# Toward Efficient Task Planning for Dual-Arm Tabletop Object Rearrangement

Kai Gao      Jingjin Yu

*Abstract*— We investigate the problem of coordinating two robot arms to solve non-monotone tabletop multi-object re-arrangement tasks. In a non-monotone rearrangement task, complex object-object dependencies exist that require moving some objects multiple times to solve an instance. In working with two arms in a large workspace, some objects must be handed off between the robots, which further complicates the planning process. For the challenging dual-arm tabletop rearrangement problem, we develop effective task planning algorithms for scheduling the pick-n-place sequence that can be properly distributed between the two arms. We show that, even without using a sophisticated motion planner, our method achieves significant time savings in comparison to greedy approaches and naive parallelization of single-robot plans.
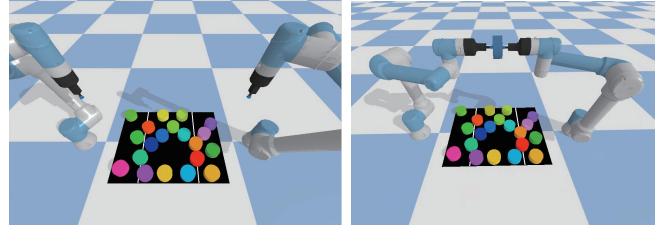
Fig. 1: [Left] PyBullet setup for the Cooperative Multi-Robot Rearrangement problem, where only a portion of the environment (the region between two white lines) is reachable by both arms. [Right] Handoff operation at a pre-computed pose above the environment.

## I. INTRODUCTION

In solving multi-object rearrangement problems, employing multiple arms is a straightforward way to expand the workspace [1] and increase overall system throughput [2]. Toward enabling effective dual-arm (and multi-arm) coordination, a difficult challenge at present, we investigate the *cooperative dual-arm rearrangement* (CDAR) problem, where two robot arms' workspaces partially overlap and each robot is responsible for a portion of the workspace (Fig. 1[Left]). In CDAR, robot-robot collaboration must be considered for realizing higher throughput. Certain settings benefit from having more robots, e.g., one arm may hold an object while other arms make the goal pose of the object available. This will lead to efficiency gain as compared with using a single robot. There are also added challenges, however, e.g., some objects must be handed off from one robot to another which requires careful synchronization.

Rearrangement using a single robot is already challenging to optimally solve due to complex object-object dependencies [3]. CDAR is more involved with an additional arm coordination element. To tackle CDAR and minimize task completion makespan, we adopt a lazy buffer allocation approach [4] and solve a rearrangement task in two phases. In the first phase, we compute a sequence or schedule of "primitive rearrangement actions" without carefully checking feasibility using heuristic search algorithms. In particular, it may be infeasible to relocate certain object as planned due to insufficient free space in the workspace. In the second phase, based on the computed schedule, we then allocate intermediate object poses while carefully consider potential object-object collisions. The heuristic search algorithms are guided by object dependency graphs [3], [5], which is pre-constructed based on the inherent combinatorial constraints induced by start and goal object arrangements. With the dependency graph, there is no additional collision checking needed during the scheduling of primitive actions.

This study brings forth several contributions. Our method for solving CDAR, employing lazy or delayed verification, can efficiently compute high-quality solutions for non-monotone instances in a cluttered environment. As a key component of our algorithmic solution, we propose dependency graph guided heuristic search procedure for coordinating robot-robot object handoffs and individual pick-n-places that supports multiple makespan evaluation metrics. Extensive experiments are conducted to evaluate the performance of proposed algorithms and the quality of CDAR plans in a PyBullet environment with different overlap ratio of robot reachable regions, which, in addition to demonstrating the effectiveness of our method, provides insights for dual-arm manipulation system design.

**Paper Organization**. The rest of the paper is organized as follows. We provide an overview of related literature in Sec. II. In Sec. III, we formally define CDAR problems, discuss the inherent object dependencies and temporary displacements (buffers). Then we describe our proposed heuristic search algorithms guided by the pre-computed dependency graph in Sec. IV. The algorithms efficiently compute task schedules minimizing makespan under two different metrics. In Sec V, we outline the lazy buffer allocation approach. Evaluation follows in Sec. VI. We conclude with Sec. VII.

## II. RELATED WORK

**Multi-Object Rearrangement:** Single arm object rearrangement lies within the broader area of Task and Motion Planning (TAMP). Typical problems in this domain [4], [6]– [9] involve arranging multiple objects to assume specific goal

poses. Certain variations, however, such as NAMO (Navigation among Movable Obstacles) [10], [11], and retrieval problems [12]–[15], focus on clearing out a path for a target object or robot. During the process, obstacles that need to be displaced are identified. Rearrangement may be approached either via simple but inaccurate non-prehensile actions, e.g., pushes [6], [14], [16], [17], or more purposeful prehensile actions, such as grasps [9], [18], [19].

Focusing on inherent combinatorial challenges associated with rearrangement tasks, some planners assume external space for temporary object storage [3], [8], [12], [20], while others exploit problem linearity to simplify the search [10], [11], [21], [22]. By linking multi-object rearrangement to established graph-based problems, efficient algorithms have been obtained for various tasks and objectives [3], [8], [20]. When external free space is not available, the robot arm needs to allocate collision-free locations for temporary object displacements [4], [19], [23]. In our work, we adopt the idea of "lazy buffer allocation" [4] to the dual-arm scenario.

Multi-robot rearrangement requires additional computation on task assignment and coordination. Ahn et al. [24] coordinate robots by maximizing the number of "turn-taking" moments when one arm is picking an obstacle from the workspace while the other arm is placing the previous obstacle at the external space. Shome et al. [2] assume robot arms pick-n-places simultaneously and only solve monotone rearrangement problems, where each object can move directly to the goal pose. For objects that need to be moved to a pose outside the reachable region of a robot arm, the task can be accomplished by coordinating multiple arms to handoff the objects around [1]. Cooperative pick-n-place, a problem related to multi-arm rearrangement, is well-studied in the area of printed circuit board assembly tasks [25], [26]. These problems do not have ordering constraints on pick-n-place tasks, i.e. all the items can move directly to goal poses in any ordering. Research in these topics tends to consider multiple grippers equipped on a single arm rather than multiple arms, which reduces system flexibility. In our work, we compute dual-arm rearrangement plans for dense non-monotone instances without external storage space.

**Dependency Graph:** Dependencies between objects in rearrangement tasks can be naturally represented as a dependency graph, which was first applied to general multi-body planning problems [5] and then rearrangement [18], [19]. In rearrangement, different choices of grasp poses and paths give rise to multiple dependency graphs for the same problem instance, which limits the scalability in computing a solution via such representations. Prior work [3] has applied full dependency graphs to address TORO (Tabletop Object Rearrangement with Overhand Grasps), showing that the problem embeds a NP-hard Feedback Vertex Set problem [27] and a Traveling Salesperson Problem [28].

More recently, some of the authors [8] examined another optimization objective, running buffers, which is the size of the external space needed for temporary object displacements in the rearrangement task, and also examined an unlabeled setting, where goal poses of objects are interchangeable.

Similar graph structures are also used in other robotics problems, such as packing problems [29]. Deep neural networks have been also applied to detect the embedded dependency graph of objects in a cluttered environment to determine the ordering of object retrieval [13].

## III. PRELIMINARIES

### A. Problem Statement

Consider a 2D bounded workspace $\mathcal{W} \subset \mathbb{R}^2$ containing a set of $n$ cylindrical objects $\mathcal{O} = \{o_1, ..., o_n\}$. An arrangement $\mathcal{A}$ of these objects is a set of poses $\{p_1, ..., p_n\}$ with each pose $p_i = (x_i, y_i) \in \mathcal{W}$. $\mathcal{A}$ is *feasible* if the footprints of objects are inside $\mathcal{W}$ and pair-wise disjoint. Outside the workspace, two robot arms $\mathcal{R} = \{r_1, r_2\}$ are tasked to manipulate objects from a feasible start arrangement $\mathcal{A}_s$ to another feasible goal arrangement $\mathcal{A}_g$ (e.g., Fig. 1[Left]). Each robot arm $r_i$ has a connected reachable region $\mathcal{S}(r_i) \subseteq \mathcal{W}$. Robot $r_i$ can only manipulate within $\mathcal{S}(r_i)$.

It is assumed that $\mathcal{S}(r_1) \cup \mathcal{S}(r_2) = \mathcal{W}$. The overlap ratio $\rho$ is defined as $\rho = \dfrac{|\mathcal{S}(r_1) \cap \mathcal{S}(r_2)|}{|\mathcal{W}|}$, which is the proportion of the environment that can be reached by both robots. In Fig. 2[Left], we show a workspace with $\rho = 0.3$. Robot arm $r_1$ can reach objects whose centers are on the left of the red dashed line, which include all the objects at start poses($O_1, O_2, O_3$) and $o_1$, $o_2$ at goal poses($G_1, G_2$). Robot arm $r_2$ can reach objects whose centers are on the right of the blue dashed line, which include all the objects at goal poses($G_1, G_2, G_3$) and $o_1$, $o_2$ at start poses($O_1, O_2$). Objects with centers between the dashed lines can be reached by both.
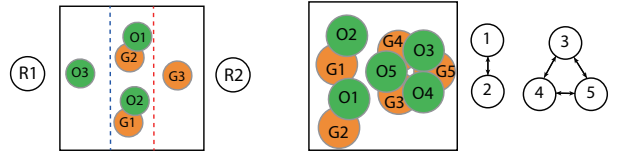


Fig. 2: [Left] A working example of CDAR instance with $\rho = 0.3$. For each object $o_i$, the start and goal poses are represented by green disc $O_i$ and orange disc $G_i$ respectively. [Right] A CDRF instance and its corresponding dependency graph.

We consider two manipulation primitives: (overhand) *pick-n-place* and *handoff*. In each pick-n-place, an arm moves above an object $o_j$, grasps it from $\mathcal{W}$, transfers it atop the workspace, and places it at a collision-free pose inside $\mathcal{W}$. We allow a *handoff* operation when an object needs to cross between $\mathcal{S}(r_1)\backslash\mathcal{S}(r_2)$ and $\mathcal{S}(r_2)\backslash\mathcal{S}(r_1)$. In each handoff, an arm, say $r_1$, grasps an object $o_j$ from $\mathcal{W}$, passes the object to the other arm $r_2$ at the predefined handoff pose above the environment. $r_2$ receives $o_j$ and places it at a collision-free pose in $\mathcal{W}$. A handoff is shown in Fig. 1[Right].

As for the motion of each arm $r_i$, $\mathcal{C}_{free}^i$ is a set of collision-free configurations for $r_i$ without considering collisions with workspace objects or the other robot arm. Throughout the rearrangement, each arm has $n + 1$ motion modes: transit mode $TS$ (moving on its own) and transfer

modes $TF_j$ (transferring the object $o_j$), $\forall j = 1, ..., n$. Therefore, the state space $\Gamma_i$ of an arm $r_i$ is the Cartesian product $\mathcal{C}^i_{free} \times M$, where $M$ is the motion mode set. A path of a robot arm $r_i$ is defined as $\pi_i : [0, T] \to \Gamma_i$ with $\pi_i[0] = \pi_i[T] = (q_i^*, TS)$, where $T$ is the makespan of the rearrangement plan and $q_i^*$ is the rest pose of arm $r_i$.

The rearrangement problem we study seeks a path set $\Pi = \{\pi_1, \pi_2\}$, such that robot paths are collision-free: for all $t \in [0, T]$, arm $r_1$ at state $\pi_1[t]$ does not collide with arm $r_2$ at state $\pi_2[t]$. Additionally, the solution quality is evaluated by the makespan under two levels of fidelity:

1) (MC) manipulation cost based makespan;
2) (FC) estimated execution time based makespan.

In MC (manipulation cost) makespan, we assume the manipulation time dominates the execution time, which is often the case in today's systems. MC makespan is defined as the number of *steps* for the rearrangement task. In each step, each robot arm can complete an individual pick-n-place, execute a coordinated handoff, or idle.

In FC (full cost) makespan, solution quality is measured by the estimated complete time based on four parameters: maximum horizontal speed $s$ of the end-effectors, the execution time of each pick ($t_g$), place ($t_r$), and handoff ($t_h$). The details of FC metric and its corresponding heuristic search method is presented on the extended version of this paper [30].

Given the setup, the problem studied in this paper can be summarized as:

**Problem 1** (CDAR: Collaborative Dual-Arm Rearrangement). *Given feasible arrangements $\mathcal{A}_s$, $\mathcal{A}_g$, and makespan metric (MC/FC), determine a collision-free path set $\Pi$ for $\mathcal{R}$ moving objects from $\mathcal{A}_s$ to $\mathcal{A}_g$ minimizing the makespan $T$.*

We also investigate two special cases of CDAR. On one hand, when $\rho = 1$, $\mathcal{S}(r_1) = \mathcal{S}(r_2) = \mathcal{W}$, which yields to Collaborative Dual-Arm Rearrangement with Full Overlap (CDRF). In this case, handoff is not needed since each robot can execute pick-n-place operations individually inside the workspace. On the other hand, when $\rho = 0$, $\mathcal{S}(r_1) \bigsqcup \mathcal{S}(r_2) = \mathcal{W}$, which yields to Collaborative Dual-Arm Rearrangement with no Overlap (CDRN).

### B. Dependency Graph and Object Displacement

In multi-object rearrangement, minimizing the number of pick-n-places directly leads to increased throughput. Previous works [3] [8] treat the combinatorial challenge in the single robot rearrangement problem with a dependency graph structure $\mathcal{G}$. $o_i$ *depends on* $o_j$ when $o_i$'s goal pose overlaps with $o_j$'s start pose. In this case, $o_j$ needs to be moved away before $o_i$ is moved to the goal. $\mathcal{G}$ is constructed with vertices representing objects and arcs representing the dependencies.

If $\mathcal{G}$ is acyclic, then the rearrangement problem is *monotone*, where objects can be directly moved to goal poses based on the topological ordering implied by $\mathcal{G}$. On the other hand, if there are cycles in the dependency graph, then the problem is *non-monotone*; some object(s) in the cycles need to be displaced to "break" the cycles. For a single arm, the

displaced object must be placed down at some temporary pose. In a dual-arm system, the extra arm can temporarily hold an object to be displaced. For example, Fig. 2[Right] shows a CDRF instance and its corresponding dependency graph $\mathcal{G}$. In this case, the cycle of $o_1$ and $o_2$ can be broken by letting $r_1$ move $o_1$ and $r_2$ move $o_2$ at the same time to the goal poses. However, the cycle among object $o_3$, $o_4$ and $o_5$ cannot be broken without an extra action moving an object (e.g. $o_3$) to some temporary pose.

The free space for a temporary object displacement is called a *buffer*; the planner must allocate buffers inside the workspace. We apply a lazy buffer allocation [4] which is shown to be efficient in computing high quality rearrangement plans for a single arm. In the framework, we first compute a valid schedule of primitive actions, which only indicates whether the objects are handed off, moved to goal poses, or to buffers. And then based on the primitive plan, we allocate proper buffer locations as needed. The scheduling problem of primitive actions is discussed in Sec. IV, and the buffer allocation process is discussed in Sec. V.

### IV. TASK SCHEDULING FOR CDAR

In this section, we study the scheduling problem of primitive actions in CDAR. A primitive action can be represented by a tuple $(r_i, o_j, v)$ indicating that $r_i$ is tasked to transfer $o_j$ to new status $v$, where $v$ is one kind of the object status in $\{S, G, H, \mathcal{B}(r_1), \mathcal{B}(r_2)\}$. $S$, $G$, $H$, $\mathcal{B}(r_1)$, and $\mathcal{B}(r_2)$ represent that the object is at the start pose, goal pose, handoff pose, a buffer in $\mathcal{S}(r_1)$, and a buffer in $\mathcal{S}(r_2)$ respectively. For each primitive action $a$, the schedule provides the starting time and estimated ending time of $a$. Noting that buffer locations are not determined in the task scheduling process, they are categorized into $\mathcal{B}(r_1)$ and $\mathcal{B}(r_2)$ to indicate the arm that they are reachable by. The buffer allocation is discussed in Sec. V. Since the scheduling problem only depends on the availability of goal poses and reachability of robot arms to the start and goal poses, the constraints can be fully expressed by the dependency graph. We describe a dependency graph guided heuristic search method for CDAR under MC makespan metrics (Sec. IV-A). Its variant for FC makespan is shown in the extended version of this paper [30].

### A. Arrangement Space Heuristic Search for CDAR in MC Makespan (MCHS)

When the makespan is counted in MC metric, we can assume primitive actions are executed simultaneously in the task scheduling process without loss of generality and the makespan is counted as the number of *action steps* in the schedule. Each action at action step $t$ is scheduled to start at time $t$ and end at time $t + 1$. Similar to the single arm rearrangement problem, we search for a plan by maintaining a search tree in the arrangement state space. Each arrangement state in the tree represents an object arrangement, which can be expressed as a mapping $\mathcal{L} : \mathcal{O} \to \{S, G, \mathcal{B}(r_1), \mathcal{B}(r_2)\}$. The arrangement state represents a moment when all objects are stably located in the workspace and both arms are empty.

Arrangements are connected by primitive actions of two arms, which consist of one out of the three possible options: (1) individual pick-n-places: one or both arms move objects to poses that are currently collision-free; (2) coordinated pick-n-places: move an object to the goal pose while the other arm tries to clear away the last obstacle at its goal pose, e.g. swapping poses of $o_1$ and $o_2$ in Fig. 2[Left]; (3) a coordinated handoff for an object when its start and goal poses are not both reachable by either arm. We ignore primitive actions that hold an object for more than one action step due to the unnecessary loss of efficiency.

Fig. 3 shows the flow charts indicating the decision process of generating possible primitive actions in each arrangement state. Fig. 3[Upper] shows the decision process for $r_1$ to pick-n-place $o \in \mathcal{O}$ without cooperation with the other arm. That for $r_2$ is equivalent. Fig. 3[Middle] shows the decision process for $r_1$ to manipulate an object $o \in \mathcal{O}$ when only one object $o'$ is blocking the goal pose of $o$. In this case, two arms cooperate to send $o$ to the goal. That for $r_2$ is equivalent. Fig. 3[Below] shows the decision process when an object $o \in \mathcal{O}$ needs handoff. In the process, the availability of goal poses can be checked in the precomputed dependency graph and the reachability of arms is indicated by the object status in the arrangements. Therefore, with the dependency graph, no additional collision-checking is needed in the task scheduling process of CDAR.

For the working example in Fig. 2[Left], a corresponding rearrangement plan, in the form of a path in the arrangement state space is presented in Fig. 4[Left]. At the start arrangement state $s_1$, $r_1$ and $r_2$ move $o_2$ and $o_1$ respectively to the goal poses, which yields the next arrangement state $s_2$. At $s_2$, two arms coordinate to execute a handoff delivering $o_3$ to the goal and reach the goal arrangement state $s_3$. By concatenating primitive actions from $s_1$ to $s_3$, we obtain a corresponding task schedule for the instance (Fig. 4).

The arrangement state heuristic search based on MC makespan (MCHS) explores the arrangement state space in a best-first manner, always developing the arrangement state $s$ with the lowest $f(s) = g(s) + h(s)$. The $g$ function is number of action steps from the start arrangement state to the current arrangement state. For example, in Fig. 4, $g(s_3) = 2$. For each state $s$, we define the heuristic function $h(s)$ based on the needed cost to move all objects directly to goals.

The details are presented in Algo.1. In CDAR, costs can be either exclusive cost of $r_1$, or exclusive cost of $r_2$, or shared cost of $r_1$ and $r_2$. For example, on one hand, a pick-n-place in region $\mathcal{S}(r_1)\backslash\mathcal{S}(r_2)$ can only be executed by $r_1$, so it leads to an exclusive cost of $r_1$. On the other hand, a pick-n-place in the overlapping region $\mathcal{S}(r_1)\bigcap\mathcal{S}(r_2)$ can be executed by either arm so the cost can be shared by both arms. Based on this observation, we count the costs cost[$r_1$], cost[$r_2$], and sharedCost in the heuristic and initialize them as 0 (Line 1). We ignore the objects at the goal poses (Line 3). For the remaining objects, the cost category is determined by $r_1$, $r_2$'s reachability to the current pose (buffer or start pose) and goal pose of the object. ArmSet is the set of arms able to reach both the current pose and goal pose (Line 4). If
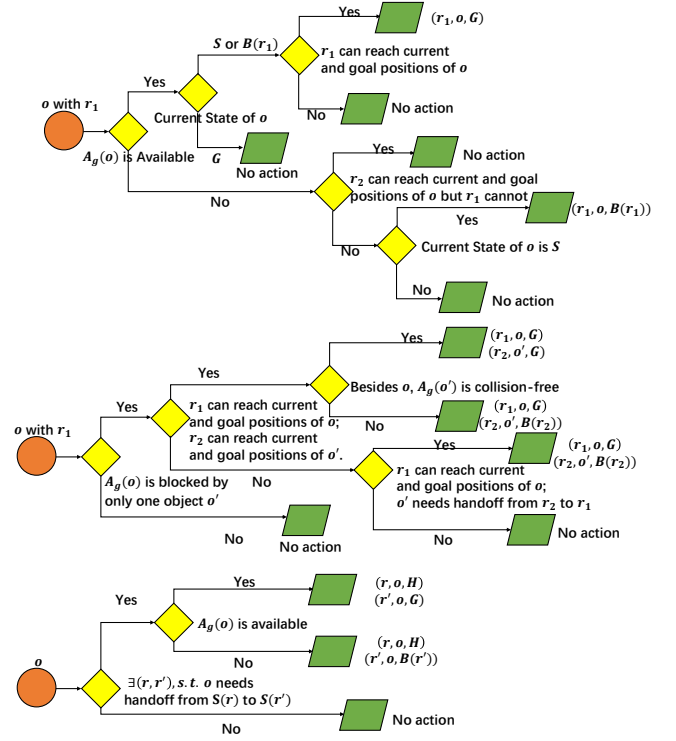


Fig. 3: Decision process of generating possible primitive actions. [Upper] The decision process for $r_1$ to manipulate $o \in \mathcal{O}$ without cooperation with the other arm. That for $r_2$ is equivalent. [Middle] The decision process for $r_1$ to manipulate an object $o \in \mathcal{O}$ when only one object $o'$ is blocking the goal pose of $o$. That for $r_2$ is equivalent. [Bottom] The decision process when an object $o \in \mathcal{O}$ needs handoff.
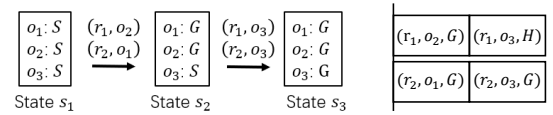


Fig. 4: [Left] A path on the arrangement state space for the instance in Fig. 2[Left] under MC metric. [Right] The corresponding schedule of the example instance in Fig. 2[Left].

both arms are in armSet (Line 5), then the cost of moving $o_i$ to the goal will be added to sharedCost, which means that it is a pick-n-place in the overlapping region. If only one arm is in armSet (Line 6-7), then the cost of moving $o_i$ to the goal will be added to the cost of the specific arm, which means that this action happens out of reach of the other arm. If neither arm is in armSet (Line 8-10), then the object needs a handoff to the destination. In this case, both arms need to take one action for the handoff operation. Finally, on one hand, if the difference of the exclusive costs is smaller than sharedCost (Line 11), then $h(s)$ is half of the total cost (Line 12). In other words, the cost of two arms may be balanced by splitting the sharedCost. On the other hand, if the difference of the exclusive costs is too large (Line 13), then the unoccupied arm bears all the shared cost, and $h(s)$ equals the cost of the busy arm.

Since $cost[r_1]$ and $cost[r_2]$ provide lower bounds of necessary manipulations for each arm, $h(s)$ is a lower bound of the

**Algorithm 1:** Manipulation Cost Search Heuristic

---

**Input** : $s$: current state
**Output:** h: heuristic value of $s$

1   cost[$r_1$], cost[$r_2$], sharedCost $\leftarrow 0, 0, 0$
2   **for** $o_i \in \mathcal{O}$ **do**
3     **if** $\mathcal{L}(o_i)$ *is* $G$ **then** continue;
4     armSet $\leftarrow$ ReachableArms($s, o_i, \{\mathcal{L}(o_i), G\}$)
5     **if** *armSet is* $\{r_1, r_2\}$ **then** sharedCost++;
6     **else if** *armSet is* $\{r_1\}$ **then** cost[$r_1$]++;
7     **else if** *armSet is* $\{r_2\}$ **then** cost[$r_2$]++;
8     **else**
9       cost[$r_1$] ++;
10      cost[$r_2$] ++;
11 **if** $\|cost[r_1]\text{-}cost[r_2]\| \leq sharedCost$ **then**
12 **return** (cost[$r_1$]+cost[$r_2$]+sharedCost)/2 ;
13 **else return** max(cost[$r_1$], cost[$r_2$]) ;
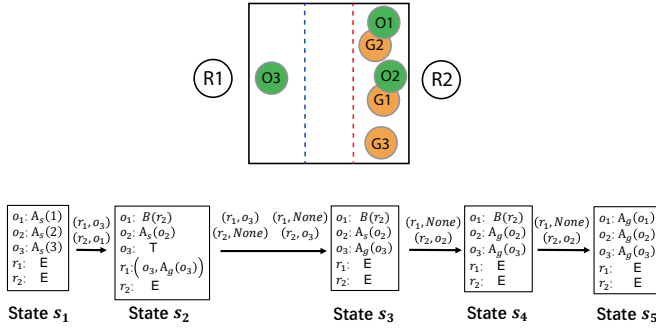
---



Fig. 5: [Top] A CDAR instance that needs a buffer. [Bottom] An rearrangement plan for the example.

manipulation cost from the state $s$ to the goal arrangement. Therefore, we have:

**Proposition IV.1.** *The manipulation cost based heuristic $h(s)$ is consistent and admissible.*

## V. BUFFER ALLOCATION

In this section, we allocate feasible buffer locations based on the primitive plan computed in Sec. IV following the idea of lazy buffer allocation [4]. Given the primitive plan, we know not only the specific timespans that objects stay in buffers, but also the staying poses of the other objects in the timespans, which are treated as fixed obstacles that the buffer location needs to avoid. Besides the obstacles, the buffer location of an object moved to $\mathcal{B}(r)$ in the primitive plan is restricted to be inside $\mathcal{S}(r)$, the reachable region of $r$.

We illustrate the buffer allocation process with a CDAR instance in Fig. 5[Top], which is a variant of the example in Fig. 2[Left]. When $o_1$ and $o_2$ are in $\mathcal{S}_2 \setminus \mathcal{S}_1$, at least one of them needs to be moved to buffer locations to break the cycle in the dependency graph. A rearrangement plan is shown in Fig. 5[Bottom]. In this plan, $o_1$ is moved to a buffer when $o_2$ is at $\mathcal{A}_s(o_2)$, and $o_3$ waits for a handoff. And $o_1$ leaves the buffer when $o_2$ and $o_3$ are both at goal positions. Therefore, the buffer location of $o_1$ needs to be

collision-free with $o_2$ at $\mathcal{A}_s(o_2)$, $\mathcal{A}_g(o_2)$ and $o_3$ at $\mathcal{A}_g(o_3)$. Additionally, it is restricted in $\mathcal{S}(r_2)$. When multiple objects stay in buffers at the same time, the buffer locations need to be allocated disjointedly. In this paper, we search for feasible buffer locations by sampling candidates in the reachable region and check collisions with the fixed obstacles indicated by the primitive plan. The same as TRLB [4], when buffer allocation fails, we accept partial plans and conduct a bi-directional search in the arrangement state to connect the start and goal arrangement.

## VI. EXPERIMENTAL STUDIES

To evaluate the effectiveness of our dual-arm task planning algorithms, we integrate them with a simple priority-based motion planner. Essentially, in the presence of potential arm-arm conflict in the workspace, the motion planner will give the arm that is moving first priority while the second arm yields. The yielding arm will either take a detour to the target pose or go back to the rest pose and wait for the execution of the other arm. We make this choice, instead of using more sophisticated sampling-based asymptotically optimal planners, to highlight the benefit of our task planner.

The proposed algorithms are implemented in Python; we choose objects to be uniform cylinders. In each instance, robot arms move objects from a randomly sampled start arrangement to an ordered goal arrangement (see, e.g., Fig. 6). Besides different overlap ratios $\rho$, we test our algorithms in environments with different density levels $D$, defined as the proportion of the tabletop surface occupied by objects, i.e., $D := (\Sigma_{o_i \in \mathcal{O}} S_{o_i})/S_{\mathcal{W}}$, where $S_{o_i}$ is the base area of $o_i$ and $S_{\mathcal{W}}$ is the area of $\mathcal{W}$. The computed rearrangement plans are executed in PyBullet with UR-5e robot arms.

For the FC makespan, we show the performance of an interval state space heuristic search algorithm (FCHS) [30]. In experiments, we let the execution time of a pick $t_g$, a place $t_r$, and a handoff $t_h$ equal to the time spent for an arm traveling across the diameter of the workspace $t_d$, i.e. $t_g = t_r = t_h = t_d$. This setting mimics general manipulation scenarios in the industry, where a successful pick-n-place relies on accurate pose estimation of the target object, a careful picking to firmly hold the object, and a careful placing to stabilize the object at the desired pose. The experiments are executed on an Intel® Xeon® CPU at 3.00GHz. Each data point is the average of 20 test cases except for unfinished trials, if any, given a time limit of 300 seconds for each test case.
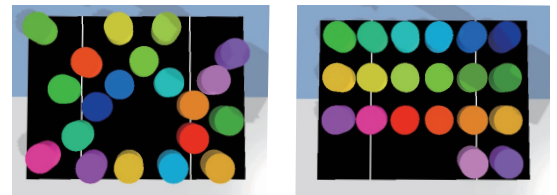


Fig. 6: A 20-cylinder example of rearrangement instance with $D = 0.4$ and $\rho = 0.5$, moving object from a randomly sampled start arrangement [Left], to an organized goal arrangement [Right].

**Manipulation versus Full Cost as Proxies.** In Fig.7, we first compare optimal task schedules in both MC and FC makespans in instances with $\rho = 0.5$, $D = 0.3$. While FCHS shows slight advantage in actual execution time, it incurs significantly more computation time and is more prone to planning failure. The results suggest that MC is more suitable as a proxy for optimizing CDAR planning process. We also compare the execution time of MC-optimal and FC-optimal plans in instances under different density levels and overlap ratios and observe similar outcome. Based on the observation, for later experiments, MCHS is used exclusively.
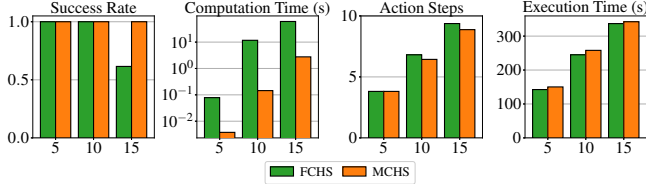


Fig. 7: Performance of FCHS and MCHS in instances with $\rho = 0.5$, $D = 0.3$. The x-axis represents the number of objects.

**Comparison with Baselines.** We compare the proposed algorithms with two baseline methods:

1) *Single-MCHS-Split:* A schedule of primitive actions for a single arm is first computed, minimizing the MC makespan, i.e. the total number of actions. Then, the tasks are assigned to the two arms as evenly as possible. A handoff is coordinated if the manipulating object is moving between $\mathcal{S}(r_1)$ and $\mathcal{S}(r_2)$.

2) *Greedy*: Each arm prioritizes the manipulations moving objects from buffers to goals. When there is no such manipulation available, the arm will choose the object closest to the current end-effector position.

In Fig. 8, we compare MCHS to the baseline algorithms in the environments with $\rho = 0.5$ and different density levels $D$. Comparing with Single-MCHS-Split, MCHS saves up to $10\%$ execution time, which is fairly significant for logistic applications. We note that Single-MCHS-Split, guided by dependency graph, already performs quite well in dense environments. The execution time gap comes from failures in coordinating Single-MCHS-Split plans, e.g. an arm idles for a few action steps waiting for a handoff or an arm holds an object for a few action steps waiting for obstacle clearance. Comparing with the greedy method, MCHS saves more execution time as $D$ increases. Specifically, in 20-cylinder instances with $D = 0.4$, the execution time of MCHS plans is $35\%$ shorter than greedy plans. Without a long-term plan in mind, greedy planning incurs more temporary object displacements.

In instances with $D = 0.3$, we also evaluate the tendency of the makespan and execution time as $\rho$ varies. The conflict proportion shows the proportion of execution time for an arm to yield to the other arm during the execution. The results are shown in Fig. 9. As $\rho$ increases, the shared area of robot arms expands. On one hand, the number of objects that need handoffs decreases. So does the number of action steps in
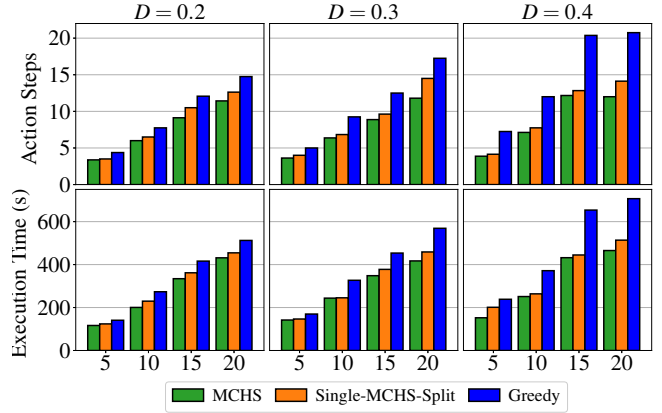


Fig. 8: Algorithm comparison in environments with different density levels $D$. The x-axis represents the number of objects.

the schedule. On the other hand, we see an increase of path conflicts. In instances with larger $\rho$, robot arms spend more time on yielding. Based on the two factors, there is a shallow "U" shape in execution time as $\rho$ increases.



Fig. 9: Evaluation of instances with different $\rho$. The x-axis represents the number of objects.

**Fully Overlapping Workspaces.** MCHS can also compute CDAR plans with full overlap (CDRF), i.e., $\rho = 1.0$. As shown in Fig. 10, due to the reachability limit of UR-5e, the workspace is a bounded square with sides of length $0.6m$. In this case, each arm can reach every corner of the workspace but its manipulation is likely to be blocked by the other arm. We compare dual-arm MCHS plans to the single-arm MCHS plans, where $r_1$ take responsibility of all rearrangement tasks and the number of total actions is minimized. The results (Fig. 11) indicate that each arm in the dual-arm system spends around $18\%$ of execution time yielding or making detours due to the blockage of the other arm. Therefore, even though MCHS saves $50\%$ action steps, the execution of the plans is only around $10\%$ faster than that of the single-arm rearrangement plans. However, the efficiency gain shown in action steps also suggests that the dual-arm system has the potential to save up to half of the execution time with specially designed arms for CDRF problems.
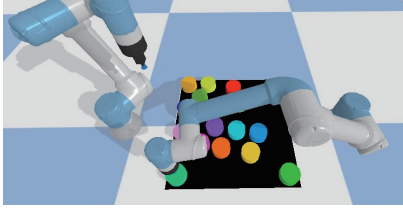
Fig. 10: An instance of CDAR with full overlap (CDRF), where each arm can reach every corner of the workspace but its manipulation is likely to be blocked by the other arm.
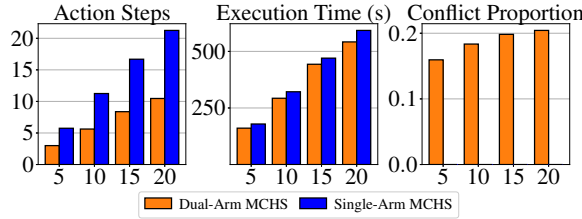


Fig. 11: Algorithm performance in CDAR instances with $\rho = 1.0$. The x-axis represents the number of objects.

## VII. Conclusion

In this work, we investigate the cooperative dual-arm rearrangement (CDAR) problem in a partially shared workspace. We employ a lazy strategy for buffer allocation in the dual-arm rearrangement system. To coordinate manipulations of the arms, we develop dependency graph guided heuristic search algorithms computing optimal primitive task schedules under two makespan evaluation metrics. The effectiveness of the proposed procedure is demonstrated with extensive simulation studies in the PyBullet environment with two UR-5e robot arms. Specifically, plans computed by our algorithms are up to 35% shorter than greedy ones in execution time in dense rearrangement instances. Moreover, we observe a tradeoff between the number of needed hand-offs and the occurrence of path conflicts in CDAR problems as the overlap ratio of the workspace varies.

## References

[1] R. Shome and K. E. Bekris, "Synchronized multi-arm rearrangement guided by mode graphs with capacity constraints," *arXiv preprint arXiv:2005.09127*, 2020.

[2] R. Shome, K. Solovey, J. Yu, K. Bekris, and D. Halperin, "Fast, high-quality two-arm rearrangement in synchronous, monotone tabletop setups," *IEEE Transactions on Automation Science and Engineering*, 2021.

[3] S. D. Han, N. M. Stiffler, A. Krontiris, K. E. Bekris, and J. Yu, "Complexity results and fast methods for optimal tabletop rearrangement with overhand grasps," *The International Journal of Robotics Research*, vol. 37, no. 13-14, pp. 1775–1795, 2018.

[4] K. Gao, D. Lau, B. Huang, K. E. Bekris, and J. Yu, "Fast high-quality tabletop rearrangement in bounded workspace," *arXiv preprint arXiv:2110.12325*, 2021.

[5] J. van Den Berg, J. Snoeyink, M. C. Lin, and D. Manocha, "Centralized path planning for multiple robots: Optimal decoupling into sequential plans." in *Robotics: Science and systems*, vol. 2, no. 2.5, 2009, pp. 2–3.

[6] A. Cosgun, T. Hermans, V. Emeli, and M. Stilman, "Push planning for object placement on cluttered table surfaces," in *2011 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2011, pp. 4627–4632.

[7] 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 *IEEE International Conference on Robotics and Automation*, 2021.

[8] K. Gao, S. W. Feng, and J. Yu, "On running buffer minimization for tabletop rearrangement."

[9] R. Wang, Y. Miao, and K. E. Bekris, "Efficient and high-quality prehensile rearrangement in cluttered and confined spaces," *arXiv preprint arXiv:2110.02814*, 2021.

[10] M. Stilman and J. J. Kuffner, "Navigation among movable obstacles: Real-time reasoning in complex environments," *International Journal of Humanoid Robotics*, vol. 2, no. 04, pp. 479–503, 2005.

[11] M. Stilman and J. Kuffner, "Planning among movable obstacles with artificial constraints," *The International Journal of Robotics Research*, vol. 27, no. 11-12, pp. 1295–1307, 2008.

[12] C. Nam, J. Lee, Y. Cho, J. Lee, D. H. Kim, and C. Kim, "Planning for target retrieval using a robotic manipulator in cluttered and occluded environments," *arXiv preprint arXiv:1907.03956*, 2019.

[13] H. Zhang, Y. Lu, C. Yu, D. Hsu, X. Lan, and N. Zheng, "INVIG-ORATE: Interactive Visual Grounding and Grasping in Clutter," in *Proceedings of Robotics: Science and Systems*, Virtual, July 2021.

[14] E. R. Vieira, D. Nakhimovich, K. Gao, R. Wang, J. Yu, and K. E. Bekris, "Persistent homology for effective non-prehensile manipulation," *arXiv preprint arXiv:2202.02937*, 2022.

[15] B. Huang, T. Guo, A. Boularias, and J. Yu, "Self-supervised monte carlo tree search learning for object retrieval in clutter," *arXiv preprint arXiv:2202.01426*, 2022.

[16] J. E. King, V. Ranganeni, and S. S. Srinivasa, "Unobservable monte carlo planning for nonprehensile rearrangement tasks," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 4681–4688.

[17] B. Huang, S. D. Han, A. Boularias, and J. Yu, "Dipn: Deep interaction prediction network with application to clutter removal," in *IEEE International Conference on Robotics and Automation*, 2021.

[18] A. Krontiris and K. E. Bekris, "Dealing with difficult instances of object rearrangement." in *Robotics: Science and Systems*, vol. 1123, 2015.

[19] ——, "Efficiently solving general rearrangement tasks: A fast extension primitive for an incremental sampling-based planner," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 3924–3931.

[20] S. Bereg and A. Dumitrescu, "The lifting model for reconfiguration," *Discrete & Computational Geometry*, vol. 35, no. 4, pp. 653–669, 2006.

[21] K. Okada, A. Haneda, H. Nakai, M. Inaba, and H. Inoue, "Environment manipulation planner for humanoid robots using task graph that generates action sequence," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 2. IEEE, 2004, pp. 1174–1179.

[22] M. Levihn, J. Scholz, and M. Stilman, "Hierarchical decision theoretic planning for navigation among movable obstacles," in *Algorithmic Foundations of Robotics X*. Springer, 2013, pp. 19–35.

[23] S. H. Cheong, B. Y. Cho, J. Lee, C. Kim, and C. Nam, "Where to relocate?: Object rearrangement inside cluttered and confined environments for robotic manipulation," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 7791–7797.

[24] J. Ahn, C. Kim, and C. Nam, "Coordination of two robotic manipulators for object retrieval in clutter," *arXiv preprint arXiv:2109.15220*, 2021.

[25] S. Li, C. Hu, and F. Tian, "Enhancing optimal feeder assignment of the multi-head surface mounting machine with genetic algorithms," *Applied Soft Computing*, vol. 8, no. 1, pp. 522–529, 2008.

[26] M. Moghaddam and S. Y. Nof, "Parallelism of pick-and-place operations by multi-gripper robotic arms," *Robotics and Computer-Integrated Manufacturing*, vol. 42, pp. 135–146, 2016.

[27] R. M. Karp, "Reducibility among combinatorial problems," in *Complexity of computer computations*. Springer, 1972, pp. 85–103.

[28] C. H. Papadimitriou, "The euclidean travelling salesman problem is np-complete," *Theoretical computer science*, vol. 4, no. 3, pp. 237–244, 1977.

[29] F. Wang and K. Hauser, "Robot packing with known items and nondeterministic arrival order," *IEEE Transactions on Automation Science and Engineering*, 2020.

[30] K. Gao and J. Yu, "Toward efficient task planning for dual-arm tabletop object rearrangement," *arXiv preprint arXiv:2207.08078*, 2022.