High-Performance Dual-Arm Task and Motion Planning for Tabletop Rearrangement

Duo Zhang Junshan Huang Jingjin Yu

***Abstract*— We propose Synchronous Dual-Arm Rearrangement Planner (SDAR), a task and motion planning (TAMP) framework for tabletop rearrangement, where two robot arms equipped with 2-finger grippers must work together in close proximity to rearrange objects whose start and goal configurations are strongly entangled. To tackle such challenges, SDAR tightly knit together its dependency-driven task planner (SDAR-T) and synchronous dual-arm motion planner (SDAR-M), to intelligently sift through a large number of possible task and motion plans. Specifically, SDAR-T applies a simple yet effective strategy to decompose the global object dependency graph induced by the rearrangement task, to produce more optimal dual-arm task plans than solutions derived from optimal task plans for a single arm. Leveraging state-of-the-art GPU SIMD-based motion planning tools, SDAR-M employs a layered motion planning strategy to sift through many task plans for the best synchronous dual-arm motion plan while ensuring high levels of success rate. Comprehensive evaluation demonstrates that SDAR delivers a 100% success rate in solving complex, non-monotone, long-horizon tabletop rearrangement tasks with solution quality far exceeding the previous state-of-the-art. Experiments on two UR-5e arms further confirm SDAR directly and reliably transfers to robot hardware.**

I. Introduction

Task and motion planning (TAMP) [1] represents a fundamental computation challenge in robotics, in which a robot system, e.g., one or more robot arms, must break down a given, potentially long-horizon task into suitable "bite-sized" sub-tasks that can be executed through short-horizon robot motions. TAMP, which integrates high-level reasoning with low-level motion control, stands as a cornerstone in robotics as it holds the promise to empower robots to carry out the full spectrum of daily human tasks that demand physical interaction with the world. Yet, "solving" TAMP has remained elusive because doing so must handle the combinatorial explosion of discrete sub-task partitioning and sequencing, and simultaneously tackle the complexity of generating high-quality motion in high-dimensional environments, both of which are hard computational challenges themselves [2]–[4].

Despite intractability obstacles, TAMP continues to attract significant research attention with remarkable progress being made [5]–[12], which have explored solution schemes ranging from combinatorial search to data-driven, and anywhere in between. With an effective planning framework [5], speedy optimal task planners [13], [14], and recent advances in CPU/GPU-SIMD accelerated motion planning [15], [16], the computation bottleneck for systems with a single robot arm has largely shifted from integrating task and motion planning to complex task/scene understanding.

D. Zhang, J. Huang, and J. Yu are with the Department of Computer Science, Rutgers, the State University of New Jersey, Piscataway, NJ, USA. E-Mails: {duo.zhang, junshan.huang, jingjin.yu} @rutgers.edu.

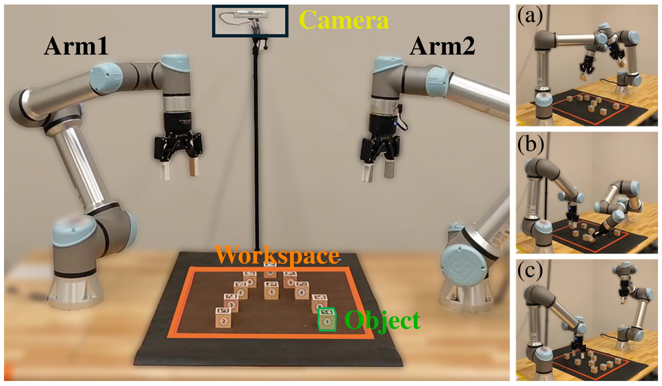

Fig. 1: Illustration of the dual-arm tabletop rearrangement setup examined in this work. (a) Motion planner performing regular pick-and-place, (b) Arms working in close proximity to handle objects that are close to each other, (c) Falling back to single-arm sequential execution when no feasible dual-arm solution can be quickly found.

As we move from a single robot arm to dual-arm systems [17]–[20] operating in close proximity, TAMP remains a key computation bottleneck. This is due to the dramatic increase in complexity from both task planning and motion planning. From the task planning side, the introduction of a second arm doesn't just allow the system to do two things at once, but can also do many new things, i.e., two arms can now collaboratively perform many tasks impossible for a single arm. This also means that the search space for possible task plans grows exponentially. From the motion planning side, the degree-of-freedoms (DoFs) of the system doubles from 6-7 for a single arm to 12+, making motion planning again challenging. While planners like Curobo [16] can compute nice motions for dual-arm systems for certain pre-specified start/goal configurations, as we have found in our project, doing so reliably for random start/goal configurations remains difficult. Integrating task and motion planning further compounds the dual-arm TAMP challenge.

In this paper, toward the goal of near-optimally solving task and motion planning challenges for dual-arm systems in real-time, we examine the long-horizon tabletop rearrangement task where many objects concentrated on a small tabletop must be rearranged using a dual-arm system (Fig. 1). Our framework, *Synchronous Dual-Arm Rearrangement Planner* (SDAR), follows the typical hierarchical algorithmic structure of TAMP solvers, bringing the following key contributions to dual-arm TAMP:

- **Dependency-Driven Dual-Arm Task Planning**. SDAR's task planner (SDAR-T) efficiently resolves task dependencies through a layered approach, generating sub-tasks by "peeling off" tasks with increasing complexity. Unlike typical task planners, SDAR-T outputs multiple high-quality next-step sub-tasks. Taking full advantage of dual-arm systems, SDAR-T reduces the total sub-task counts as compared to optimal single-arm task planners [20].

- **Sampling-Based Motion Generation and Optimization**. Borrowing the tried-and-true practice from sampling-based motion planning, SDAR's motion planner (SDAR-M) examines multiple sub-tasks and for each sub-task, multiple potential grasp poses to select the highest quality (short-term) motion plan. This results in SDAR-M having a much higher success rate than [16], which is a building block of SDAR-M.
- **Robust Failure Recovery**. In rare cases, SDAR-M may fail to plan an optimized trajectory using synchronous dual-arm motions, e.g., due to arm-arm collisions. When this happens, SDAR detects it and engages a multi-stage fallback plan that gradually sacrifices solution quality to boost planning success rates.

Fusing SDAR-T and SDAR-M, SDAR delivers a highly effective, *dual-arm native* solution for the tabletop rearrangement task. Extensive evaluation shows that SDAR can handle a variety of challenging, dense tabletop rearrangement problems with 100% success rates. In contrast, a baseline TAMP planner [20] only achieves a success rate of 85%. Simultaneously, SDAR drastically shortens the task execution time by nearly three times, as compared with the baseline. Moreover, SDAR is highly computationally efficient, taking only about 5 seconds to compute the task and motion for a single (dual-arm) sub-task, making it near real-time.

II. Related Work

Task Planning and Multi-Object Rearrangement. Task planning is a central problem in robotics, defining how symbolic actions are structured to achieve complex goals. Prior work spans diverse domains, from conflict resolution in multi-agent path finding [21] to everyday tasks such as table setting with commonsense knowledge from large language models (LLMs) [22]. Within this scope, object rearrangement has drawn particular attention for both its practical significance and theoretical difficulty. Chang et al. [23] proposed a language-guided MCTS framework for natural-language-driven tabletop rearrangement. Other studies addressed additional complexity, including varying object shapes, weights, or multi-layer stacking [6], [24], [25], which further complicates task planning. From a computational standpoint, minimizing pick-and-place operations [26] or the number of temporarily displaced objects [27] has been proven NP-hard.

Motion Planning. Motion planning originated with 2D path-finding for point or polygonal robots. Early exact methods included cell decomposition [28], roadmap construction [29], and rotation-stacked visibility graphs [30], applicable mainly to $\mathbb{R}^2$ or $SE(2)$. To address higher-dimensional spaces, sampling-based algorithms such as RRT [28], and PRM [31], along with optimal variants [32]–[34], became standard, though they face exponential complexity growth in the number of DoFs. Gradient-based methods like TrajOPT [35], [36] and CHOMP [37] directly optimize trajectories, and recent semi-infinite programming formulations guarantee provably collision-free paths [38], [39]. Yet global optimality is not guaranteed, motivating hybrid approaches such as the graph of convex sets [40] and cuRobo [16], which leverage GPU parallelization for roadmap construction and trajectory optimization.

Multi-arm coordination has gained interest for enabling parallel and flexible task execution [41], [42]. A straightforward extension applies planning methods in the full joint space, though dimensionality becomes prohibitive. Alternatives include hierarchical roadmaps merged into a super-graph [43], conflict-based search (CBS) adaptations [44], [45], and shortcutting techniques to improve trajectory quality [46]. Learning-based methods also show promise: Ha et al. [47] trained reinforcement learning policies from sampling-based demonstrations, achieving improved scalability and faster planning times.

Multi-Arm Task and Motion Planning. Multi-arm task and motion planning (MATAMP) inherits the combinatorial difficulty of task planning and the geometric complexity of motion planning, yet it is crucial for real-world applications. Representative domains include cooperative assembly [17], [48], roof bolting [49], tabletop rearrangement [20], [50], and bi-manual object retrieval from clutter [51], [52]. Classical approaches extend TAMP frameworks to handle inter-arm coordination and collisions: Chen et al. [17] coupled MILP-based task allocation with multi-agent motion planning for assembly, while Zhang et al. [49] introduced a graph-guided MCTS for collaborative manipulation.

Tabletop rearrangement has been a major focus, as it captures core MATAMP challenges. Shome et al. [53] optimized synchronous dual-arm rearrangement under monotone assumptions via MILP, while Gao et al. [20], [50] developed dependency-graph and buffer-based methods for non-monotone cases. Learning approaches are also emerging: attention-based imitation learning improves robustness in dual-arm manipulation [54], and DG-MAP [55] leverages diffusion models to iteratively resolve collisions, scaling to as many as eight arms in simulation.

III. Preliminaries

A. Multi-Object Tabletop Rearrangement

In a tabletop rearrangement problem, we work with a rectangular workspace $\mathcal{W} \subset \mathbb{R}^2$. There are two robot arms r_1, r_2, the end effector of each of which can reach the entirety of $\mathcal{W}$. There are n objects fully residing on $\mathcal{W}$, the tabletop, where object i configuration is specified by $o_i = (x_i, y_i, \theta_i) \in SE(2)$ with $(x_i, y_i) \in \mathcal{W}$.[1] An *arrangement* of the objects is specified as $O = \{o_1, \ldots, o_n\}$. An arrangement O is *feasible* if there is no collision between any pair of objects i and j, $i \neq j$, considering the objects' physical footprint. For example, Fig. 2 (a)(b) show two feasible

[1]An object i may be held by a robot arm r_j, in which case the object will have a *special* configuration in $\{r_1, r_2\}$.

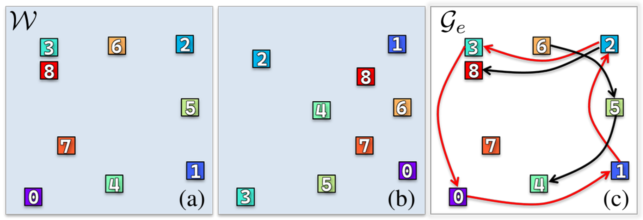

Fig. 2: Illustration of a tabletop rearrangement instance that will be used as a running example. $\mathcal{W}$ is the workspace (a) Start configuration O^s. (b) Goal configuration O^g. (c) The induced dependency graph $\mathcal{G}_e$.

configurations of eight cuboids (top-down view). We define the dual-arm rearrangement problem as follows.

Problem 1 (k-Arm Tabletop Rearrangement)**.** *Given two feasible arrangements $O^s = \{o_1^s, \ldots, o_n^s\}$ and $O^g = \{o_1^g, \ldots, o_n^g\}$, a k-arm tabletop rearrangement problem asks for a sequence of intermediate arrangements $\{O^s = O^1, \ldots, O^m = O^g\}$ such that:*

1) O^i, $1 \leq i \leq n$, is a feasible arrangement.

2) For each consecutive pair of arrangements (O^i, O^{i+1}), $1 \leq i < n$, there exists feasible motions for the robots to transition from O^i to O^{i+1} in which each arm manipulates at most a single object.

In this work, $k = 2$. Fig. 2 illustrates an instance of Problem 1 where Fig. 2(a) corresponds to O^s and Fig. 2(b) corresponds to O^g. This instance will be used a running example for explaing how SDAR work.

We note that Problem 1 is a more challenging version than a similar problem from [20] as a suction-based grasping model is assumed by [20] to grasp cylindrical objects, which is much easier than using 2-finger grippers to grasp non-cylindrical objects from the sides.

B. Dependency Graph

In rearranging objects, a key challenge is the untangling of object dependencies. To effectively accomplish this, we leverage a data structure called the dependency graph (DG) [2], induced by the start and goal configurations of an object rearrangement problem. In a DG $\mathcal{G} = (V, E)$, vertex $v_i \in V$ corresponds to object i, and a directed edge $(v_i \to v_j) \in E$ indicates that object i cannot be placed at its goal pose until object j has been moved away.

Fig. 2(c) provides the DG induced by Fig. 2(a),(b) as the start and goal configurations, respectively. In the DG $\mathcal{G}_e$, object 7 has no dependencies, suggesting it can be directly "solved". Similarly, objects 4 and 8 do not depend on other objects (no outgoing edge) and can be directly moved to their goals. All other objects, on the other hand, have outgoing edges, meaning that they can not be solved without moving some other objects first. In particular, the red edges form a *directed cycle*, meaning that they form a cyclic dependency among themselves. Resolving such cyclic dependencies requires either picking up and holding an object or relocating an object to a temporary location in $\mathcal{W}$.

IV. Synchronous Dual-Arm Rearrangement

Using the example illustrated in Fig. 2, in this section, we elaborate on how our Synchronous Dual-Arm Rearrangement Planner (SDAR) framework functions.

A. Dependency-Driven Dual-Arm Task Planning

1) Structure of the Dependency Graph: A DG is constructed in a straightforward manner. Given two configuration O^s and O^g, geometric conflicts are examined for each o_i^g and o_j^i; an edge $(v_i \to v_j)$ is added if o_i^g and o_j^s overlap spatially. Collecting all these dependency edges then yields a DG for (O^s, O^g).

The task planner of SDAR, SDAR-T, identifies from a DG the following:

- **Independent Objects:** These are vertices with zero out-degree and can be moved immediately, e.g., objects 4, 7, and 8 in Fig. 2.
- **Chains:** Chains are directed paths in a DG that impose strict order with which objects can be moved sequentially, e.g., objects 4, 5, and 6 in Fig. 2.
- **Cycles:** Cycles are strongly connected components that require more coordination to resolve, e.g., objects 0, 1, 2, and 3 in Fig. 2.

In Fig. 2, there are nine objects to be rearranged (labeled 0–8), forming a dependency graph (DG) $\mathcal{G}_e$ that includes one (directed) cycle (objects 0–3), one chain (objects 4–6), and two independent objects (objects 7 and 8).

2) Sub-Task Decomposition and Assignment: Similar to [20], SDAR-T derive possible task plans through a DG-driven analysis, decomposing the rearrangement problem into tractable sub-tasks and generating a task plan that respects all constraints. However, the task planner in [20] first searches for an optimal plan for a *single* robot arm, which does not effectively leverage a dual-arm system's capabilities. SDAR-T, instead, seeks to break down a problem directly using two robot arms, via simultaneous *node removal sequencing* and *node-arm assignment*. As will be demonstrated in the evaluation, compared with [20], SDAR-T generally takes fewer/same number of actions.

Node Removal Sequencing. SDAR-T begins by rearranging all independent objects, which is straightforward. Doing so effectively *deletes* all *orphan nodes* on the DG. For $\mathcal{G}_e$ in Fig. 2, object 7 gets removed. Then, SDAR-T looks at chains, the *leaf nodes* of which can be resolved one by one because they have no dependencies. In $\mathcal{G}_e$, these are 4 and 8 initially. After 4 is moved and deleted from $\mathcal{G}_e$, 5 becomes a leaf node, and so on. After removing all orphan and leaf nodes, the DG is either empty or consists only of cycles. In the case of $\mathcal{G}_e$, only the cycle $0 \to 1 \to 2 \to 3 \to 0$ remains. These remaining cycles are handled last by *cycle breaking*, which may create new orphan and leaf nodes. SDAR-T distinguishes between two cases for cycles. For a 2-object cycle, a direct dual-arm swap is sufficient. For longer cycles, SDAR-T selects two consecutive objects (e.g., 0 an 1 in $\mathcal{G}_e$) such that one is temporarily relocated to a *buffer* because it cannot be directly placed at its goal (if 0 and 1 are selected for $\mathcal{G}_e$, 1 must be placed at a buffer because of the $1 \to 2$ dependency). SDAR-T only marks that a buffer is needed, leaving the motion planner (SDAR-M) to find it.

For $\mathcal{G}_e$, one possible sequence is $7, 8, 4, 5, 6, 0, 1, 3, 2, 1$, where 1 appears twice because it needs to be temporarily relocated. Such sequence is not unique; SDAR-T always works with multiple sequences implicitly.

Node-Arm Assignment. As the DG is being decomposed, removed/updated objects are *assigned* to arms. Essentially, this is done by taking pairs of nodes in the front of node removal sequences and pairing them up. For $\mathcal{G}_e$, the first assignment pair can be $(4, 7)$, $(7, 8)$, $(4, 8)$, and so on, where the order is interchangeable (e.g., $(4, 7) \equiv (7, 4)$, likewise for $(7, 8)$ and $(4, 8)$).

3) Algorithm Sketch: We now outline SDAR-T. Let r_1 and r_2 denote the two robot arms. A task plan is denoted by (i, j), where $i \neq j$: arm r_1 rearranges (s_i, g_i) and arm r_2 rearranges (s_j, g_j). (s_i, g_i) are the *current* start and goal configurations for i, respective, which may be different from (o_i^s, o_i^g). The same goes for j. Each arm maintains the following state information: (1) Gripper state, which may be OPEN or CLOSE, (2) Assigned object ID, and (3) Grasp angle, (4) Execution stage $ToStart$ and $ToGoal$, indicating whether r_1 and r_2 are heading to start poses. A sketch of SDAR-T is outlined in Alg. 1.

Algorithm 1: SDAR-T Task Planner

```
1  Input: Start Configuration $S$, Goal Configuration $G$
2  Output: Task plan for the next step
3  Build the dependency graph DG($S, G$)
4  if Both arms are are $ToGoal$ then
5      Set gripper actions to OPEN
6      Maintain current assignments and angles
7      return task plan
8  end
9  if Both arms are $ToStart$ then
10     taskList ← $\varnothing$
11     if only one object left then
12         Assign one arm to complete it, while the other
             goes back to the retract position
13         return task plan
14     else if multiple independent objects then
15         Append the permutations of independent objects to
             taskList
16     else if a chain exists in DG then
17         Append the terminal pair to taskList
18     else if a cycle exists in DG then
19         Append all adjacent pairs to taskList.
20         Set NeedBuffer flag to true
21     end
22     Set both gripper actions to CLOSE, assign taskList
23     return task plan
24 end
```

The task plan structure encapsulates all relevant execution parameters, including: (1) Potential task list with the form $\{(i^{(1)}, j^{(1)}), (i^{(2)}, j^{(2)}), \ldots\}$, (2) Gripper actions for r_1 and r_2 that can be OPEN or CLOSE, which will be executed after reaching the the start/goal of the arm, (3) Assigned object IDs and grasp angles for both arms, and (4) Buffer needed flag.

B. Sampling-Based Synchronous Motion Generation

SDAR-M, our motion planner, leverages Nvidia tools, assembled within cuRobo [16], to perform GPU-SIMD-based accelerated motion generation and optimization. In this study, motions for the two robot arms are synchronized at the sub-task level to limit the search space that is explored, because SDAR-T already generates many potential sub-tasks for every pair of robot motions and SDAR-M further explores many possible parameters to select the best sub-task.

1) Sub-Task Selection and Instantiation: The general idea behind our motion planner, SDAR-M, is to sample many possible sub-tasks and select the one with the lowest cost. In other words, SDAR-M takes a best-first approach in arm motion generation. The amount of sampling done is determined by the amount of computation budget, the amount of parallelism, and the feasibility of individual samples.

In a nutshell, the "best" sub-task is selected based on inverse kinematics (IK) feasibility. For each sub-task in the current sub-task list as generated by SDAR-T, recall that it is in the form of (s_i, g_i) for arm r_1 and (s_j, g_j) for arm r_2. For generate arm motion, object poses (s_i, g_i) and (s_j, g_j) must be paired with corresponding arm poses. s_i and s_j are already bound to the ending pose of the previous arm motion; SDAR-M needs to sample potential arm poses for g_i and g_j. To do this, multiple 2-finger gripper approaching angles are examined in an expanding manner (shown in Fig. 3). That is, top-down grasps (for a cuboid, there are two of these) will be tried first, after which sideway grasps with increasingly larger approaching angles will be attempted. The process also instantiates the sub-task for the robot arms.

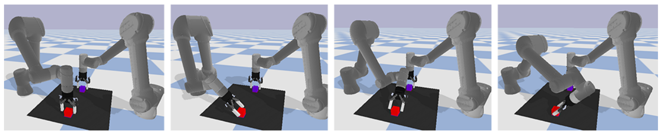
Fig. 3: Illustration of possible sampled grasp poses for grasping the two cuboids. For the arm on the right, the grasp poses are the same as the top-down ones. For the left arm, each is a different pose from a different approaching angle.

When a sub-task indicates that a temporary buffer location is needed, SDAR-M will sample k non-overlapping positions in the workspace as potential buffers. Grasp poses for these will be generated similarly.

For actual IK computation, we employ the SIMD-based batch IK computation capability of [16]. In a single batch, on an Nvidia RTX 4090 with 24GB RAM, it is possible to compute ∼256 IKs. On average, for each sub-task selection, 20 batches are sufficient for sifting through enough pose candidates to select a high-quality sub-task with ∼1.7s to compute all the IKs.

2) Motion Planning with Arm Untangling: After a sub-task is selected and instantiated for the robot arms, motion plans can be generated. SDAR-M uses a two-stage process for completing this process. First, SDAR-M attempts to generate smooth optimized motions from the current arm positions to the target positions (can be start or the goal of the arm depending on the Execution Stages) using the MOTIONGEN module from cuRobo [16]. Despite carefully selecting poses for the robots, MOTIONGEN can fail, in which case SDAR-M falls back to a rule-based planner, ARMUNTANGLING, that follows a set of rules to untangle

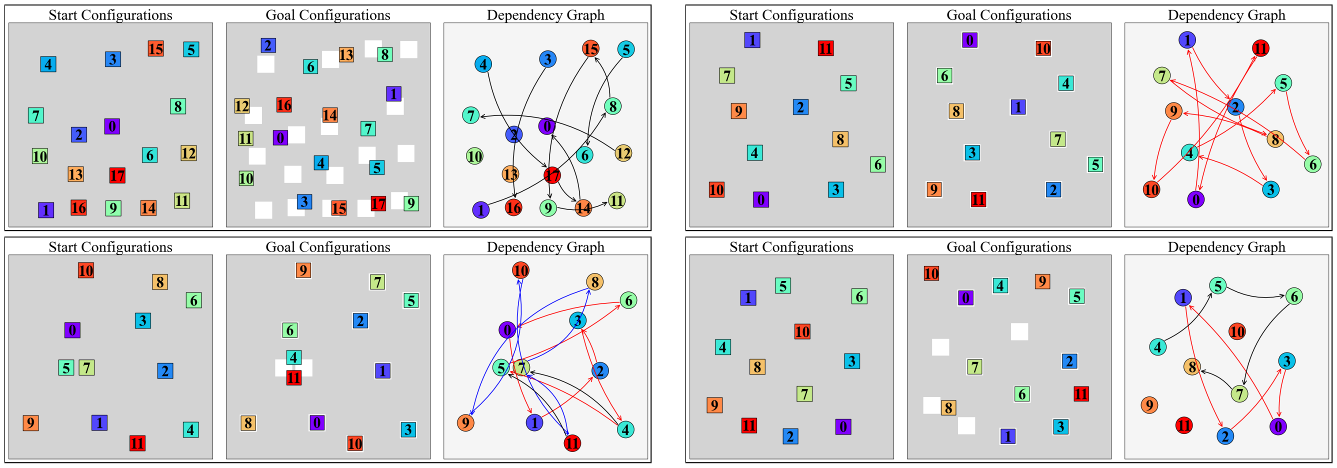
Fig. 4: Examples of start and goal configurations along with their corresponding dependency graphs. In the goal panels, the start configurations are also shown as white squares for reference. The four categories are shown: **(top left)** random configuration, **(top right)** single-cycle configuration, **(bottom left)** double-cycle configuration, and **(bottom right)** mixed configuration containing independent tasks, chains, and cycles. Black arrows denote chain dependencies in the task graph, while red and blue arrows represent different cycles.

Algorithm 2: SDAR-M Motion Planner

```
Input: Potential task list $\mathcal{T}$
Output: Executable motion plan
$(i, j) \leftarrow$ SelectBestTask($\mathcal{T}$)
$(s_i, s_j) \leftarrow$ current arm poses
$(t_i, t_j) \leftarrow$ target poses for objects $i$ and $j$ depending on
  execution stages
$(\alpha_1, \alpha_2) \leftarrow$ selected grasp angles
$P \leftarrow$ cuRobo.Plan($s_i, t_i, s_j, t_j, \alpha_1, \alpha_2$)
if $P$ is valid then
    return $P$
end
$P_1 \leftarrow$ ArmUntangle($s_i, t_i, s_j, t_j, \alpha_i, \alpha_j$)
Get new arm poses $s_i', s_j'$ from $P_1$
$P_2 \leftarrow$ cuRobo.Plan($s_i', g_i, s_j', g_j, \alpha_1, \alpha_2$)
if $P_1$ and $P_2$ are valid then
    return $P_1 + P_2$
end
$P_1^{single} \leftarrow$ cuRobo.Plan($s_i$, retract$_1$, $s_j, t_j, \alpha_1, \alpha_2$)
$P_2^{single} \leftarrow$ cuRobo.Plan(retract$_1$, $t_i, t_j$, retract$_2$, $\alpha_1, \alpha_2$)
if $P_1^{single}$ and $P_2^{single}$ are valid then
    return $P_1^{single} + P_2^{single}$
end
return failure
```

the motions of the two robot arms to increase the motion planning success rates.

In rare cases, ArmUntangling may also fail, in which case SDAR-M falls back to sequential planning and execution. Outline of SDAR-M is outlined in Alg. 2. Our overall method, SDAR, calls SDAR-T and SDAR-M sequentially until an instance is fully resolved.

V. Evaluation

We evaluate SDAR through a series of simulated experiments designed to assess its efficiency, robustness, and scalability across diverse task settings. All algorithms are implemented in `Python` and executed on a workstation equipped with an Intel Core i9-14900K CPU and an NVIDIA RTX 4090 GPU.

Four categories of test cases are generated with distinct dependency graph structures:

1) **Random (R):** random start/goal configurations.
2) **Single cycle (S):** configurations inducing one DG cycle.
3) **Double cycle (D):** configurations inducing two distinct DG cycles.
4) **Mixed (M):** configurations inducing a mixture of independent objects, chains, and cycles.

For each category, randomness is injected whenever appropriate in creating the start/goal configurations. Representative cases of the test cases are illustrated in Fig. 4. We denote each case as *R#*, *S#*, or *D#*, where the number indicates the number of objects, and as *M#* for mixed cases, where the number is simply an index as all mixed cases use 12 objects.

We evaluate SDAR against the previous state-of-the-art [20] as the baseline. Beyond the direct comparison, we also consider several hybrid configurations to independently evaluate SDAR-T and SDAR-M (TP means task planner and MP means motion planner): (1) Baseline TP + SDAR-M, (2) SDAR-T + Baseline MP, (3) SDAR-T + cuRobo, and (4) Baseline TP + cuRobo.

A. Task Planning Performance

We first evaluate the quality of the task plans produced by SDAR-T in comparison to the baseline. The number of actions in a plan is computed by counting each task assigned to either arm as one action and summing across both arms. Although SDAR-T may generate multiple potential tasks at each step, the motion planner ultimately selects one option, and this determined sequence is used for evaluation.

Fig. 6 shows the absolute number of actions for the subset of test cases where SDAR-T and the baseline produce different results. Lighter bars correspond to SDAR-T and darker bars to the baseline, with colors indicating the four dependency graph categories (red/M, green/S, blue/D, yellow/R). In nearly all cases, SDAR-T uses fewer or an equal number of actions. An exception occurs in D7 (double cycle, 7 objects), where the baseline produces a plan with fewer actions by holding one object continuously. The

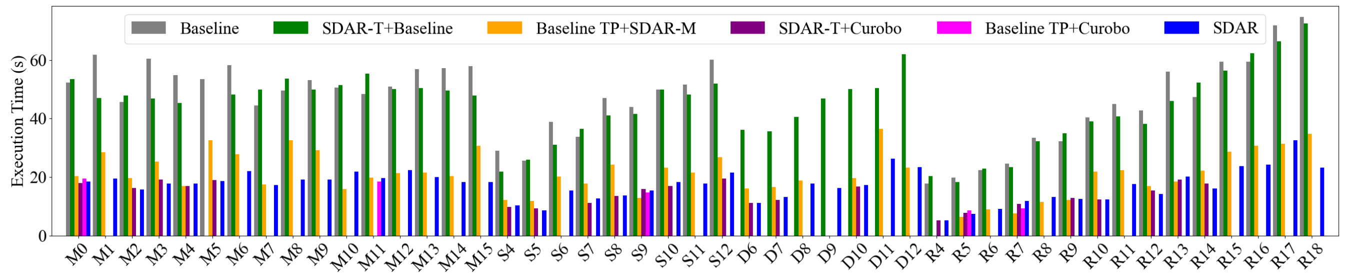

Fig. 5: Execution time for all simulation-evaluated test cases. Where the horizontal legends are as defined at the start of this section (e.g., R10 means the random setting with 10 objects). Missing bars for a given test case and algorithm combination indicate the method failed to produce a valid solution.

baseline plan, however, is very inefficient because that arm is essentially idling when holding an object, leaving the other arm to do all the work.

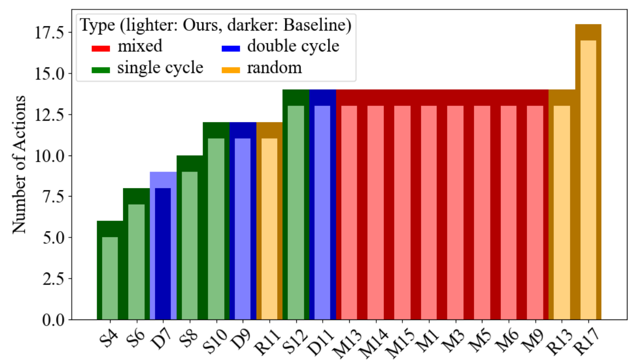

Fig. 6: Absolute number of actions per test case for instances where our method and the baseline differ. Lighter bars correspond to our method and darker bars to the baseline, with colors indicating configuration type (red = mixed, green = single cycle, blue = double cycle, yellow = random).

Fig. 7 reports the ratio of action counts of baseline over SDAR-T. Values above 1.0 indicate that the baseline required more actions. SDAR-T consistently produces shorter or equal plans in mixed, single-cycle, and double-cycle settings, while achieving parity in random configurations. These results demonstrate that our method generally reduces the number of actions compared to the baseline, which builds on an optimal single-arm task planner [3].

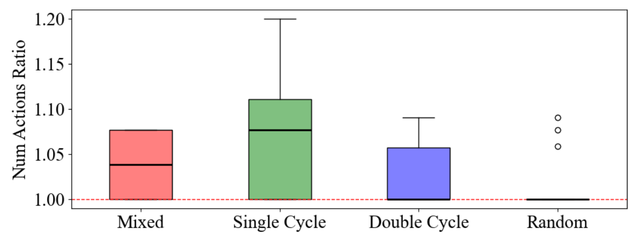

Fig. 7: Ratio of the number of actions (baseline/SDAR) grouped by dependency type without case D7.

We now evaluate the end-to-end performance of SDAR. Fig. 10 reports the success rates, defined as the ability to generate a feasible dual-arm plan that can be executed without collision or deadlock. Our full method achieves a 100% success rate, demonstrating the robustness of SDAR's multi-level search architecture. The baseline method achieves 85% success, while hybrid variants that mix task and motion planners perform better but still fall short of SDAR. Baseline TP+cuRobo in particular suffers from extremely low success (11%), showing that even strong motion planning cannot compensate for weak task planning. Similarly, SDAR-T +cuRobo achieves only 49% success, underscoring the importance of robust motion planning strategies.

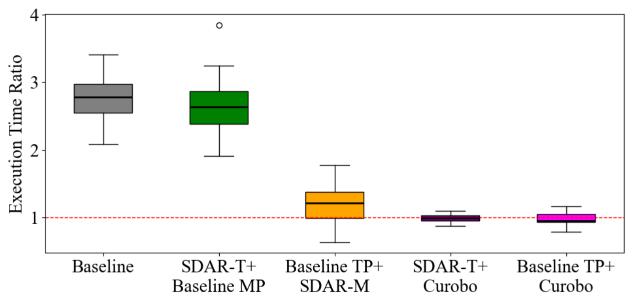

Fig. 8: Execution time ratios compared to our method (red dashed line = 1.0). Motion planner quality is the dominant factor for execution efficiency.

B. Overall Task and Motion Planning Performance

Fig. 8 summarizes execution time performance across categories relative to our method, SDAR. The results suggest execution efficiency is largely dictated by the motion planner. When paired with the same motion planner (either ours or the baseline's), our task planner (SDAR-T) consistently leads to lower execution time compared to the baseline task planner, delivering over 60% execution time savings. At the same time, replacing a weaker motion planner with a stronger one yields the largest gains regardless of the task planner. This explains why both SDAR-T +cuRobo and Baseline TP+cuRobo achieve very low execution time ratios: cuRobo produces high-quality motion plans efficiently, but without fallback mechanisms, these methods fail frequently, as reflected in their success rates. Fig. 5 presents the absolute execution time across cases and methods for a quick assessment of all methods.

Taken together, these results demonstrate that robust and efficient dual-arm planning requires not only high-performance task planner and motion planner individually, but also tight integration between the two, which underlies the design principle of SDAR.

C. Computation Time

We profiled SDAR across all test cases and observed that SDAR-M takes nearly all planning time. Within SDAR-M, 9% of computation time (averaging 0.52 seconds per step) is

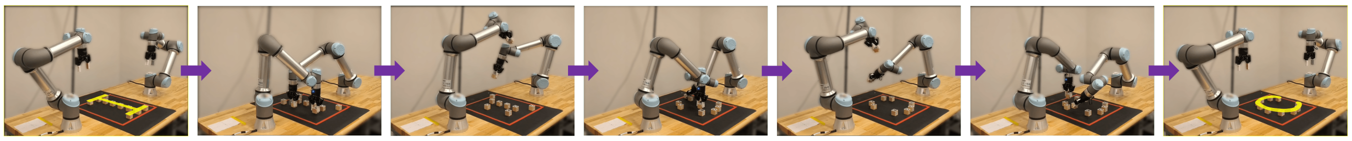

Fig. 9: Snapshots from the real-robot experiment on dual-arm rearrangement. Two UR5e manipulators with Robotiq grippers rearrange ArUco-tagged cubes from the letter “I” to “C” as part of forming “ICRA2026”.

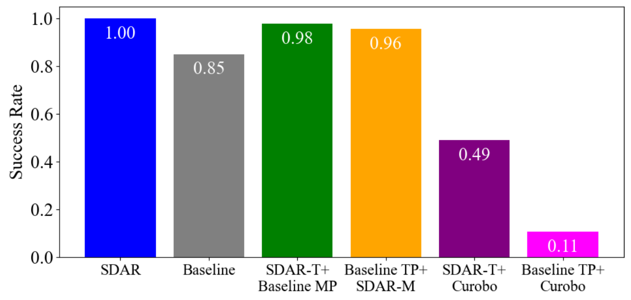

Fig. 10: Success rates achieved by different task planner and motion planner combinations over all test cases.

spent on IK computation for plan selection, while 91% (5.17 seconds per step) is used to compute motion plans for the selected task plan. This further corroborates the importance of an efficient motion planner with a high success rate.

D. Real-Robot Experiment

We conducted real-world experiments using two UR5e manipulators equipped with Robotiq 2F-85 grippers to confirm that SDAR indeed produces high-quality, collision-free plans. The task asks the two robots to rearrange 10 wooden cuboids to form letters in “ICRA2026”, one by one. Our planner successfully generated collision-free trajectories for both arms, which were executed in parallel on robot hardware. Snapshots from the experiment are shown in Fig. 9, highlighting an intermediate step where the arrangement transitioned from the letter “I” to “C”. A full video demonstration is also included, showing the complete execution.

VI. Conclusion and Discussions

This work introduced the Synchronous Dual-Arm Rearrangement Planner (SDAR), a task and motion planning framework designed for solving long-horizon tabletop rearrangement tasks with complex (non-monotone) object dependencies. SDAR tightly integrates dependency-driven task planning with robust synchronous dual-arm motion generation. By coupling layered decomposition of dependency graphs with sampling-based motion optimization and fallback strategies, SDAR consistently delivers high-quality solutions, near real-time efficiency, and a 100% success rate across challenging long-horizon rearrangement tasks. Comprehensive evaluation confirms that SDAR delivers state-of-the-art task planning and motion planning performances separately, and as a whole, achieves 60+% reduction in task execution time. Real-robot experiments with dual UR5e manipulators demonstrate the framework’s ability to transfer readily and reliably to hardware.

Looking ahead, many promising avenues remain; we mention a few here. From the tasking planning perspective, dual-arm systems provide the unique “swapping” primitive that is beyond the capability of a single arm. It is interesting to explore what a k-arm system can achieve with regard to task planning. From the motion planning side, currently, we are exploring improving motion planning efficiency through better parallelization and learning-guided sampling to further reduce latency toward true real-time dual-arm deployment. Beyond improving task and/or motion planning performance, to make frameworks like SDAR more useful for general-purpose manipulation, richer perception, and online feedback should be tightly integrated into the task-motion planning loop. Lastly, getting back to SDAR, an elephant in the room is the word “synchronous”. Asynchronous dual-arm coordination will drastically expand the solution space and can potentially lead to a far superior TAMP solution. However, how to tame the ensuing combinatorial explosion?

References

[1] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Pérez, “Integrated task and motion planning,” *Annual review of control, robotics, and autonomous systems*, vol. 4, no. 1, pp. 265–293, 2021.

[2] 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.

[3] K. Gao, S. W. Feng, B. Huang, and J. Yu, “Minimizing running buffers for tabletop object rearrangement: Complexity, fast algorithms, and applications,” *The International Journal of Robotics Research*, vol. 42, no. 10, pp. 755–776, 2023.

[4] J. E. Hopcroft, J. T. Schwartz, and M. Sharir, “On the complexity of motion planning for multiple independent objects; pspace-hardness of the ‘warehouseman’s problem’,” *The International Journal of Robotics Research*, vol. 3, no. 4, pp. 76–88, 1984.

[5] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling, “Pddlstream: Integrating symbolic planners and blackbox samplers via optimistic adaptive planning,” in *Proceedings of the international conference on automated planning and scheduling*, vol. 30, 2020, pp. 440–448.

[6] K. Gao, Y. Ding, S. Zhang, and J. Yu, “Orla*: Mobile manipulator-based object rearrangement with lazy a*,” 2023.

[7] A. Curtis, X. Fang, L. P. Kaelbling, T. Lozano-Pérez, and C. R. Garrett, “Long-horizon manipulation of unknown objects via task and motion planning with estimated affordances,” in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 1940–1946.

[8] X. Zhang, Y. Zhu, Y. Ding, Y. Zhu, P. Stone, and S. Zhang, “Visually grounded task and motion planning for mobile manipulation,” in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 1925–1931.

[9] L. Wang, X. Meng, Y. Xiang, and D. Fox, “Hierarchical policies for cluttered-scene grasping with latent plans,” *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 2883–2890, 2022.

[10] M. Shridhar, L. Manuelli, and D. Fox, “Perceiver-actor: A multi-task transformer for robotic manipulation,” in *Conference on Robot Learning*. PMLR, 2023, pp. 785–799.

[11] Z. Yang, C. Garrett, D. Fox, T. Lozano-Pérez, and L. P. Kaelbling, “Guiding long-horizon task and motion planning with vision language models,” in *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2025, pp. 16 847–16 853.

[12] Y. Xu, J. Mao, L. Li, Y. Du, T. Lozáno-Pérez, L. P. Kaelbling, and D. Hsu, “” set it up”: Functional object arrangement with compositional generative models,” *arXiv preprint arXiv:2508.02068*, 2025.

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

[14] K. Gao, D. Lau, B. Huang, K. E. Bekris, and J. Yu, "Fast high-quality tabletop rearrangement in bounded workspace," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 1961–1967.

[15] W. Thomason, Z. Kingston, and L. E. Kavraki, "Motions in microseconds via vectorized sampling-based planning," 2023.

[16] B. Sundaralingam, S. K. S. Hari, A. Fishman, C. Garrett, K. Van Wyk, V. Blukis, A. Millane, H. Oleynikova, A. Handa, F. Ramos, N. Ratliff, and D. Fox, "Curobo: Parallelized collision-free robot motion generation," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 8112–8119.

[17] J. Chen, J. Li, Y. Huang, C. Garrett, D. Sun, C. Fan, A. Hofmann, C. Mueller, S. Koenig, and B. C. Williams, "Cooperative task and motion planning for multi-arm assembly systems," *arXiv preprint arXiv:2203.02475*, 2022.

[18] H. Zhang, S.-H. Chan, J. Zhong, J. Li, S. Koenig, and S. Nikolaidis, "A mip-based approach for multi-robot geometric task-and-motion planning," in *Proceedings of the IEEE International Conference on Automation Science and Engineering (CASE)*, 2022, pp. 2102–2109.

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

[20] K. Gao and J. Yu, "Toward efficient task planning for dual-arm tabletop object rearrangement," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022.

[21] J. Li, A. Felner, E. Boyarski, H. Ma, and S. Koenig, "Improved heuristics for conflict-based search for multi-agent path finding," in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2019, pp. 442–449.

[22] Y. Ding, X. Zhang, C. Paxton, and S. Zhang, "Task and motion planning with large language models for object rearrangement," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 2086–2092.

[23] H. Chang, K. Gao, K. Boyalakuntla, A. Lee, B. Huang, J. Yu, and A. Boularias, "Lgmcts: Language-guided monte-carlo tree search for executable semantic object rearrangement," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024, pp. 13 607–13 612.

[24] K. Gao, J. Yu, T. S. Punjabi, and J. Yu, "Effectively rearranging heterogeneous objects on cluttered tabletops," 2023.

[25] A. Xu, K. Gao, S. W. Feng, and J. Yu, "Optimal and stable multi-layer object rearrangement on a tabletop," 2023.

[26] 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. [Online]. Available: https://doi.org/10.1177/0278364918780999

[27] K. Gao, S. W. Feng, and J. Yu, "On minimizing the number of running buffers for tabletop rearrangement," in *Robotics: Science and Systems (RSS)*, 2021.

[28] S. M. La Valle, "Motion planning," *IEEE Robotics & Automation Magazine*, vol. 18, no. 2, pp. 108–118, 2011.

[29] T. Lozano-Pérez and M. A. Wesley, "An algorithm for planning collision-free paths among polyhedral obstacles," *Communications of the ACM*, vol. 22, no. 10, pp. 560–570, 1979.

[30] D. Zhang, Z. Ye, and J. Yu, "Asymptotically-optimal multi-query path planning for a polygonal robot," in *IEEE International Conference on Robotics and Automation*, 2025.

[31] R. Bohlin and L. E. Kavraki, "Path planning using lazy prm," in *Proceedings 2000 ICRA. Millennium conference. IEEE international conference on robotics and automation. Symposia proceedings (Cat. No. 00CH37065)*, vol. 1. IEEE, 2000, pp. 521–528.

[32] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.

[33] M. P. Strub and J. D. Gammell, "Adaptively informed trees (ait*): Fast asymptotically optimal path planning through adaptive heuristics," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 3191–3198.

[34] J. D. Gammell, T. D. Barfoot, and S. S. Srinivasa, "Batch informed trees (bit*): Informed asymptotically optimal anytime search," *The International Journal of Robotics Research*, vol. 39, no. 5, pp. 543–567, 2020.

[35] J. Schulman, J. Ho, A. X. Lee, I. Awwal, H. Bradlow, and P. Abbeel, "Finding locally optimal, collision-free trajectories with sequential convex optimization." in *Robotics: science and systems*, vol. 9, no. 1. Berlin, Germany, 2013, pp. 1–10.

[36] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, "Motion planning with sequential convex optimization and convex collision checking," *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1251–1270, 2014.

[37] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa, "Chomp: Covariant hamiltonian optimization for motion planning," *The International journal of robotics research*, vol. 32, no. 9-10, pp. 1164–1193, 2013.

[38] D. Zhang, C. Liang, X. Gao, K. Wu, and Z. Pan, "Provably robust semi-infinite program under collision constraints via subdivision," *arXiv preprint arXiv:2302.01135*, 2023.

[39] C. Liang, X. Gao, K. Wu, and Z. Pan, "Second-order convergent collision-constrained optimization-based planner," *IEEE Robotics and Automation Letters*, vol. 9, no. 6, pp. 4950–4957, 2024.

[40] T. Marcucci, M. Petersen, D. von Wrangel, and R. Tedrake, "Motion planning around obstacles with convex optimization," *Science robotics*, vol. 8, no. 84, p. eadf7843, 2023.

[41] C. Smith, Y. Karayiannidis, L. Nalpantidis, X. Gratal, P. Qi, D. V. Dimarogonas, and D. Kragic, "Dual arm manipulation—a survey," *Robotics and Autonomous systems*, vol. 60, no. 10, pp. 1340–1353, 2012.

[42] Y. Koga and J.-C. Latombe, "On multi-arm manipulation planning," in *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*. IEEE, 1994, pp. 945–952.

[43] M. Gharbi, J. Cortés, and T. Siméon, "Roadmap composition for multi-arm systems path planning," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2009, pp. 2471–2476.

[44] Y. Shaoul, I. Mishani, M. Likhachev, and J. Li, "Accelerating search-based planning for multi-robot manipulation by leveraging online-generated experiences," in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 34, 2024, pp. 523–531.

[45] Y. Shaoul, R. Veerapaneni, M. Likhachev, and J. Li, "Unconstraining multi-robot manipulation: Enabling arbitrary constraints in ecbs with bounded sub-optimality," in *Proceedings of the International Symposium on Combinatorial Search*, vol. 17, 2024, pp. 109–117.

[46] P. Huang, Y. Shaoul, and J. Li, "Benchmarking shortcutting techniques for multi-robot-arm motion planning," *arXiv preprint arXiv:2508.05027*, 2025.

[47] H. Ha, J. Xu, and S. Song, "Learning a decentralized multi-arm motion planner," *arXiv preprint arXiv:2011.02608*, 2020.

[48] P. Huang, R. Liu, C. Liu, and J. Li, "Apex-mr: Multi-robot asynchronous planning and execution for cooperative assembly," *arXiv preprint arXiv:2503.15836*, 2025.

[49] H. Zhang, S.-H. Chan, J. Zhong, J. Li, P. Kolapo, S. Koenig, Z. Agioutantis, S. Schafrik, and S. Nikolaidis, "Multi-robot geometric task-and-motion planning for collaborative manipulation tasks," *Autonomous Robots*, vol. 47, no. 8, pp. 1537–1558, 2023.

[50] K. Gao, Z. Ye, D. Zhang, B. Huang, and J. Yu, "Toward holistic planning and control optimization for dual-arm rearrangement," *arXiv preprint arXiv:2404.06758*, 2024.

[51] Y. Wang and H. Kasaei, "Learning dual-arm push and grasp synergy in dense clutter," *IEEE Robotics and Automation Letters*, 2025.

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

[53] 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.

[54] H. Kim, Y. Ohmura, and Y. Kuniyoshi, "Transformer-based deep imitation learning for dual-arm robot manipulation. in 2021 ieee," in *RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 8965–8972.

[55] V. Parimi and B. C. Williams, "Diffusion-guided multi-arm motion planning," *arXiv preprint arXiv:2509.08160*, 2025.