	6.8 / 25.8 / 139.9	5.2 / 36.6 / 120.4	12.5 / 21.5 / 98.5	5.7 / 23.1 / 76.1	10.5 / 51.5 / 200.5	8.6 / 41.9 / 138.1

clearing the NGR and solving the MAMO problem. The addition of prehensile rearrangement actions in I-M4M-PnP increases the branching factor of the search which increases planning and simulation times when compared against E-M4M. The major difference between the solutions found by I-M4M and I-M4M-PnP is observed in the number of rearrangement actions in their solution. I-M4M solves problems with 1.8 ± 0.8 (Easy), 2.9 ± 1.3 (Medium), and 6.2 ± 3.0 (Hard) pushes while I-M4M-PnP requires 1.7 ± 1.1 (Easy), 2.4 ± 0.9 (Medium), and 4.5 ± 1.8 (Hard) rearrangement actions in comparison. This is because if we find a valid pick-and-place rearrangement action, given the strict conditions required of it (Section IV-A), we exactly achieve the desired configuration of a movable object as suggested by our MAPF solver. As expected, Lazy I-M4M and Lazy I-M4M-PnP outperform their non-lazy versions in all metrics. Laziness is particularly effective in combating the computational overhead of a greater branching factor as the median planning time is almost halved between I-M4M-PnP and Lazy I-M4M-PnP.

In the second experiment, the timing statistics for Robust I-M4M and $\widehat{\text{I-M4M}}$ are comparable. However Robust I-M4M solves (successfully executes) many more problems than $\widehat{\text{I-M4M}}$. Robust I-M4M ensures that each action is considered valid only if it is valid for 90% of the samples, as opposed to $\widehat{\text{I-M4M}}$ which hopes to get lucky during execution because the solution found by the planner was only valid for one sampled value of all parameters.

B. Real-World Experiments

We solved 10 MAMO problems in the real-world with our PR2 robot using I-M4M-PnP. Fig. 7 shows the robot executing a solution where the PR2 first pushes the coffee can aside, then moves the tomato soup can via a prehensile action, before finally retrieving the OoI potted meat can. We used objects from the YCB dataset [49] for our experiments and they were localised using PERCH 2.0 [50]. We initialised each MAMO problem with either the tomato soup can or the potted meat can as the OoI and $2-4$ other movable objects. Although I-M4M-PnP found solutions to all problems, 3 out of 10 executions failed. In two of the runs, the sim-to-real gap led to discrepancies between the result of a pushing action. In one case an object toppled over when it did not in simulation, and another time an object was pushed into the OoI which is an interaction constraint violation that did not occur during planning. The third failure was due to the robot failing to grasp an object during a prehensile rearrangement action. This could be due to either small object localisation errors or due to trajectory execution errors.

Fig. 7. A real-world MAMO problem being solved by the PR2 using I-M4M-PnP. The potted meat can is the OoI (outlined in yellow), all other objects are movable.

VI. CONCLUSION AND DISCUSSION

This paper extends [1] and presents I-M4M, a fast and versatile MAMO solver – it utilises a diverse action space consisting of both nonprehensile and prehensile rearrangement actions, can be run lazily, and can leverage parallelised simulations to make it robust to inaccurate estimates of physics parameters of objects.

I-M4M spends the majority of its time simulating actions since it is difficult to satisfy all interaction constraints on contact, tilting, toppling etc. in our domain. To improve performance, we would like to incorporate learned rearrangement skills that are aware of the physical properties of all objects in the scene. We are also interested in extending I-M4M to deal with occlusions in the scene while maintaining the property that the solution will satisfy all interaction constraints at all times.

REFERENCES

[1] D. Saxena and M. Likhachev, "Planning for manipulation among movable objects: Deciding which objects go where, in what order, and how," *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 33, no. 1, pp. 668–676, Jul. 2023.

[2] R. Alami, T. Simeon, and J.-P. Laumond, "A geometrical approach to planning manipulation tasks. The case of discrete placements and grasps," in *The fifth international symposium on Robotics research* , H. Miura, Ed. MIT Press, 1990, pp. 453–463.

[3] 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.

[4] T. Siméon, J. Laumond, J. Cortés, and A. Sahbani, "Manipulation planning with probabilistic roadmaps," *Int. J. Robotics Res.*, vol. 23, no. 7-8, pp. 729–746, 2004.

[5] K. K. Hauser and J. Latombe, "Multi-modal motion planning in non-expansive spaces," *Int. J. Robotics Res.*, vol. 29, no. 7, pp. 897–915, 2010.

[6] D. M. Saxena and M. Likhachev, "Planning for complex non-prehensile manipulation among movable objects by interleaving multi-agent pathfinding and physics-based simulation," in *IEEE International Conference on Robotics and Automation, ICRA 2023, London, UK, May 29 - June 2, 2023*. IEEE, 2023, pp. 8141–8147.

[7] N. Jakobi, P. Husbands, and I. Harvey, "Noise and the reality gap: The use of simulation in evolutionary robotics," in *Advances in Artificial Life, Third European Conference on Artificial Life, Granada, Spain, June 4-6, 1995, Proceedings*, ser. Lecture Notes in Computer Science, F. Morán, A. Moreno, J. J. M. Guervós, and P. Chacón, Eds., vol. 929. Springer, 1995, pp. 704–720.

[8] Y. Chebotar, A. Handa, V. Makoviychuk, M. Macklin, J. Issac, N. D. Ratliff, and D. Fox, "Closing the sim-to-real loop: Adapting simulation randomization with real world experience," in *International Conference on Robotics and Automation, ICRA 2019, Montreal, QC, Canada, May 20-24, 2019*. IEEE, 2019, pp. 8973–8979.

[9] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Prez, "Integrated task and motion planning," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 4, no. 1, pp. 265–293, 2021.

[10] N. T. Dantam, Z. K. Kingston, S. Chaudhuri, and L. E. Kavraki, "Incremental task and motion planning: A constraint-based approach," in *Robotics: Science and Systems XII, University of Michigan, Ann Arbor, Michigan, USA, June 18 - June 22, 2016*, D. Hsu, N. M. Amato, S. Berman, and S. A. Jacobs, Eds., 2016.

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

[12] B. Kim, L. Shimanuki, L. P. Kaelbling, and T. Lozano-Prez, "Representation, learning, and planning algorithms for geometric task and motion planning," *The International Journal of Robotics Research*, vol. 41, no. 2, pp. 210–231, 2022.

[13] C. R. Garrett, T. Lozano-Prez, and L. P. Kaelbling, "FFRob: Leveraging symbolic planning for efficient task and motion planning," *The International Journal of Robotics Research*, vol. 37, no. 1, pp. 104–136, 2018.

[14] R. Chitnis, D. Hadfield-Menell, A. Gupta, S. Srivastava, E. Groshev, C. Lin, and P. Abbeel, "Guided search for task and motion plans using learned heuristics," in *2016 IEEE International Conference on Robotics and Automation, ICRA 2016, Stockholm, Sweden, May 16-21, 2016*, D. Kragic, A. Bicchi, and A. D. Luca, Eds. IEEE, 2016, pp. 447–454.

[15] 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 *Eighth International Workshop on the Algorithmic Foundations of Robotics, WAFR 2008*.

[16] 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.

[17] 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.

[18] 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.

[19] C. Nam, J. Lee, S. Cheong, B. Y. Cho, and C. Kim, "Fast and resilient manipulation planning for target retrieval in clutter," in *2020 IEEE International Conference on Robotics and Automation, ICRA 2020, Paris, France, May 31 - August 31, 2020*. IEEE, 2020, pp. 3777–3783.

[20] 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.

[21] 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.

[22] R. Wang, K. Gao, J. Yu, and K. Bekris, "Lazy rearrangement planning in confined spaces," *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 32, no. 1, pp. 385–393, Jun. 2022.

[23] J. E. King, J. A. Haustein, S. S. Srinivasa, and T. Asfour, "Nonprehensile whole arm rearrangement planning on physics manifolds," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 2508–2515.

[24] J. E. King, "Robust rearrangement planning using nonprehensile interaction," Ph.D. dissertation, Carnegie Mellon University, Pittsburgh, PA, December 2016.

[25] 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.

[26] E. Huang, Z. Jia, and M. T. Mason, "Large-scale multi-object rearrangement," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 211–218.

[27] 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.

[28] S. Park, Y. Chai, S. Park, J. Park, K. Lee, and S. Choi, "Semi-autonomous teleoperation via learning non-prehensile manipulation skills," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2022.

[29] 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.

[30] J. L. Barry, L. P. Kaelbling, and T. Lozano-Pérez, "A hierarchical approach to manipulation with diverse actions," in *2013 IEEE International Conference on Robotics and Automation, Karlsruhe, Germany, May 6-10, 2013*. IEEE, 2013, pp. 1799–1806.

[31] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling, "Backward-forward search for manipulation planning," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2015, Hamburg, Germany, September 28 - October 2, 2015*. IEEE, 2015, pp. 6366–6373.

[32] W. Vega-Brown and N. Roy, "Asymptotically optimal planning under piecewise-analytic constraints," in *Algorithmic Foundations of Robotics XII, Proceedings of the Twelfth Workshop on the Algorithmic Foundations of Robotics, WAFR 2016, San Francisco, California, USA, December 18-20, 2016*, ser. Springer Proceedings in Advanced Robotics, K. Goldberg, P. Abbeel, K. E. Bekris, and L. Miller, Eds., vol. 13. Springer, 2016, pp. 528–543.

[33] Z. K. Kingston, M. Moll, and L. E. Kavraki, "Sampling-based methods for motion planning with constraints," *Annu. Rev. Control. Robotics Auton. Syst.*, vol. 1, pp. 159–185, 2018.

[34] W. B. Thomason and R. A. Knepper, "A unified sampling-based approach to integrated task and motion planning," in *Robotics Research - The 19th International Symposium ISRR 2019, Hanoi, Vietnam, October 6-10, 2019*, ser. Springer Proceedings in Advanced Robotics, T. Asfour, E. Yoshida, J. Park, H. Christensen, and O. Khatib, Eds., vol. 20. Springer, 2019, pp. 773–788.

[35] C. R. Garrett, "Sampling-based robot task and motion planning in the real world," Ph.D. dissertation, Massachusetts Institute of Technology, USA, 2021.

[36] W. Thomason, M. P. Strub, and J. D. Gammell, "Task and motion informed trees (tmit*): Almost-surely asymptotically optimal integrated task and motion planning," *IEEE Robotics Autom. Lett.*, vol. 7, no. 4, pp. 11 370–11 377, 2022.

[37] S. B. Bayraktar, A. Orthey, Z. K. Kingston, M. Toussaint, and L. E. Kavraki, "Solving rearrangement puzzles using path defragmentation in factored state spaces," *IEEE Robotics Autom. Lett.*, vol. 8, no. 8, pp. 4529–4536, 2023.

[38] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Conflict-based search for optimal multi-agent path finding," in *Proceedings of the Fifth Annual Symposium on Combinatorial Search, SOCS 2012, Niagara Falls, Ontario, Canada, July 19-21, 2012*, D. Borrajo, A. Felner, R. E. Korf, M. Likhachev, C. L. López, W. Ruml, and N. R. Sturtevant, Eds. AAAI Press, 2012.

[39] N. Haghtalab, S. Mackenzie, A. D. Procaccia, O. Salzman, and S. S. Srinivasa, "The provable virtue of laziness in motion planning," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, S. Kraus, Ed. ijcai.org, 2019, pp. 6161–6165.

[40] A. Mandalika, S. Choudhury, O. Salzman, and S. S. Srinivasa, "Generalized lazy search for robot motion planning: Interleaving search and edge evaluation via event-based toggles," in *Proceedings of the Twenty-Ninth International Conference on Automated Planning and Scheduling, ICAPS 2019, Berkeley, CA, USA, July 11-15, 2019*, J. Benton, N. Lipovetzky, E. Onaindia, D. E. Smith, and S. Srivastava, Eds. AAAI Press, 2019, pp. 745–753.

[41] S. Mukherjee, S. Aine, and M. Likhachev, "MPLP: massively parallelized lazy planning," *IEEE Robotics Autom. Lett.*, vol. 7, no. 3, pp. 6067–6074, 2022.

[42] B. J. Cohen, M. Phillips, and M. Likhachev, "Planning single-arm manipulations with n-arm robots," in *Proceedings of the Eighth Annual Symposium on Combinatorial Search, SOCS 2015, 11-13 June 2015, Ein Gedi, the Dead Sea, Israel*, L. Lelis and R. Stern, Eds. AAAI Press, 2015, pp. 226–227.

[43] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes," in *Robotics: Science and Systems XIV, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA, June 26-30, 2018*, H. Kress-Gazit, S. S. Srinivasa, T. Howard, and N. Atanasov, Eds., 2018.

[44] J. Tremblay, T. To, B. Sundaralingam, Y. Xiang, D. Fox, and S. Birchfield, "Deep object pose estimation for semantic robotic grasping of household objects," in *2nd Annual Conference on Robot Learning, CoRL 2018, Zürich, Switzerland, 29-31 October 2018, Proceedings*, ser. Proceedings of Machine Learning Research, vol. 87. PMLR, 2018, pp. 306–316.

[45] M. C. Koval, J. E. King, N. S. Pollard, and S. S. Srinivasa, "Robust trajectory selection for rearrangement planning as a multi-armed bandit problem," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2015, Hamburg, Germany, September 28 - October 2, 2015*. IEEE, 2015, pp. 2678–2685.

[46] A. M. Johnson, J. E. King, and S. Srinivasa, "Convergent planning," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 1044–1051, 2016.

[47] W. C. Agboh and M. R. Dogar, "Robust physics-based manipulation by interleaving open and closed-loop execution," 2021.

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

[49] 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.

[50] 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.