

# Synchronized Multi-Arm Rearrangement Guided by Mode Graphs with Capacity Constraints

Rahul Shome and Kostas E. Bekris

Rutgers University, New Brunswick, USA

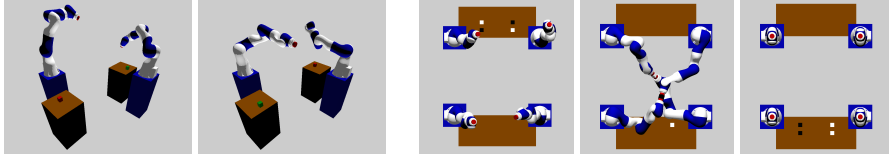
**Abstract.** Solving task planning problems involving multiple objects and multiple robotic arms poses scalability challenges. Such problems involve not only coordinating multiple high-DoF arms, but also searching through possible sequences of actions including object placements, and handoffs. The current work identifies a useful connection between multi-arm rearrangement and recent results in multi-body path planning on graphs with vertex capacity constraints. Solving a synchronized multi-arm rearrangement at a high-level involves reasoning over a modal graph, where nodes correspond to stable object placements and object transfer states by the arms. Edges of this graph correspond to pick, placement and handoff operations. The objects can be viewed as pebbles moving over this graph, which has capacity constraints. For instance, each arm can carry a single object but placement locations can accumulate many objects. Efficient integer linear programming-based solvers have been proposed for the corresponding pebble problem. The current work proposes a heuristic to guide the task planning process for synchronized multi-arm rearrangement. Results indicate good scalability to multiple arms and objects, and an algorithm that can find high-quality solutions fast and exhibiting desirable anytime behavior.

## 1 Introduction

Robotic arms are deployed in a variety of applications that involve pick-and-place tasks ranging from manufacturing to logistics and recycling. With the increasing affordability of such systems, and the availability of platforms like dual-arm humanoids, it is important to study how multiple robotic arms can expand upon the capabilities of individual arms and achieve faster execution. In some traditional deployments of multiple arms, such as automotive manufacturing, the environment is perfectly known and each arm performs its own independent task in relatively well-separated workspaces.

This work focuses on the case that the objects location are not pre-encoded and the arms are not sequestered. Instead the arms need to coordinate to solve object rearrangement tasks, where - beyond picking and placing - handoffs are also required to be performed. Such multi-arm rearrangement challenges are clearly computationally hard. The robotic arms are already high DoF systems. Coordinating multiple such arms to manipulate multiple objects results in an even larger configuration space. Furthermore, the overall planning problem involves searching both the continuous space of each robot and scheduling the discrete sequence of actions, i.e., picking, placing and handoffs.

Consider the example problem shown in Fig 1(left), which involves swapping objects between two tables not both reachable by a single arm. A greedy approach for



**Fig. 1.** SMAR problems: (*Left two:* ) A motivating problem involving switching the tables on which the objects lie. Such a problem already needs some high-level guidance to find a solution. (*Right three:* ) Steps in a larger problem instance involving 4 arms and 4 objects.

such an object swap – e.g., the left arm grasps the left object and the right arm grasps the right – results in a bottleneck, where both arms hold an object and are unable to perform the handoff. The solution is to transfer one of the objects first, and then move the second object. For this example, it may appear that enforcing moving one object at a time is a desirable approach. But this would be highly inefficient for the more general case of Fig 1(right), which involves more arms and objects. In such cases, it is also desirable to reduce the cost of the solution, typically corresponding to the makespan of the tasks, by simultaneously moving arms and manipulating multiple objects.

The current work focuses on a synchronized version of the problem, where the discrete actions of the arms (picking, placing and handoffs) are synchronized for a subset of the arms, i.e., arms can also be assigned a no-operation action. The paper studies the structure of such synchronized multi-arm rearrangement (SMAR), and argues that:

- Under assumptions, there is a novel analogy between SMAR and multi-agent path finding (MAPF) given a graph abstraction: an object-centric, mode-graph with capacity constraints. Such an abstraction has been recently studied in the MAPF literature [33]. Solutions to such problems map well to solutions to a class of SMAR problems as long as the underlying motion planning problems can also be solved.
- An integer linear programming (ILP) solution, defined by building on top of previous work [37], is practical and fast for the version of the MAPF over mode graph with capacity that is identified to be appropriate to model SMAR problems.
- This ILP solution for the MAPF over mode graph with capacity is effective in guiding the exploration of a forward search tree (SMART) for the problem.

Given these observations, the paper demonstrates the applicability to problems involving up to 9 arms and 4 objects in simulation, taking at most 10s for the harder cases.

## 2 Related Work

**Rearrangement:** Rearrangement planning [22] is a class of manipulation task and motion planning (TAMP) problems. Earlier work focused on efficient solutions to monotone instances [31]. Efficient solutions to the related assembly planning problem [9] also often assume monotonicity. Over the last decade, the focus incrementally moved to harder single-arm instances of rearrangement and manipulation task planning [19,20], and hybrid approaches [14]. Progress was made in studying the structure of hard problem instances that lead to efficient solutions [11]. The domain of object stacking [10] with a single arm was also explored. This motivated work on synchronized, dual-arm rearrangement [28], which managed to map the problem to a sequence of simpler sub-problems. The current work follows the same philosophy to lend structure to a subset of SMAR problems that leverages their combinatorial structure and allows for efficiency. The object-centric focus of the current work is similar to previous approaches [11,10].

The current effort, however, explicitly handles the complexity of dealing with multiple arms and identifies the relationship to capacity constraints on *pebble graphs*.

**Manipulation TAMP:** Early work focused on formalizing the problem’s multi-modal structure [29,13]. TAMP can also be approached in an integrated manner via constrained optimization formulations [34]. There are also hierarchical search strategies, which at a low-level call time-budgeted motion planning subroutines, and based on their outcomes, they guide the search [1,7] over actions in the task space [4,16]. Heuristics are important to effectively guide such TAMP algorithms. For instance, in multi-arm manipulation an effective heuristic is to consider the path for the object as a free-flying rigid body [3]. More recently multi-arm task planning has been studied using an answer-set programming-based hybrid approach [23]. Such hybrid approaches, which take into account symbolic constraints, can address a general set of task planning problems, and can guarantee probabilistic completeness. The current work focuses on the scalability of object rearrangement problems to multi-arm settings and considers aspects of solution quality as well. The asymptotic optimality of TAMP problems has been investigated [35,24]. As the number of robots increases in TAMP, so does the number of modes in the search space of task planning [5,12]. An efficient TAMP planner was proposed for single-object, multi-arm manipulation [25]. The current work builds on top of these efforts [13,35,25]. The current work focuses on rearrangement problems involving both multiple arms and multiple objects and aims to identify effective heuristics for an otherwise asymptotically optimal search of the overall search space.

**Motion Planning:** Sampling-based approaches have been a popular class of algorithms in motion planning research [18,21], including more recently asymptotically optimal (AO) variants [17,15]. Recent advances in sampling-based, multi-robot motion planning focused on high DoF systems, such as the dRRT\* method [6,27], which effectively decomposes the planning space, while guaranteeing completeness and asymptotic optimality [30]. The current work uses an underlying dRRT\*-like approach to compute simultaneous motions for multiple arms.

**Multi-agent Path Finding (MAPF):** Coupled approaches [30,36] operate in the composite, high-DoF configuration space. They can achieve completeness and optimality in principle but are often computationally intractable. Decoupled solutions [8,2] reduce the size of the search space by committing to individual agent solutions. They typically lack completeness and optimality. Fast, efficient coupled algorithms on graphs have been proposed [37], which formalize the optimal multi-agent path finding problem as integer linear program. Other efforts have focused on solving these problems with SAT solvers [32], and more recently by modeling vertex capacities [33]. The current work will draw the relationship between the graphical structures of MAPF problems [33] and borrows solution frameworks from the corresponding literature [37] to generate actions sequences for synchronized, multi-arm rearrangement.

### 3 Problem Setup and Terminology

Consider a workspace  $\mathcal{W} \subset SE(3)$ , which contains a set of obstacles, a set of  $r$  stationary robot arms  $\mathcal{M} = \{m_1, \dots, m_r\}$ , and a set of  $k$  objects  $\mathcal{O} = \{o_1, \dots, o_k\}$ . Each of the arms  $m_i$  has a state  $q_i$  in a  $d_i$ -dim. configuration space  $\mathbb{C}_{i \in [1 \dots r]} \subset \mathbb{R}^{d_i}$ . Each object can attain a pose  $p_j$  in  $\mathcal{P}_{j \in [1 \dots k]} \subset SE(3)$ . The combined planning space is:

$$\mathcal{T} = \prod_{i \in [1 \dots r]} \mathbb{C}_i \times \prod_{j \in [1 \dots k]} \mathcal{P}_j.$$

Let a task space state then be  $Q = (q_1, \dots, q_r, p_1, \dots, p_k) \in \mathcal{T}$ . A subset of this planning space is collision-free given all possible interactions of the arms, objects and obstacles, defined as  $\mathcal{T}_{\text{free}} \subset \mathcal{T}$ . The arms' end-effectors are allowed to touch the objects for grasping purposes in  $\mathcal{T}_{\text{free}}$ .

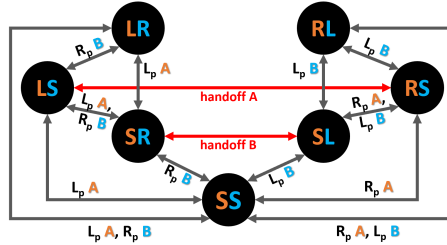
A path for an arm  $m_i$  is defined as  $\pi_i : [0, 1] \rightarrow \mathbb{C}_i$ . Define by  $\pi(t_1, t_2)$  the subsequence of the path between  $t_1 \rightarrow t_2$ , where  $t_1, t_2 \in [0, 1]$ . A composite path for all the arms  $\Pi : [0, 1] \rightarrow \prod_{i \in [1 \dots r]} \mathbb{C}_i$  is defined as the concurrent motion of all the arms. There are  $k + 1$  task modes of arm motions depending on interactions with the  $k$  objects: (i) **Move mode**: Moves (or transit) paths  $\pi_i^M(t_{\text{init}}, t_{\text{end}})$  are motions of an arm  $i$  when no object is carried by the end-effector. (ii) **Transfer mode**: Transfer paths  $\pi_i^T(t_{\text{init}}, t_{\text{end}})$  are motions of an arm  $i$  when an object is carried by the end-effector. There are  $k$  such transfer modes i.e., one per object.

A complementary set of  $r + 1$  task modes exist per object. Each object is constrained by either resting stably on a supporting surface or being grasped by an arm: (i) **Stable Pose**: A stable pose  $p_j^{\text{place}}$  is a pose for object  $j$  that is statically stable given a supporting surface, say a tabletop, which is otherwise an obstacle. (ii) **Grasped**: An end-effector can maintain a constant relative pose in  $SE(3)$  with an object when a *grasp* is engaged. Manipulation actions that affect the object states above corresponds to *picks*, *placements*. This is demonstrated in Fig 2. Additionally, a special interaction between two arms and one object introduces an additional action **Handoff**: an instantaneous grasp operation by one arm, and a release operation by another on the same object.

Let a set of object poses define an arrangement  $A = (p_1 \dots p_k)$ . A multi-arm manipulation path  $\Pi$  is valid if every task space state along it is collision free for all arms and objects.

**Rearrangement Problem**: An object rearrangement problem consists of an initial arrangement of objects at  $A_{\text{init}} = (p_1^{\text{init}} \dots p_k^{\text{init}})$  and a target arrangement  $A_{\text{goal}} = (p_1^{\text{goal}} \dots p_k^{\text{goal}})$ . Each pose in  $A_{\text{init}}$  and  $A_{\text{goal}}$  is a stable pose. A feasible solution to the rearrangement problem is a valid multi-arm manipulation path that transfers the objects from  $A_{\text{init}}$  to  $A_{\text{goal}}$ . Along this solution the objects are acted upon by sequences of *picks*, *places*, and *handoffs*.

The cost of the solution path  $\mathcal{C} : \prod \rightarrow \mathbb{R}$  is assumed to be the maximum of the Euclidean arc lengths for each  $\pi_i$  in  $\Pi$ , more commonly called makespan. The optimal solution to the multi-arm rearrangement problem  $\Pi^*$  is a feasible solution that minimizes the cost.



**Fig. 2.** The image shows the connectivity of different modes for a SMAR problem with 2 arms  $L$ , and  $R$  describing pick/place actions  $L_p, R_p$  and 2 objects  $(A, B)$ , with object-centric modes (S-stable, L-grasped by left, and R-grasped by right), black pick or placement edges, and red handoff edges.

### 3.1 Modeling Choices and Assumptions

A popular framework [13][5] for TAMP problems corresponds to searching over the space of modes by building a graph that transitions between neighboring modes at the task planning level. This approach makes motion planning calls within each mode so as to identify the transitions. The current work follows a similar framework and builds a tree in the s

**Conditions on the High-level Task Planner and Solutions Discovered:** In order to study the complexity of the combinatorially large problem of multi-arm rearrangement, the current work considers a simplification of the search-space. We assume that the task planning solution trajectory is decomposed into discrete steps along which the arms can perform *synchronized* execution of manipulation actions, similar to previous work [28], while allowing a subset of arms to not be involved in any action at a step. Effectively, the set of actions available to each arm is  $(pick, place, handoff, NOACT)$ , where *NOACT* means that the arm has not immediate objective during the current step. This narrows down the search space in two ways: a) each action has a fully defined set of choices (available objects for picks, available placement regions for place, and free arms for handoffs) which makes it sufficient to evaluate these combinations, and b) a sequence of steps is a sequence of actions being assigned to arms. This essentially poses steps as the discretization of time over the task planning solution. Note that during each step all the arms still have to perform centralized coordinated motions.

**Assumption 1** (*Synchronicity*) *The task planning solution is decomposed into a sequence of synchronized manipulation actions (picks, placements, handoffs, and NOACT), such that the end of the step is attained only when all the manipulators (except those assigned NOACT) complete their respective actions. It is assumed that a synchronized multi-arm rearrangement (SMAR) problem is solvable by such a synchronized solution.*

Now that the general problem has been simplified to narrow it down to a search over assignment of manipulation actions to arms over a sequence of steps, the next section goes on to describe how this can be efficiently solved, and then incorporated into a high-level task planner for efficiently guiding solutions.

**Conditions for the Heuristic to be Effective:** Despite the previous simplification, the number of possible choices available for the assignment of actions to arms and their sequence remains a prohibitively large space to naively search. We introduce additional set of assumptions to bring some structure to the search space, that can describe a subset of the synchronized multi-arm rearrangement problems.

**Condition 1** (*Discrete Placement Regions*) *Assume that there exists a set of discrete placement regions*

$$\mathcal{S} = \{S_1 \dots S_P\}, S_p \subset SE(3), p_j^{\text{place}} \in S_p \forall p_j^{\text{place}}.$$

*It is assumed that for each such placement region, the set of arms that can reach all the poses contained in the set is the same for every object pose in the region. Additionally, each placement region also has a specific capacity describing the maximum number of objects that can be guaranteed to concurrently lie in it.*

**Condition 2** (*Uniformity of Object Reachability*) *Though there is no explicit restriction on the objects being different, they must possess uniform reachability w.r.t. the decomposition of the discrete placement regions. This means, for every object the set of arms that can reach all poses in a specific placement region are identical. Note that the capacity of the placement region is also assumed to account for the different object sizes.*

This object reachability condition is trivially satisfied if the objects are identical. It should be pointed out that it is desirable to have a minimal number of placement regions, which permits not having to reason about individual poses during the high-level search, thereby reducing the search space. These capacity constrained placement regions can more accurately capture the *discretization* of the workspace (Fig 3). The sequence of arm actions required for a problem involving a specific combination of placement region should remain unchanged by the specific poses involved. This can promote the reuse of such high-level plans as well.

**Condition 3** (*Object Non-interactivity*) *For a pick action by an arm  $m$  performed on an object at pose  $p$ , if the pose lies in a reachable placement region, the feasibility of the pick action is independent of any other object. For a place action by an arm  $m$  performed on an object to take it to pose  $p$ , if the pose lies in a reachable placement region, the feasibility of the place action is only violated by an object currently at a pose intersecting with  $p$ .*

Note that object non-interactivity is guaranteed for a tabletop setup with overhead grasps where the start and goal poses of the objects do not intersect. The condition itself does not preclude non-monotonicity, as demonstrated in Section 7.4. Consider a problem that violates Condition 3 and is non-monotone: two objects placed one behind another inside a narrow shelf requiring grasping the deeper object first. Such problems lie outside the efficient subset of SMAR problems addressed in this work.

## 4 Mode Graph with Capacity Constraints

Despite the assumptions formulated in the previous section, as the number of arms and objects grows, given the number of available manipulation actions, the assignment of actions to arms, and sequencing them expresses a large search space (shown for only 2 arms and 2 objects in Fig 2). The possible actions available seem to create a notion of connectivity, where available actions are expressed by the setup of the workspace. For instance, if arm  $m_1$  can reach the placement region  $S_1$ , but  $m_2$  cannot, then a pick action on an object in  $S_1$  is not a valid action available to  $m_2$ . A key insight is that it might be possible to think of the problem in terms of the objects (as in previous work [3]) as they traverse the connectivity expressed by allowable actions performed upon them. Another key insight is that the connectivity only expresses one of the constraints. There exists a notion of *capacity* for manipulators and placement regions i.e., maximum number of objects involved in single manipulation actions.

Inspired by the formulation of the multi-agent path finding problem MAPF with vertex capacity constraints [33], a contribution of the current work is to pose a object-centric mode graph to express the SMAR.

**Definition 1.** (*Object-Centric Mode Graph*) An object-centric mode graph is a directed, weighted graph with vertex capacities. The vertices are either (i) placement regions or (ii) arms; edges map to manipulation actions with the cost of the action encoded in the edge weight; the capacities denote how many objects can occupy a vertex.

$$\begin{aligned} \mathbb{G}(\mathbb{M}; \mathbb{E}), \quad & \text{has nodes } \mathbb{M} = \{\mathbf{m} \in \mathcal{S} \cup \mathcal{M}\}, \quad \text{and edges } \mathbb{E} = \{e(\mathbf{m}_u, \mathbf{m}_v)\} \\ e(\mathbf{m}_u, \mathbf{m}_v) \in \mathbb{E} \quad & \text{if } \begin{cases} \text{Pick: } \mathbf{m}_u \in \mathcal{S}, \mathbf{m}_v \in \mathcal{M}, \text{ and } \mathbf{m}_v \text{ can reach } \mathbf{m}_u \\ \text{Place: } \mathbf{m}_u \in \mathcal{M}, \mathbf{m}_v \in \mathcal{S}, \text{ and } \mathbf{m}_u \text{ can reach } \mathbf{m}_v \\ \text{Handoff: } \mathbf{m}_u \in \mathcal{M}, \mathbf{m}_v \in \mathcal{M}, \mathbf{m}_u \neq \mathbf{m}_v, \text{ and } \mathbf{m}_u \text{ can reach } \mathbf{m}_v \end{cases} \\ \mathbb{W} : \mathbb{E} \rightarrow \mathbb{R} \quad & \text{are the edge weights} \\ \mathbb{F} : \mathbb{M} \rightarrow \mathbb{W} \quad & \text{are the vertex capacities} \\ \mathbb{F}(\mathbf{m}_u) = \begin{cases} 1 & \text{if } \mathbf{m}_u \in \mathcal{M} \\ \text{number of objects that can fit in } \mathbf{m}_u & \text{if } \mathbf{m}_u \in \mathcal{S} \end{cases} \end{aligned}$$

Each mode  $\mathbf{m}$  in the set of vertices  $\mathbb{M}$  corresponds either to placement regions  $\mathcal{S}$  or arms  $\mathcal{M}$ . This means that  $\text{card}(\mathbb{M}) = \text{card}(\mathcal{S}) + \text{card}(\mathcal{M})$ .

Given  $k$  objects, let the set of modes they instantaneously occupy define a *multi-modal state* on the mode graph,  $\mathbb{Q} = (\mathbf{m}_1, \dots, \mathbf{m}_k)$ .

By definition of the placement regions  $\mathcal{S}$ , a stable arrangement consists of objects lying in a set of nodes of the mode graph  $\mathbb{G}$ . This means,  $A_{\text{init}} = (p_1^{\text{init}} \dots p_k^{\text{init}})$ ,  $p_j^{\text{init}} \in \mathcal{S}_p \in \mathbb{M} \quad \forall p_j^{\text{init}}$ . Define this mapping as  $\mathbb{M}^{-1}(A_{\text{init}}) = \mathbb{Q}^{\text{init}}$ . This defines a set  $(p_1^{\text{init}} \dots p_k^{\text{init}})$  maps to a set of mode-graph nodes  $\mathbb{Q}^{\text{init}} = (\mathbf{m}_1^{\text{init}} \dots \mathbf{m}_k^{\text{init}})$  for  $o_j$ .

Similarly define  $\mathbb{Q}^{\text{goal}} = (\mathbf{m}_1^{\text{goal}} \dots \mathbf{m}_k^{\text{goal}})$  for  $A_{\text{goal}}$  for each object  $o_j$ . With a slight abuse of notation, define an inverse mapping a fully defined task space configuration maps to a set of modes on  $\mathbb{G}$ , such that  $\mathbb{M}^{-1}(Q) = \mathbb{Q}$ .

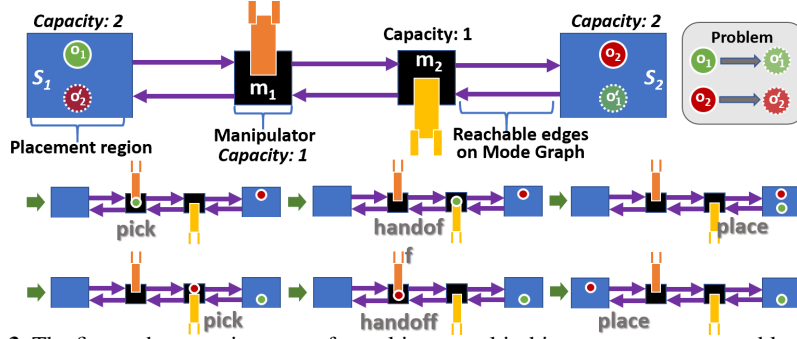
*Connection to Task Modes:* Compared to Fig 2,  $L, R$  map to modes for the arms in  $\mathbb{G}$ , All the placement regions correspond to  $\mathcal{S}$ . For 2 objects, their multi-modal state  $\mathbb{Q}$  lies on one of the nodes in Fig 2. Transitioning to an adjacent  $\mathbb{Q}$  moves both objects, and traverses two edges in  $\mathbb{G}$  and one edge in the task mode graph.

By the formulation of feasible multi-arm manipulation solutions, each arm executes a sequence of *moves* and *transfers*, where an arms can manipulate a specific object by *picking*, *placing*, or *handoff*. For each object  $o_j$  during the execution of a valid  $\Pi$  denote the discrete manipulation actions by the tuples  $(m_i, t^j)$  for picks,  $(p_p, t^j)$  for placements, and  $(m_i, t^j)$  for handoff to  $m_i$  at instants  $\Pi(t^j)$  respectively. As an illustrative example,  $\Pi$  yields the following sequences of timed manipulation actions from the initial to target poses of each of the  $k$  objects, using *picks*, *placements*, and *handoffs*.

This describes:  $o_j \xrightarrow{\Pi} [(p_j^{\text{init}}, 0), (m_i, t_j^1), \dots, (p_j^{\text{goal}}, 1)] \quad \forall o_j$

Each of the poses in the above sequences correspond to some placement region  $\mathcal{S}$  by Assumption 1. Each pick and handoff corresponds to an arm in  $\mathcal{M}$ . This means every tuple in the sequences above correspond to some  $(\mathbf{m}, t)$  where  $\mathbf{m} \in \mathbb{M}$  in the mode graph at a specific time parametrization  $t$ .

**Definition 2.** (*Multi-agent Path Finding (MAPF)*) Given a starting configuration  $\mathbb{Q}^{\text{init}}$ , and a desired final configuration is  $\mathbb{Q}^{\text{goal}}$  on the mode graph  $\mathbb{G}$ , a multi-agent path



**Fig. 3.** The figure shows an instance of a multi-arm multi-object rearrangement problem, and the corresponding mode graph  $\mathbb{G}$ , with the MAPF solution obtained for affecting the rearrangement of both the red and green objects across the two placement regions  $S_1$  and  $S_2$ .

*finding solution finds a sequence of valid vertices and edges for each object that takes them from their start to goal vertices.*

A solution to the MAPF problem defines for every object a sequence of nodes on the graph along with the time at which the object starts occupying the node, such that each object begins at the initial node, and ends at the target node. Let  $\Pi_{\mathcal{O}}$  be a feasible solution to the MAPF problem of  $\mathcal{O}$  agents on  $\mathbb{G}$ , consisting of a sequence of multi-modal states  $\Pi_{\mathcal{O}} = [Q^{\text{init}}, \dots, Q^{\text{goal}}]$ . In terms of each object:

$$\Pi_{\mathcal{O}} = \left\{ \pi_{o_j} : [(\mathbf{m}_{\text{init}}^j, 0), \dots, (\mathbf{m}_l^j, t) \dots (\mathbf{m}_{\text{goal}}^j, 1)] \quad \forall \pi_{o_j} \right.$$

**Constraints on motions:** Certain constraints on the allowable motions of the objects over the mode graph  $\mathbb{G}$  respect the restrictions of the problem. The following two are derived from the classical formulation of MAPF: (i) at any time  $t$ , no vertex  $\mathbf{m}$  exceeds its capacity  $\mathbb{F}(\mathbf{m})$ , and (ii) agents do not cross each other over the same edge. Vertex capacities in  $\mathbb{G}$  is an additional constraint that needs to be imposed to make the problem amenable to multi-arm rearrangement. *Vertex capacities cap the maximum number of objects that can move over all the in-edges and out-edges of every vertex at any step.*

**Cost of MAPF:** Each edge on the graph has a weight defined by  $\mathbb{W}$ . For each object, the solution cost is  $\mathcal{C}(\pi_{o_j}) = \sum \mathbb{W}(\mathbf{e}_t)$ ,  $\forall \mathbf{e}_t$  in the solution. The cost of the solution  $\mathcal{C}(\Pi_{\mathcal{O}})$  is typically defined as the maximum of each component  $\mathcal{C}(\pi_{o_j})$ .

**Optimal MAPF:** An optimal MAPF solution,  $\Pi_{\mathcal{O}}^*$  is a feasible MAPF solution that also minimizes the cost of the MAPF solution.

**Theorem 1.** *(Multi-arm Rearrangement Solution Solves MAPF on  $\mathbb{G}$ ) Each feasible solution to the multi-arm multi-object rearrangement problem  $\Pi$  for  $r$  arms and  $k$  objects from arrangement  $A_{\text{init}} = (p_1^{\text{init}} \dots p_k^{\text{init}})$  to  $A_{\text{goal}} = (p_1^{\text{goal}} \dots p_k^{\text{goal}})$  reduces to a feasible solution to a multi-body path planning problem of  $k$  agents on  $\mathbb{G}$  defined in Def 1 from the corresponding  $Q^{\text{init}}$  to  $Q^{\text{goal}}$ .*

*Proof.* By construction, there is a one-to-one correspondence between the manipulation actions necessary for the rearrangement problem and edges in  $\mathbb{G}$ . Solutions in the complex task planning space map to the solution to the MAPF problem on a graph, where each object starts moving at the timestamp corresponding to action involving the object in the task planning solution  $\Pi$ , before progressing to the next vertex on  $\mathbb{G}$ . At the end of the solution, every object ends up in the target vertices on  $\mathbb{G}$ .  $\square$

**Definition 3.** (Discrete MAPF on  $\mathbb{G}$ ) In this problem, starting from an initial set of vertices  $\mathbb{Q}^{\text{init}}$  in  $\mathbb{G}$ , at each discrete step  $t$  each object can move to an adjacent vertex or stay in place. Each edge traversal incurs a cost corresponding to the edge weight. After tracing the solution steps each object reaches the final configuration  $\mathbb{Q}^{\text{goal}}$ .

Note that in the discrete version, every time step and edge traversal is atomic. This aligns with the synchronicity assumption of SMAR as well. The number of edges in the solution, per object, represents the number of *picks, places or handoffs* involved. For each object the cost of the solution being the number of actions is analogous to counting the number of edges in the object’s solution, i.e., if  $\mathbb{W} : \mathbb{E} \rightarrow 1$ ,  $\mathcal{C}(\pi_{o_j}) = \#\text{actions}$ .

**Theorem 2.** (An action optimal MAPF solution is an admissible heuristic for Synchronized Multi-arm Rearrangement) The number of steps in the MAPF solution over  $\mathbb{G}$  is less than or equal to the number of synchronized actions (*picks, places, and handoffs*) in the corresponding synchronized multi-arm rearrangement solution.

*Proof.* The cost of the solution is  $\mathcal{C}(\Pi_{\mathcal{O}}) = \max(\mathcal{C}(\pi_{o_j}), \dots, \mathcal{C}(\pi_{o_j}))$ . Using this cost, the optimal solution  $\Pi_{\mathcal{O}}^*$  minimizes the maximum number of actions on any object. This analogy is applicable under the assumption set up in Section 3.1. The synchronicity assumption allows for these synchronized actions that reflect the atomic multi-agent steps over the mode graph. Additionally, the object non-interactivity, and reachability assumptions ensure that the edges on the mode graph are effectively independent of each other and reflect *feasible* and *reachable* actions for the arms.  $\square$

In general the feasibility of high-dimensional motion planning for each manipulation action on the MAPF solution is not guaranteed. The MAPF solution can still be useful, however, as a suggested sequence of actions that can be checked first for feasibility.

**Observation 1** (A MAPF solution on  $\mathbb{G}$  describes a sequence of actions for each arm) A MAPF solution to  $\mathbb{Q}^{\text{goal}}$  over the mode graph corresponds to a sequence of manipulation actions, which if collision-free will bring the objects to the poses in  $\mathbb{Q}^{\text{goal}}$ .

A configuration of arms and objects  $Q^{\text{current}} \in \mathcal{T}$  describes the corresponding multi-modal state  $\mathbb{Q}^{\text{current}} = \mathbb{M}^{-1}(Q)$ . Define the MAPF problem from  $\mathbb{Q}^{\text{current}}$  to  $\mathbb{Q}^{\text{goal}}$ . The MAPF solution  $\Pi_{\mathcal{O}} = (\mathbb{Q}^{\text{current}}, \mathbb{Q}^{\text{next}}, \dots, \mathbb{Q}_l \dots \mathbb{Q}^{\text{goal}})$  describes a sequence of vertices on  $\mathbb{G}$  and corresponding edges or actions (*picks, places and handoffs*) for each object and arm.

$$\mathcal{H} = (\mathbf{e}_1, \mathbf{e}_i, \dots, \mathbf{e}_r), \mathbf{e}_i = \begin{cases} \emptyset, & \text{if arm } m_i \text{ not in solution } \Pi_{\mathcal{O}} \\ \mathbf{e}, & \text{the first action involving arm } m_i \text{ in } \Pi_{\mathcal{O}} \end{cases}$$

A slightly different estimate is used as the multi-modal goal  $\mathcal{G} = (\mathbf{e}_1, \mathbf{e}_i, \dots, \mathbf{e}_r)$ , which comprises of an action per arm required next, i.e.,  $\mathbf{e}_i = \emptyset$  if arm  $m_i$  not used to transition to  $\mathbb{Q}^{\text{next}}$ . Otherwise the corresponding the action involving the arm  $i$  in  $\mathbb{Q}^{\text{next}}$  is used. Thus, we obtain a longer horizon heuristic and an immediate biasing goal for defining the next action of each arm to trace the MAPF solution towards  $\mathbb{Q}^{\text{goal}}$ .

Note that while  $\mathcal{G}$  makes stepwise progress along the MAPF solution, the heuristic expresses something more powerful. By biasing towards the heuristic, each arm can *pre-empt* what is required of it next, even if it is farther into the future than one single step of the MAPF solution.

## 5 Integer Linear Program for MAPF on Mode Graph

This section outlines the solution to the MAPF problem over the mode graph. An integer linear programming model is set up on the lines of previous work [37]. At a high-level, the method takes as input the mode graph, and constructs a *time-expanded* graph where a) nodes are replicated across time slices, and b) each pair of bidirectional (or undirected) edge is replaced with a gadget (Fig 4) across two time slices. By connecting the start and goal nodes between the first and last time slices, the solution to the multi-commodity flow problem over the time-expanded graph ( $G^T$ ) describes an MAPF solution.

For the sake of brevity most of the replicated details are omitted here. The specific addition required to guarantee solutions that are usable over the mode graph, formulates the capacity constraints at the modes. Inspired by previous work [33] the notion of capacity has to be modeled correctly. These capacity constraints essentially encode the restriction that *through* any instant step (or time slice) at capped number of objects can interact with the mode.

Let  $v$  denote a node in the time expanded graph corresponds to some mode  $\mathbf{m}_v$ , and  $e_j$  denote each edge. There are indicator variables assigned to each robot ( $i \in [1 \dots k]$ ) moving across an edge  $e_j$  as  $x_{i,j}$ . Let  $\delta^+(v)$  be the out-edges and  $\delta^-(v)$  be out-edges.

$$\textbf{Capacity Constraint: } \sum_{e_j \in \delta^+(v)} x_{i,j} + \sum_{e_j \in \delta^-(v)} x_{i,j} \leq \mathbb{F}(\mathbf{m}_v) \forall v \in G^T, 1 \leq i \leq k$$

Minimizing the maximum traveled distance with uniform modal edge costs minimizes the maximum number instantaneous actions in the MAPF solution over  $G$ .

**Implementation Details:** The current objective is to use an underlying MAPF solver as a quick heuristic over the mode graph  $G$ . It is therefore beneficial to encode some measure of the cost of the actions represented by edges in the mode graph in MAPF.

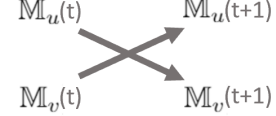
**Costs:** The cost of an edge in the mode graph provided by the  $W$  function is assumed to be the position distance in  $\mathbb{R}^3$  of the a) centroid of the placement region poses, b) the positions of the root frames for each arm. The objective remains minimizing the maximum traveled distance on  $G^T$ .

**ILP Invocation:** In previous work [37] the underlying ILP model was augmented for a range of time-steps ( $T_{\min}$  to  $T_{\max}$ ) and called repeatedly. The current work needs the solution to be obtained ideally in a single ILP call. We restrict the number of ILP invocations by the following change:

$$T_{\max} = r \times k. \quad \mathcal{C}'(e_{i,j}) = \text{time}(e_{i,j}) \cdot \mathcal{C}(e_{i,j})$$

where  $\text{time}(e_{i,j})$  returns the time slice corresponding to the source of  $e_{i,j}$ . Using the reformulated cost  $\mathcal{C}'$ , in combination with the weighted edges, we might lose some of the strict optimality guarantees, but this lets us get solutions which a) reason about heuristic costs, and b) try to minimize the number of time steps the robots move (i.e., prefer to lower maximum number of actions).

The solution can be retrieved by pruning the end of the solution where the objects reach their target modes and stay still for the remainder of the time-steps. The results from experiments indicate that the modified MAPF solver remains useful as a heuristic.



**Fig. 4.** Gadget for constructing the time-expanded graph from the mode graph.

## 6 Integrated Task and Motion Planning

An integrated task and motion planning approach has to simultaneously explore (i) arm configurations  $\mathbb{C}_1 \times \dots \mathbb{C}_r$  for every arm, and (ii) *picks, places, and handoffs* that change object poses for every object.

The underlying search is similar to previous work [25], which described a multi-modal integrated task and motion planning algorithm using an underlying dRRT\*[27]-like decomposition. It has been shown in previous work [27,28,25] that this search strategy is critical for solving these high-dimensional multi-robot problems. The key differences in the proposed approach are outlined below.

a) the integration the sampling of transition configurations that achieve neighboring modes inside the online search process. For instance, if a neighboring mode on  $\mathbb{G}$  involves the arm  $m_i$  picking up  $o_j$ , an IK solver can be invoked to find a set of grasping configurations to add as possible grounded configurations to plan to.

b) A multi-body path planning heuristic over the mode-graph is used to bias the high-level search, that determines what arm actions would be preferred. In a lot of problem instances, the search space is so large, and the multi-body coordination so constraining, this heuristic proves crucial to not only solution quality, but also feasibility within a limited time budget. Next we describe some underlying modules that are assumed to be available to the algorithm.

**Mode-graph Generator:** There is a module which is aware of the spatial configuration of all the robots and placement regions, and their respective reachability, in order to generate a set of placement regions satisfying Assumption 2, returning  $\mathbb{G}$ .

**MAPF Solver:** An MAPF solver subroutine is assumed to be available (as is described in Section 5). Given an initial multi-modal state, the module returns the multi-body path planning solution for the objects over the mode graph with capacity constraints, that ideally minimizes the maximum path cost by of object.

**Transition Sampler:** It is assumed that a transition sampler can generate complete configurations defining manipulation actions like picks and placements involving an object and an arm, or handoffs involving two arms and an object. The sampler should be able to provide any number of these mode transitions, whenever invoked.

### 6.1 Algorithm

Bringing together all the tools described so far, a forward search tree to solve synchronized multi-arm rearrangement (SMART) is outlined in this section. The method constructs a search tree  $\mathbb{T}$  of task space configurations  $Q \in \mathcal{T}$ , rooted at  $Q_{\text{init}}$  with the objects at  $A_{\text{init}}$ , the objective being to find a continuous sequence of motions that reach a state such that the objects are in their final arrangement  $A_{\text{goal}}$ .

Each vertex of the tree consists of  $Q$  and an unique identifier  $\tau$  keeping track of the sequence of manipulation actions that led to the current tree vertex. This identifier is similar to *orbits* described in previous work [35]. The high-level algorithm is described in Algo 1. The method first builds a mode graph calling the subroutine `build_mode_graph`. As mentioned, an internal counter must keep track of different sequences of actions ( $\tau_{\text{count}}$ ).

During each iteration, a `select_mode` selects a mode  $\mathbf{m}$  and the transition id  $\tau$  from the tree. A fraction of the invocations, the subroutine is designed to *goal bias* by selecting modes that have made the most progress towards the goal. If the selected  $\tau$  has never been expanded before, the heuristic function for it would be empty. In such a case, the multi-body motion planning subroutine MAPF is called to obtain a sequence of usable actions ( $\mathcal{H}$ ) that can guide the arms. The computation of this heuristic follows Theorem 1. If not invoked before, or a fraction of the expansions, neighboring grounded transition configurations are added. This is tantamount to inspecting the current multimodal state and expanding the set of available *grasping, placement or handoff* configurations. The subroutines `select`, `extend`

and `rewire` follow standard definitions except the restriction that they operate on the set of vertices that have the same id  $\tau$ . The heuristic is used in `select` and `extend` to bias towards making progress towards various modal goals. Specific consideration has to be provided for modal guidance that is inherently coupled (hand-offs) versus single arm targets (picks or places). If the extension is valid and collision-free, and the new node satisfies a transition to an adjacent mode, the search can progress to an adjacent multimodal state through the current transition sequence (tracked by incrementing  $\tau$ ). If the adjacent mode attains the target arrangement, the solution path  $\Pi$  is updated if the cost improves.

*Note on synchronicity:* In the algorithm, the synchronicity assumption is encoded into the function `modal_check` which checks against the multimodal goals like  $\mathcal{G}$  returned by the MAPF solver or fully defined multimodal states defined by the `sample_trans` subroutine. This restricts the search from adding new *transition id-ed* components to  $\mathbb{T}$  every time partial progress has been made, and enforces the synchronization assumption.

---

**Algorithm 1:** SMART( $Q_{\text{init}}, A_{\text{goal}}$ )

---

```

1  $\Pi \leftarrow \emptyset; \quad \tau_{\text{count}} \leftarrow 0;$ 
2  $\mathbb{G} \leftarrow \text{build\_mode\_graph}();$ 
3  $\mathbb{T}.V \leftarrow Q_{\text{init}}, \text{increment}(\tau_{\text{count}}) >;$ 
4 for  $\text{max\_iters}$  do
5    $(\mathbf{m}, \tau) \leftarrow \text{select\_mode}();$ 
6   if  $\mathcal{H}(\tau) = \emptyset$  then
7      $\mathcal{H}(\tau) \leftarrow \text{MAPF}(\mathbf{m}, \mathbb{M}^{-1}(A_{\text{goal}}));$ 
8   if  $\mathcal{H}(\tau) \neq \emptyset$  or add_fraction() then
9      $\text{sample\_trans}(\text{adj}(\mathbb{G}, \mathbf{m}), t);$ 
10     $Q_{\text{near}} \leftarrow \text{select}(\tau, \mathcal{H});$ 
11     $Q_{\text{new}} \leftarrow \text{extend}(Q_{\text{near}}, \tau, \mathcal{H});$ 
12     $Q_{\text{best}} \leftarrow \text{rewire}(Q_{\text{new}});$ 
13    if  $\Pi(Q_{\text{best}} \rightarrow Q_{\text{new}}) \in \mathcal{T}_{\text{free}}$  then
14       $\mathbb{T}.V \leftarrow \mathbb{T}.V \cup Q_{\text{new}}, \tau >;$ 
15       $\mathbb{T}.E \leftarrow \mathbb{T}.E \cup$ 
16         $(\langle Q_{\text{best}}, \tau \rangle, \langle Q_{\text{new}}, \tau \rangle);$ 
17      if modal_check( $\mathbb{M}^{-1}(Q_{\text{new}}), \tau$ ) then
18         $\tau_{\text{new}} \leftarrow \text{increment}(\tau_{\text{count}});$ 
19         $\mathbb{T}.V \leftarrow \mathbb{T}.V \cup \langle Q_{\text{new}}, \tau_{\text{new}} \rangle;$ 
20         $\mathbb{T}.E \leftarrow \mathbb{T}.E \cup$ 
21           $(\langle Q_{\text{new}}, \tau \rangle, \langle Q_{\text{new}}, \tau_{\text{new}} \rangle);$ 
22      if  $\mathbb{M}^{-1}(Q_{\text{new}}) = \mathbb{M}^{-1}(A_{\text{goal}})$  then
23         $\Pi_{\text{new}} \leftarrow \text{retrace}(Q_{\text{new}});$ 
24        if  $\mathcal{C}(\Pi_{\text{new}}) < \mathcal{C}(\Pi)$  then
25           $\Pi \leftarrow \Pi_{\text{new}};$ 
26 return  $\Pi$ 
```

---

## 7 Results

This section goes over the experimental evaluations performed to demonstrate the effectiveness of the proposed approach. The same underlying task planning framework is used in each experiment. The key metrics we want to measure are *the time it takes to find the initial solution*, *the solution cost returned after 30s*, *the number of actions*, and *success ratio*. All experiments were run on a single core of an Intel(R) Xeon(R) CPU E5-1660 v3 @ 3.00GHz processor having a maximum available 16GB of RAM, and data is reported averaged over 20 trials.

**Comparison Points:** Three strategies are used as guidance in the task planner.

**SMART** (proposed): Algo 1 uses the MAPF over the mode graph with capacity constraints to the goal arrangement of objects in the mode graph, and uses this solution as the guidance from the current mode.

**Sequential:** The objects are prioritized in an arbitrary way that is fixed per experiment. The *Sequential* heuristic solves the MAPF problem over the mode graph *for the next remaining object*. This chains a sequence of single-object solutions.

**Greedy:** The *Greedy* heuristic solves the MAPF problem for every object, and guides towards the next mode for *every* object greedily. Any conflicts that might arise from this guidance have to be overcome by the exploration in the underlying task planner.

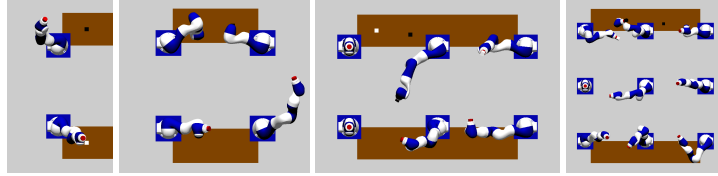


Fig. 5. Benchmarks setups with (left to right) two, four, six, and nine arms respectively.

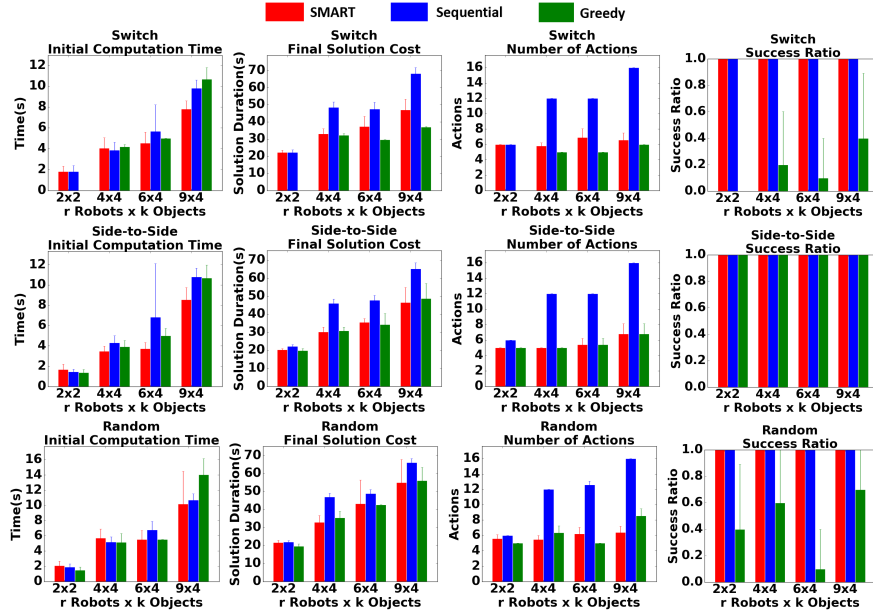
**Benchmarks:** The experiments inspect the problem of transferring objects between two tables at two ends of the workspace using a set of 7DoF *Kuka iiwa14* manipulator. Fig 5 outlines the combinations of  $r \times k$  as the number of robots and objects used in four different such setups:  $2 \times 2$ ,  $4 \times 4$ ,  $6 \times 4$ ,  $9 \times 4$ . Note that simply the centralized motion planning problem for 9 robots lies in a 63-dimensional space.

To simplify manipulation the objects are  $6cm \times 6cm$  cubes with a grasp on each side, and 5 IK solutions for each (*pick*, *place*, or *handoff*), whose computation is included in the reported times. Three of the benchmarks have non-intersecting starts and goals.

**Switch:** In this benchmark objects are split into two halves. Each set has poses sampled on one of the tables and final poses on the other one. For the other half the transfer direction is reversed. Given the setup, this can cause bottlenecks for *Greedy* which can get stuck if the problem involves poses on opposite placement regions.

**Side-to-side:** In this benchmark all the objects have their initial poses sampled on one of the tables, while the final poses are sampled on the other table. This promotes concurrent transfers using all the arms adjacent to the tables.

**Random:** In this benchmark, for every object the start pose is selected on either table, and the target lies on the opposite table. This in essence is a combination of the other two benchmark with the direction of the transfers randomized.



**Fig. 6.** Benchmarks per row from top to bottom, the switch, side-to-side, and sort benchmark data is reported. From left to right, (*First:*) shows the initial computation times, (*Second:*) shows the solution duration after 30s of computation, (*Third:*) shows the total number of discrete instants of object transitions or actions in the solution, and (*Fourth:*) the success ratio.

### 7.1 Switch Benchmark

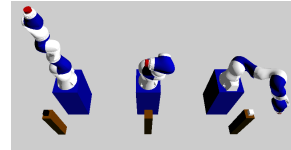
The data as shown in Fig 6(*top*). *Greedy* always gets bottlenecked in  $2 \times 2$ . With more manipulators, the *Greedy* success stays low. Note that both SMART and *Sequential* have similar performance for  $2 \times 2$  since the solutions should be very similar. As the number of robots increases SMART the initial times are either comparable or better than *Sequential*, while providing better solutions, and far fewer manipulation actions (since SMART minimizes this). The improvement in the number of actions is not identical to the solution durations since there is an overhead of arm coordination that might exist since SMART allows an arbitrary number of arms to interact at any time.

### 7.2 Side-to-side Benchmark

The data as shown in Fig 6(*middle*). All three comparison points succeed in this problem. SMART needs far fewer actions and a faster initial solution time. The costs are comparable for SMART and *Greedy*, while being better than *Sequential* solutions. For instance in the  $9 \times 4$  case SMART has a 20s speedup compared to *Sequential*.

### 7.3 Random Benchmark

The data is shown in Fig 6(*bottom*). In terms of computation times SMART is similar or better to *Sequential*. *Greedy* shows poor performance with bottlenecks and fails often. Its solutions are similar to SMART when it works. As the robots' number increases, its time increases but SMART manages to solve most problems under 10s with fewer manipulation action instants and better solution costs.



**Fig. 7.** A non-monotone in-place swap demonstration.

#### 7.4 Non-monotone Demonstration

Fig 7 shows a problem that involves in-place swapping of two objects. The problem necessitates both the use of a buffer, as well as multiple interactions between an object and arm. This lies in the class of efficiently solvable problems that are non-monotone (and satisfy the object non-interactivity condition). On average using SMART discovered an initial solution in 3.41s, succeeding every time with a 8 step solution spanning 34.6s.

### 8 Discussion

The current work demonstrates the connection between Synchronized Multi-Arm Rearrangement (SMAR) problems and Multi-Agent Path Finding (MAPF). The link corresponds to an object-centric mode graph with capacity constraints, for which there are efficient solvers. These MAPF solutions are shown to be beneficial as heuristics in large scale SMAR problems involving 9 arms and 4 objects, as well as a non-monotone demonstration. There are various aspects of the mode-graph that can be explored in future work. The removal of the object-non interactivity condition to encode other constraints, or dealing with mobile manipulators can lead to efficient solutions to complex non-monotone challenges. Improvements to the task planner can also allow removing the synchronization assumption. Increasing the number of objects has a multiplicative effect on the depth of the forward search tree of actions. It is interesting to further study how larger-scale problem instances in term of the number of objects can be solved efficiently. The current work motivates towards these intriguing avenues of future research.

### References

1. Akbari, A., Lagriffoul, F., Rosell, J.: Combined heuristic task and motion planning for bi-manual robots. *Autonomous Robots* pp. 1–16 (2018)
2. van den Berg, J., Snoeyink, J., Lin, M., Manocha, D.: Centralized Path Planning for Multiple Robots: Optimal Decoupling into Sequential Plans. In: *RSS* (2009)
3. Cohen, B., Phillips, M., Likhachev, M.: Planning single-arm manipulations with n-arm robots. In: *SoCS* (2015)
4. Dantam, N.T., Kingston, Z.K., Chaudhuri, S., Kavraki, L.E.: Incremental task and motion planning: A constraint-based approach. In: *RSS* (2016)
5. Dobson, A., Bekris, K.E.: Planning representations and algorithms for prehensile multi-arm manipulation. In: *IROS*, pp. 6381–6386. *IEEE* (2015)
6. Dobson, A., Solovey, K., Shome, R., Halperin, D., Bekris, K.E.: Scalable Asymptotically-Optimal Multi-Robot Motion Planning. In: *MRS* (2017)
7. Garrett, C.R., Lozano-Perez, T., Kaelbling, L.P.: Ffrob: Leveraging symbolic planning for efficient task and motion planning. *IJRR* **37**(1), 104–136 (2018)
8. Ghrist, R., O’Kane, J.M., LaValle, S.M.: Computing Pareto Optimal Coordinations on Roadmaps. *IJRR* **24**(11) (2005)
9. Halperin, D., Latombe, J.C., Wilson, R.H.: A General Framework for Assembly Planning: the Motion Space Approach. *Algorithmica* **26**(3-4) (2000)
10. Han, S.D., Stiffler, N.M., Bekris, K.E., Yu, J.: Efficient, high-quality stack rearrangement. *IEEE RAL* **3**(3), 1608–1615 (2018)
11. Han, S.D., Stiffler, N.M., Krontiris, A., Bekris, K., Yu, J.: Complexity results and fast methods for optimal tabletop rearrangement with overhand grasps. *arXiv:1711.07369* (2017)
12. Harada, K., Tsuji, T., Laumond, J.P.: A Manipulation Motion Planner for Dual-Arm Industrial Manipulators. In: *ICRA* (2014)

13. Hauser, K., Ng-Thow-Hing, V.: Randomized Multi-Modal Motion Planning for a Humanoid Robot Manipulation Task. *IJRR* **30**(6) (2011)
14. Havur, G., Ozbilgin, G., Erdem, E., Patoglu, V.: Geometric rearrangement of multiple movable objects on cluttered surfaces: A hybrid reasoning approach. In: *ICRA* (2014)
15. Janson, L., Schmerling, E., Clark, A., Pavone, M.: Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions. *IJRR* (2015)
16. Kaelbling, L.P., Lozano-Pérez, T.: Hierarchical Task and Motion Planning in the Now. In: *ICRA* (2011)
17. Karaman, S., Frazzoli, E.: Sampling-based Algorithms for Optimal Motion Planning. *IJRR* **30**(7) (2011)
18. Kavraki, L.E., Svestka, P., Latombe, J.C., Overmars, M.: Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces. *IEEE Transactions on Robotics and Automation* **12**(4) (1996)
19. Krontiris, A., Bekris, K.E.: Dealing with difficult instances of object rearrangement. In: *RSS* (2015)
20. Krontiris, A., Bekris, K.E.: Efficiently solving general rearrangement tasks: A fast extension primitive for an incremental sampling-based planner. In: *ICRA* (2016)
21. LaValle, S.M., Kuffner, J.J.: Randomized Kinodynamic Planning. *IJRR* **20** (2001)
22. Ota, J.: Rearrangement of multiple movable objects-integration of global and local planning methodology. In: *ICRA*, vol. 2 (2004)
23. Saribatur, Z.G., Patoglu, V., Erdem, E.: Finding optimal feasible global plans for multiple teams of heterogeneous robots using hybrid reasoning: an application to cognitive factories. *Autonomous Robots* **43**(1), 213–238 (2019)
24. Schmitt, P.S., Neubauer, W., Feiten, W., Wurm, K.M., Wichert, G.V., Burgard, W.: Optimal, sampling-based manipulation planning. In: *ICRA*. IEEE (2017)
25. Shome, R., Bekris, K.E.: Anytime multi-arm task and motion planning for pick-and-place of individual objects via handoffs. In: *MRS*, pp. 37–43. IEEE (2019)
26. Shome, R., Nakhimovich, D., Bekris, K.E.: Pushing the boundaries of asymptotic optimality in integrated task and motion planning (2019)
27. Shome, R., Solovey, K., Dobson, A., Halperin, D., Bekris, K.E.: drt\*: Scalable and informed asymptotically-optimal multi-robot motion planning. *Autonomous Robots* pp. 1–25 (2019)
28. Shome, R., Solovey, K., Yu, J., Halperin, D., Bekris, K.E.: Fast, high-quality dual-arm rearrangement in synchronous, monotone tabletop setups. In: *WAFR* (2018)
29. Siméon, T., Laumond, J.P., Cortés, J., Sahbani, A.: Manipulation Planning with Probabilistic Roadmaps. *IJRR* **23**(8) (2004)
30. Solovey, K., Salzman, O., Halperin, D.: Finding a needle in an exponential haystack: Discrete RRT for exploration of implicit roadmaps in multi-robot motion planning. *IJRR* **35**(5) (2016)
31. Stilman, M., Schamburek, J.U., Kuffner, J., Asfour, T.: Manipulation planning among movable obstacles. In: *ICRA* (2007)
32. Surynek, P., Felner, A., Stern, R., Boyarski, E.: Efficient sat approach to multi-agent path finding under the sum of costs objective. In: *Proceedings of the Twenty-second European Conference on Artificial Intelligence*, pp. 810–818. IOS Press (2016)
33. Surynek, P., Kumar, T.S., Koenig, S.: Multi-agent path finding with capacity constraints. In: *International Conference of the Italian Association for Artificial Intelligence* (2019)
34. Toussaint, M.: Logic-geometric programming: An optimization-based approach to combined task and motion planning. In: *International Joint Conference on Artificial Intelligence* (2015)
35. Vega-Brown, W., Roy, N.: Asymptotically optimal planning under piecewise-analytic constraints. In: *WAFR* (2016)
36. Wagner, G., Choset, H.: Subdimensional Expansion for Multirobot Path Planning. *Artificial Intelligence Journal* **219** (2015)
37. Yu, J., LaValle, S.M.: Optimal multirobot path planning on graphs: Complete algorithms and effective heuristics. *IEEE Transactions on Robotics* **32**(5), 1163–1177 (2016)

## Appendix

### Sketch of Properties

This section provides a sketch of the theoretical properties of the method proposed in the current work. At a high-level, the proposed approach uses an underlying framework for integrated task and motion planning based on previous work [25,35,13]. This framework retains its completeness guarantees for the proposed formulation. A contribution of the current work is to address instances of multi-arm multi-object task planning problems where a heuristic can be computed efficiently and provides useful guidance in the search process. The aforementioned heuristic is obtained by casting the object rearrangement problem to an analogous instance of a multi-agent path finding problem on a graph. Previous work [37] has studied this problem to demonstrate its correspondence to a multi-commodity network flow problem with its own inherent problem-class complexity.

**Properties of integrated task and motion planning** The underlying task and motion planning framework outlined in the Algorithm SMART is closely based on previously described integrated frameworks [25,35,13], which:

- samples *transitions* - e.g, object picks, handoffs, and placements;
- builds a forward search tree over possible combinations of *transitions* - object grasps, handoffs, and placements;
- and constructs a roadmap to connect each pair of consecutive transitions.

Such pipelines has been shown to be probabilistically complete [13], and under certain assumptions regarding the transitions and underlying roadmaps, also asymptotically optimal [35]. Specifically, more recent analysis [26] studies the specific conditions for arguing asymptotic optimality for such task and motion planning algorithms.

One of the modifications when applying this principle to multi-arm problems is the decomposition of the multi-arm roadmaps into constituent roadmaps of each arm, and searching over their *tensor product* [27]. This operation has also been proven to maintain the completeness and optimality properties of the underlying constituent roadmaps.

Algorithm SMART satisfies the requirements for asymptotic optimality by sampling additional transitions for a fraction of the iterations. The *select* subroutine gives every adjacent transition an opportunity to be selected in the task planning search tree, while goal biasing with the contributed heuristic. The motions between transitions operate over the *tensor* structure defined over all the arms.

**Properties of the computed heuristic** The heuristic utilized to bias exploitation of *actions* or *transition sequences* is derived from solving a simpler variant of the problem that involves only the objects, and considers picks, handoffs and placement actions. The current work shows that this can be cast as a multi-agent path finding problem by treating the objects as agents, and their motions over a discrete graph structure defined in the current work as the capacity constrained object-centric mode graph. The current work also proves that this heuristic is action-optimal, and motion planning for the arms

on top of these actions cannot decrease the number of actions involved in the solution. This renders the MAPF heuristic *admissible*.

In terms of the properties of the corresponding MAPF problem, previous work [37] has provided an ILP formulation that achieves *complete* and efficient solutions to the NP-Hard problem instance. The current work adds capacity constraints to the formulation to make the solutions amenable to the task planning domain. It is straightforward to expand upon the original analysis for unit capacities at the vertices to argue similar complexity results. Other lines of work [33] have studied capacity constraints from the point of view of SAT solvers and arrived at similar arguments. These earlier efforts guided the solution proposed in the current paper, while adhering to a linear programming framework [37].