2022 IEEE International Conference on Robotics and Automation (ICRA)
May 23-27, 2022. Philadelphia, PA, USA

Coordination of two robotic manipulators for object retrieval in clutter

Jeeho Ahn[1], ChangHwan Kim[1], and Changjoo Nam[2,*]

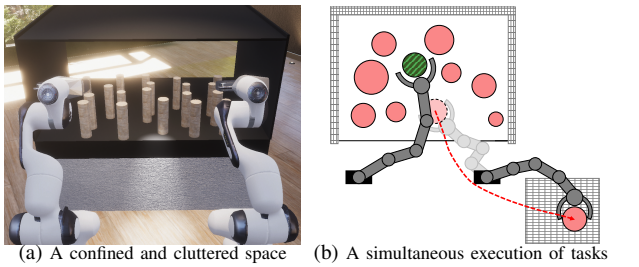
***Abstract*— We consider the problem of retrieving a target object from a confined space by two robotic manipulators where overhand grasps are not allowed. If other movable obstacles occlude the target, more than one object should be relocated to clear the path to reach the target object. With two robots, the relocation could be done efficiently by simultaneously performing relocation tasks. However, the precedence constraint between the tasks (e.g, some objects at the front should be removed to manipulate the objects in the back) makes the simultaneous task execution difficult.**

We propose a coordination method that determines *which robot relocates which object* so as to perform tasks simultaneously. Given a set of objects to be relocated, the objective is to maximize the number of switches between the robots in performing relocation tasks. Thus, one robot can pick an object in the clutter while the other robot places an object in hand to the outside of the clutter. However, the object to be relocated may not be accessible to all robots, so switching could not always be achieved. Our method is based on the uniform-cost search so the number of switches can be maximized. We also propose a greedy variant whose computation time is shorter. From experiments, we show that our method reduces the completion time of the mission by at least 22.9% (at most 27.3%) compared to the methods with no consideration of switching.

I. Introduction

Manipulation planning to rearrange objects in clutter has recently received significant attention [1]–[4] as it has a wide variety of applications from domestic services to manufacturing. Among them, there has been a line of research [5]–[10] that aims to retrieve a target object from cluttered and confined spaces where overhand grasps are not allowed, like shelves and fridges. These works focus on generating plans for what to relocate in what order. Since multiple objects are manipulated, distributing the manipulation tasks to multiple robots may enable faster completion of the mission.

However, coordinating multiple manipulators in a confined space (Fig. 1a) is challenging as the robots have limited (and varying) configuration spaces if they move simultaneously. The manipulation tasks have precedence constraints since some objects only can be manipulated after some front objects are relocated. This constraint and the limited configuration space may force only one robot to execute a task at a time. In order for the robots to work simultaneously, the actions of the robots need to be well coordinated. For example, one robot picks an object while the other places an object outside the clutter (Fig. 1b). Then the robots can comply with the precedence constraints and use a larger configuration space. However, such coordination is not possible if only one of the robots can access the objects (e.g., the path from the other robot to the objects is blocked).



(a) A confined and cluttered space (b) A simultaneous execution of tasks

Fig. 1: Coordination of two robots for object retrieval. (a) An example configuration of the robots and objects. (b) While the right robot is placing an object, the left robot can pick an object that becomes accessible after the right robot removes the object being placed.

Therefore, we develop a method that coordinates robot actions while considering the accessibility of the objects. We aim to maximize the number of switches between the robots in performing relocation tasks. A relocation task is defined as picking an object and placing it to another location. If two robots take turns, they are likely to perform tasks simultaneously so the retrieval mission can be done efficiently. Specifically, the method finds relocation tasks and allocates them to the robots using the uniform-cost search. The search finds an optimal allocation describing *what to relocate in what order using which robot* such that the number of switches is maximized. Given the allocation, the method sequences robot actions (e.g., pick, place) for simultaneous execution. Once the method finds collision-free motions for all actions, the robots start relocation.

However, the retrieval problem is computationally intractable even with only one robot [11]. Although the method can replan if some of the motions turn out to be inexecutable, repeated replanning could delay the completion of the mission. To prevent such delays, we develop a greedy variant that determines the allocation in an online manner.

The following are contribution of this work:

- We propose a method that finds an allocation of relocation tasks to robots with the maximum number of switches. Through sequencing actions to perform tasks simultaneously, the mission can be completed quickly.
- We also propose a greedy variant for situations where replanning occurs repeatedly.
- We provide analyses of the methods regarding optimality, time complexity, and completeness.
- We show the results from extensive experiments and comparisons with other methods that do not set out to optimize the number of switches.

This work was supported in part by Samsung Research Funding & Incubation Center of Samsung Electronics under Project Number SRFC-TD2103-01, the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2022R1C1C1008476), and the Technology Innovation Program and Industrial Strategic Technology Development Program (10077538, Development of manipulation technologies in social contexts for human-care service robots). [1]Korea Institute of Science and Technology, [2]Dept. of Electronic Engineering, Sogang University. *Corresponding author: cjnam@sogang.ac.kr

2022 IEEE International Conference on Robotics and Automation (ICRA) | 978-1-7281-9681-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICRA46639.2022.9811978

II. Related Work

Planning for multiple robotic arms has been studied in many contexts. One of the topics that received attention is hand-off between robots to transfer objects. [12]–[15] consider task and/or motion planning for multiple manipulators to transfer a single object between two locations via hand-offs and find collision-free motions for the transfer. While these approaches can be helpful for our object retrieval to transfer picked objects to the place outside the clutter, they do not directly solve the retrieval problem.

In [16], a planning and control method for dual-arm fruit harvesting is presented where the robot is in front of multiple fruits to be harvested. The arms of the robot can harvest fruits (aubergine) simultaneously, in turn, or in collaboration with each other. [17] use two arms to rearrange blocks on a table to a predefined configuration. Since these works consider the arrangement of objects where any object can be picked without relocating some others, the manipulation tasks do not have a precedence constraint. [18] propose methods based on precomputed information about collisions of two arms, called the coordination matrix encoding all trajectories of the arms to perform all manipulation tasks. They determine the order of the objects to be manipulated and find the motions to pick and place the objects using the matrix. The tasks in this work do not have precedence constraints, so the two arms can manipulate objects simultaneously. Also, the methods take a long time to find a solution with clutter environments, which ranges from several minutes to several tens of minutes (the time varies whether the matrix is precomputed or not).

The methods proposed in [19], [20] integrate task allocation and motion scheduling for task optimization for dual-arm coordination. They are with a solver that finds solutions for constraint satisfaction problems so they can handle precedence constraints. Despite being generic and computationally efficient, they consider environments where robots do not have cramped configuration spaces (like an open space but not a clutter). In [21], the proposed method uses a surrogate checker to evaluate the feasibility of task assignments (i.e., if collision-free motions exist) to work in an environment that is more constrained. Also, the method can handle tasks with precedence constraints. However, it requires a predefined task network specifying tasks with pre-conditions and post-conditions. We do not want to hand-code such conditions (i.e., what to relocate in what order in our problem) but find a solution using a geometric reasoner.

More importantly, these works do not directly solve the object retrieval problem. To the best of our knowledge, our work is the first to tackle the coordination of multiple manipulators for object retrieval in clutter (so motion planning is nontrivial) where manipulation tasks have precedence constraints. This work aims to fill the gap by developing fast methods to determine relocation tasks with precedence constraints and allocate them to two robots.

III. Problem Description

In order to retrieve a target object from clutter using two manipulators, there must be several processes to be completed, such as sensing, perception, task planning and allocation, motion planning for the allocated tasks, grasp planning, and control. Among these, we focus on task planning and allocation and motion planning to distribute relocation tasks. The target retrieval problem among multiple movable objects has shown to be computationally intractable with one manipulator [11]. Since adding more robots increases the complexity of the problem structure, the target retrieval with more than one robot must not make the problem easier.

A. Assumptions

We have several assumptions to focus on the problems that have not been tackled in the literature: (i) Each robot can manipulate one object at a time, and each object requires exactly one robot to manipulate it.[1] (ii) The computations for task planning, task allocation, and motion planning are centralized. (iii) The trivial problem instances where the robot can grasp the target without relocating any object are not considered. The infeasible problem instances where none of the objects can be grasped are not considered. (iv) The robots use prehensile actions only (i.e., pick, place). (v) Overhand grasps are not allowed (e.g., the top is blocked by shelves). (vi) Objects are modeled by 3D cylinders (which could have different radii) so the objects can be grasped from any direction. We will relax some of these assumptions in our future work once we solve the coordination problem.[2]

B. Problem definition

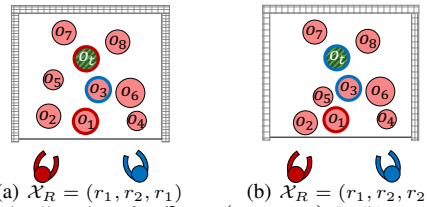
Our goal is to coordinate two manipulators efficiently to retrieve the target object from the confined space cluttered with other movable obstacles. To achieve this goal, we need to solve three subproblems: (A) determine the objects to relocate to clear the path to the target and the order of their relocation (i.e., finding the relocation tasks and their precedence constraints); (B) allocate the relocation tasks to the robots; and (C) find collision-free motions of the robots for the allocated tasks. Since the entire problem is computationally intractable, we solve them sequentially for faster computation. Thus, each subproblem is conditioned on the result of the previous subproblem.

We set objective values for the subproblems. For (A), we aim to minimize the number of relocated objects, which has shown to be effective in reducing the mission completion time [11], [25]. For (B), the objective is to maximize the number of switches between the robots in performing relocation tasks. Lastly, we aim to minimize the execution time of the relocation tasks, known as makespan.

Suppose that an environment is with $N \in \mathbb{Z}^{\geq 2}$ objects and $M \in \mathbb{Z}^{+}$ robotic manipulators where we only consider $M = 2$ in this work. The set $\mathcal{O} = \{o_1, \cdots, o_{N-1}, o_t\}$ includes all objects with $N-1$ movable obstacles and one target object o_t. Likewise, $\mathcal{R} = \{r_1, \cdots, r_M\}$ includes all robots. The centroid and radius of an object i is described by (x_i, y_i, z_i)

[1] According to the taxonomy for multi-robot task allocation [22], our problem falls into the ST-SR-TA category (**S**ingle-**T**ask robots, **S**ingle-**R**obot tasks, **T**ime-extended **A**ssignment), which is strongly $\mathcal{NP}$-hard [23].

[2] Our previous work can be employed for nonprehensile actions [24] and varying reachable directions [6].



(a) $\mathcal{X}_R = (r_1, r_2, r_1)$ (b) $\mathcal{X}_R = (r_1, r_2, r_2)$

Fig. 2: Example allocations for $\mathcal{O}_R = (o_1, o_3, o_t)$. In (b), r_1 cannot access o_t even after o_1 and o_3 are removed because o_2 and o_5 occlude o_t.

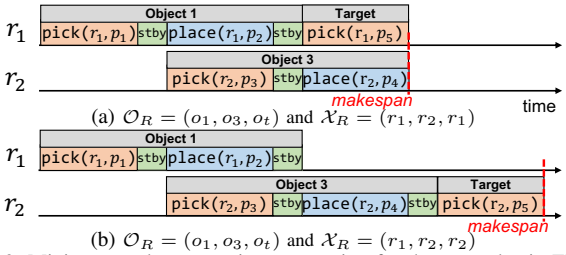
and r_i, respectively. The target is within the intersection of the workspaces of the robots so $(x_t, y_t, z_t) \in \mathcal{W} = \mathcal{W}_1 \cap \cdots \cap \mathcal{W}_M \neq \emptyset$ for all $i = 1, \cdots, N$.

We define three action primitives of the robots to perform relocation tasks: `pick`, `place`, `standby`. The `pick` and `place` actions are specified by a robot ID and a Cartesian coordinate $p \in \mathbb{R}^3$ to indicate the goal location to perform the actions. `standby`(r_i) moves r_i to a predefined standby pose. For example, `pick`(r_1, p_1) let r_1 move to p_1, and grasp the object at p_1. The action `place`(r_1, p_2) indicates that r_1 moves to p_2, and releases the object in hand.

(A) Object rearrangement planning (ORP): Let $\mathcal{O}_R$ be a tuple representing the sequence of objects to be relocated (including the target) where $|\mathcal{O}_R| = k \leq N$. A mathematical definition of the ORP is to find $\mathcal{O}_R$ that minimizes k. An example sequence $\mathcal{O}_R = (o_1, o_3, o_t)$ means that o_1 and o_3 should be removed sequentially in order to retrieve the target, as shown in the both instances in Fig. 2.

(B) Multi-manipulator task allocation (MMTA): Let $\mathcal{X}_R$ be a tuple representing the sequence of robots to relocate objects in $\mathcal{O}_R$. In other words, $\mathcal{X}_R$ is an allocation of relocation tasks to robots. Suppose that we have $\mathcal{O}_R = (o_1, o_3, o_t)$ and $\mathcal{X}_R = (r_1, r_2, r_1)$ (see the color coding in Fig. 2a). The red r_1 and blue r_2 relocate o_1 and o_3, respectively. Then r_1 finally retrieves o_t. Let t be the number of switches between the robots in $\mathcal{R}$ to relocate $\mathcal{O}_R$ where $0 \leq t \leq k-1$. The MMTA is to find $\mathcal{X}_R$ such that t is maximized. An object o_i may not be relocated by r_i if r_i cannot access o_i owing to occlusions. Thus, the robots may not always be able to switch like r_1 (red) in Fig. 2b that cannot access o_t even after o_1 and o_3 are removed owing to o_2 and o_5. Thus, $\mathcal{X}_R = (r_1, r_2, r_2)$.

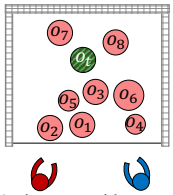
(C) Minimum makespan action sequencing (MMAS): Minimum makespan motion planning (MMMP) is to minimize the total completion time (i.e., makespan) to relocate $\mathcal{O}_R$ by $\mathcal{X}_R$. Since the MMMP on a grid (i.e., in the discrete space) is strongly $\mathcal{NP}$-hard [26], MMMP in the continuous space of robot joints is also $\mathcal{NP}$-hard. To develop a practical method that works fast, we simplify the difficult problem to an action sequencing problem (MMAS). We let the robots perform `pick`(r_i, o_a) and `place`(r_j, o_b) simultaneously if $i \neq j$ and $a \neq b$. If the robots execute the actions simultaneously, we synchronize the start time of the actions (their end time does not necessarily synchronize). `standby` actions are inserted between every `pick` and `place` to start the following action from the same pose (necessary for upfront offline motion planning). This synchronization makes the problem easy since we do not have to be concerned about



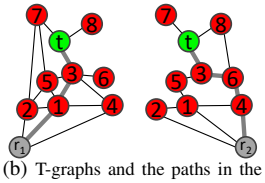
(a) $\mathcal{O}_R = (o_1, o_3, o_t)$ and $\mathcal{X}_R = (r_1, r_2, r_1)$

(b) $\mathcal{O}_R = (o_1, o_3, o_t)$ and $\mathcal{X}_R = (r_1, r_2, r_2)$

Fig. 3: Minimum makespan action sequencing for the examples in Fig. 2



(a) An instance with two robots



(b) T-graphs and the paths in the graphs

Fig. 4: Example T-graphs. The relocation plans for r_1 and r_2 are $\mathcal{O}_R^1 = (o_1, o_3, o_t)$ and $\mathcal{O}_R^2 = (o_4, o_6, o_3, o_t)$, respectively.

allocating the actions to specific time frames.[3] The action sequences of the examples from Fig. 2 are shown in Fig. 3.

IV. The Methods for Coordinating Robots

We describe the methods for the three subproblems (A–C). For (A), we introduce an existing method finding a relocation plan $\mathcal{O}_R$. Then we propose our methods for (B) and (C). The method for (B) uses the uniform-cost search with exponential time complexity. Thus, we propose a greedy variant that finds which robot relocates which object in an online fashion.

A. Preliminary: Object rearrangement planning

We employ a method proposed in [6], [11] to find $\mathcal{O}_R$. This method generates a Traversability graph (T-graph). The graph represents movable paths of objects in clutter. A node represents the location of an object or the end-effector. An edge is connected between a pair of nodes if a collision-free path of the end-effector exists between the locations corresponding to the nodes. Collisions are examined by a surrogate checker [10] without motion planning of the whole arm. Thus, the graph construction runs in polynomial time. Then a shortest path in the graph from the robot node to the target is a sequence of nodes representing the smallest number of objects to relocate. The two robots could have different T-graphs and shortest paths. Example graphs and paths (the nodes on the bold gray edges) for r_1 and r_2 are shown in Fig. 4b. The relocation plans for r_1 and r_2 are $\mathcal{O}_R^1 = (o_1, o_3, o_t)$ and $\mathcal{O}_R^2 = (o_4, o_6, o_3, o_t)$, respectively.

B. Multi-manipulator task allocation

Based on $\mathcal{O}_R^i$ for $i = 1, 2$ from (A), we compute the allocation of the relocation tasks to r_i. We propose two methods where Search finds the allocation for all tasks while Greedy instantaneously determines the robot to relocate.

1) Uniform-cost search: Given $\mathcal{O}_R^i$ and r_i for $i = 1, 2$, Search performs the uniform-cost search. Two initial nodes are generated from the root for $\mathcal{O}_R^1$ and $\mathcal{O}_R^2$, respectively (the root node is omitted in Fig. 5 to save the space). As a visual guide, we draw the T-graphs inside the nodes only

[3]However, this simplification could incur a gap between the original problem and the problem that we solve.

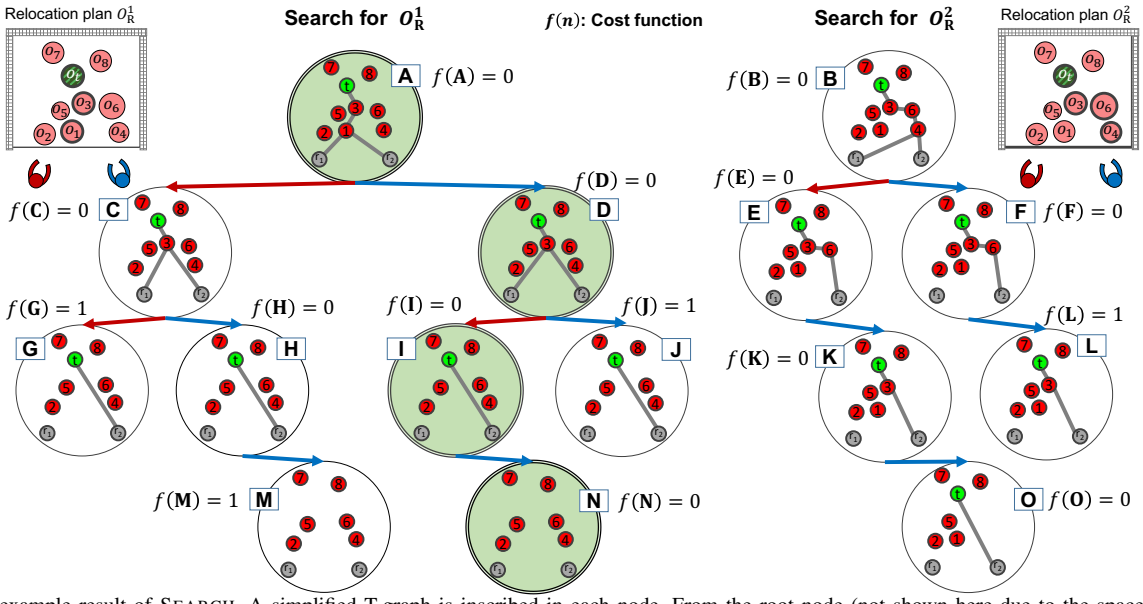


Fig. 5: An example result of SEARCH. A simplified T-graph is inscribed in each node. From the root node (not shown here due to the space limit), two children A and B are generated for $\mathcal{O}_R^1$ and $\mathcal{O}_R^2$. A is expanded as it is generated earlier. Nodes C and D are generated. Then B, C, and D with the same cost are on the frontier. The oldest B is chosen for expansion and E and F are generated. Then C is expanded so D, E, F, G, and H are in the frontier list. The oldest D among those with the minimum cost is chosen then now I and J are added to the frontier list. Among the nodes with zero cost, E and F are expanded in turn so K and L are generated. In E and F, r_1 cannot access the next object (no edge in the T-graph between the robot node and the object node) so only the right subtrees are generated. Next, M is generated after H expands. The target is removed in M, but the goal check is done after the node is chosen for expansion. Next, I is expanded and N is generated. At this point, the frontier nodes with zero cost are K and N. After O is generated, N is chosen for expansion. The goal check finds the goal state N so the search terminates.

with the path to the target node for simplicity. Since the two initial nodes, a child node represents the state of objects after one object is relocated. The child in the left (via the red arrow) represents the state where the object is relocated by r_1 (the red robot). For example, node C in Fig. 5 represents the state where o_1 is relocated by r_1 from the state in A. Similarly, the blue arrow means that the relocation is done by r_2 (the blue robot). The search terminates if the goal state, where the target is finally relocated, is found. The nodes highlighted by green in Fig. 5 represent the solution, that is $\mathcal{X}_R = (r_2, r_1, r_2)$ for $\mathcal{O}_R = (o_1, o_3, o_t)$.

The plans $\mathcal{O}_R^1$ and $\mathcal{O}_R^2$ have the smallest numbers of objects to be relocated by r_1 and r_2, respectively. Although only one search can run for either of the plans with fewer tasks, like $\mathcal{O}_R^1$, SEARCH runs the search for both plans to ensure that the solution relocates the smallest number of objects even if motion planning fails. Suppose that we expand only the left subtree of the root. If no collision-free motion is found for relocating any of the objects in $\mathcal{O}_R^1$, a new path to the target is computed after updating the T-graph. This new path could result in relocating more objects in total than executing the plan $\mathcal{O}_R^2$. If so, $\mathcal{O}_R^2$ should be chosen in order to maintain the least number of objects to be relocated.

The uniform-cost search is guided by the evaluation function $f(n) = g(n)$ for node n where $g(n)$ represents the sum of penalties imposed if no switching occurs. Nodes have zero cost up to depth 2 of the tree (from the root to its grandchildren) since performing no or a single task cannot incur any switching. From depth 3 (from G in Fig. 5), a unit penalty is given if the robots do not switch. For example, G in Fig. 5 has $g(\mathsf{G}) = 1$ since no switching occurs between C and G. The node for expansion is chosen such that the chosen node has the minimum $f(n)$ among the nodes in the frontier list. If some nodes on the frontier have the same cost, the node generated earlier is chosen. In Fig. 5, the nodes are named alphabetically according to the order that they are generated. Since the behavior of the search is the same as the traditional uniform-cost search [27] and we have the space limit, we do not provide the pseudocode but a detailed illustration in Fig. 5 instead.

Once the search is done, the tasks and their allocation $\mathcal{O}_R$ and $\mathcal{X}_R$ can be obtained from the nodes chosen for expansion. This allocation is then forwarded to the next stage (MMAS) for action sequencing, motion planning, and execution. However, motion planning could fail to pick or place some of the objects in the plan. Then, the process returns to the ORP and MMTA to replan since the task planning and allocation turn out to be invalid at runtime. Replanning goes through the same procedure described above but uses an updated T-graph where the edge between the robots and the failed object is invalidated.

2) Greedy version: SEARCH performs an upfront computation to find the allocation for all tasks. If all tasks have feasible motions so that no replanning occurs, few tens of seconds of computation time can be acceptable. If motion planning fails for the allocated tasks, another few tens of seconds for replanning is necessary. Thus, we develop an online greedy version for cases where fast planning is paramount.

Given $\mathcal{O}_R$, GREEDY chooses the robot performing a task instantaneously in each turn. The choice is made based on the penalty used in SEARCH. Unlike SEARCH summing up all penalties up to the current node, GREEDY looks if the switching occurs in this turn. There is no notion of switching

when performing the first task, so the robot closer to the object is chosen. Afterward, the robot that did not perform the relocation task in the previous turn relocates the next object. If only one robot can access the object, it should be chosen. If no robot can access the object, the distances between all pairs of the robot and accessible objects are computed. Then the pair with the shortest distance is chosen.

3) Analysis of the algorithms: We provide proofs for time complexity and completeness of SEARCH and GREEDY. The optimality of the uniform-search is already proven [27].[4] Obviously, the time complexity of SEARCH is exponential as it uses the uniform-cost search.

Theorem 4.1. GREEDY runs in polynomial time.

Proof. Given $\mathcal{O}_R$, checking if an edge exists between the robot node and the node of the first object in $\mathcal{O}_R$ is be done in $O(|E|) = O(N^2)$ without an adjacency matrix. If both robots can relocate the object, Euclidean distances between two pairs of locations are computed in constant time. If no robot can relocate the object, at most $2N$ computations are necessary to compute the distances between the robots and objects. The edge search and the distance computation repeat at most $|\mathcal{O}_R| \leq N$ times. The overall time complexity is $O((N^2 + 2N)N) = O(N^3)$. □

Theorem 4.2. SEARCH and GREEDY are complete if the T-graph is connected.

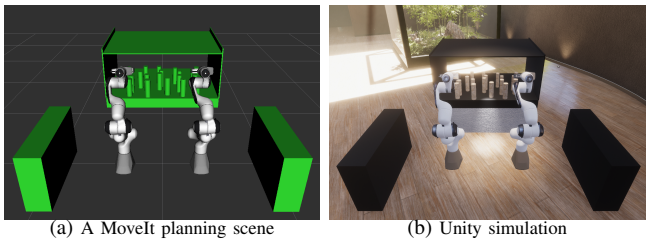
Proof. In Sec. III, we assume that at least one object can be relocated (i.e., Assumption (iii)). The algorithms always return an object and the robot to relocate the object. If the robot fails to relocate the object, replanning returns another solution. In the worst case, all objects are relocated where the target object comes last. □

C. Minimum makespan action sequencing

The most significant advantage of using multiple manipulators is that multiple actions can be done in parallel. As illustrated in Fig. 3, we develop a simple method parallelizing `pick` and `place` actions if the robots manipulate different objects. It inserts `standby` in between two consecutive `pick` and `place` (or vice versa) to let the robots start every action from a predefined pose. Once the action sequence is determined, the method performs two-arm motion planning for the parallelized actions. Even if the actions are not parallelized, two-arm motion planning is done to avoid conflicts between the robots. In order to take the most advantage out of two manipulators, we always assign a target state to the robot that is not being planned to perform `pick`. For SEARCH, we precompute motions for all sequenced actions. With GREEDY, motion planning is done after each robot-task pair is determined.

V. EXPERIMENTS

We show the results from two sets of experiments. The first set evaluates SEARCH and GREEDY without dynamic simulations (Fig. 6a). We measure the objective value, which is the number of switches and the switching rate. Since every solution could have a different number of tasks (so the number of chances for switching varies), showing the rate enables comparisons. We also report other values related to performance like the numbers of relocated objects and replanning, the success rate within a time limit (200 seconds), and task and motion planning (TAMP) time. We have another set of experiments in a dynamic simulation environment (Fig. 6b) implemented using the Unity-ROS integration [28]. In this set, we measure the execution time of the allocated tasks. We expect that the execution time decreases as the number of switches increases.



(a) A MoveIt planning scene (b) Unity simulation

Fig. 6: The environment for experiments. (a) MoveIt used for motion planning and simulations considering kinematics only, (b) Unity-ROS integration for dynamic simulations

Since we are not aware of any comparable methods in the literature, we develop two simple but reasonable methods to compare. DISTANCE allocates tasks according to the Euclidean distance between objects and robots if both robots can access the object to be relocated. Given $\mathcal{O}_R$, this method checks if the first object is accessible to both robots. If both robots can access the object, the robot whose end-effector is closer to the object is chosen. If only one robot can access the object, that robot is chosen. If no collision-free motion is found for both of the robots, $\mathcal{O}_R$ is updated for replanning. The choice repeats until the target is retrieved. RANDOM works similar to DISTANCE but allocates tasks at random if both robots can access the object to be relocated.

We use RRTConnect [29] for motion planning implemented in the Open Motion Planning Library [30] in MoveIt [31]. The system is with AMD Ryzen 7 2700 3.2GHz with 32G RAM. The dimension of the confined space is 1.1 m × 0.5 m × 0.85 m. The number of objects is up to 20 so the space is cluttered and cramped for the bulky manipulators Franka Emika Panda with 7-DOF.

A. Task and motion planning

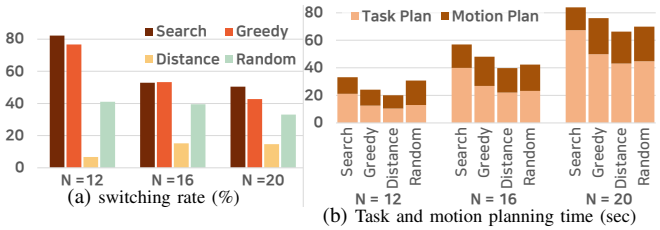
We test the four methods with 30 instances for each $N = 12, 16, 20$ where the object locations are sampled uniformly at random. For fair comparisons, the set of test instances is identical across all methods. We also report the task and motion planning time for each task in order to assess the average time necessary to generate a plan for each task.

As shown in Table I and Fig. 7, our proposed methods notably outperform in finding a solution with more switches. With $N = 12$, the switching rate of SEARCH is 82.2%, which means that the robots can switch more than four times out of five chances. An example allocation with this rate could be $\mathcal{X}_R = (r_1, r_2, r_1, r_2, r_1, r_1)$. For the most cluttered

[4]The optimality is proven only for the search to maximize the number of switches. The optimality of the entire method is not proven since the result from (A) is not proven to be optimal for the original problem relocating objects in the continuous space (the result is optimal given a T-graph but a better T-graph could exist). Also, as mentioned in Footnote 3, we simplify (C) so another optimality gap arises.

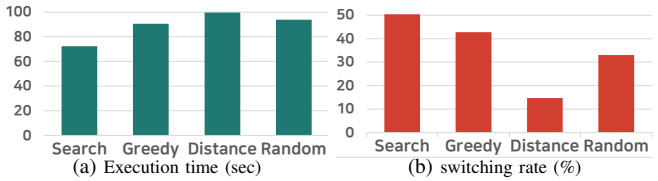
TABLE I: Results of the proposed and compared methods (30 repetitions). Time is measured in seconds. The numbers in parentheses are standard deviation.

Measure	$N = 12$				$N = 16$				$N = 20$			
	SEARCH	GREEDY	DISTANCE	RANDOM	SEARCH	GREEDY	DISTANCE	RANDOM	SEARCH	GREEDY	DISTANCE	RANDOM
#switching	1.13 (0.63)	1.17 (1.02)	0.100 (0.31)	0.633 (0.72)	1.2 (1.03)	1.47 (1.50)	0.400 (0.86)	1.17 (1.51)	1.63 (1.65)	1.87 (2.03)	0.600(1.16)	1.23 (1.33)
Switching rate (%)	82.2 (33.0)	76.7 (43.0)	6.67 (21.7)	41.1 (45.9)	52.9 (37.7)	53.3 (44.1)	15.2 (31.4)	39.5 (42.6)	50.4 (37.4)	42.8 (41.5)	14.7 (30.3)	33.1 (30.9)
#objects relocated	2.40 (0.62)	2.47 (0.82)	2.33 (0.61)	2.73 (1.51)	3.30 (1.29)	3.87 (2.18)	3.63 (2.11)	3.73 (2.21)	4.00 (1.17)	4.63 (2.09)	4.40 (1.69)	4.47 (1.83)
Success rate (%)	100	100	100	100	100	93.3	96.7	93.3	93.3	93.3	96.7	96.7
#replanning	0.333	0.267	0.100	0.733	0.300	0.933	0.567	0.733	0.0667	0.933	0.633	0.800
TAMP time	33.1 (16.3)	24.1 (15.9)	20.0 (9.18)	30.7 (34.7)	57.0 (29.3)	48.0 (32.4)	39.8 (23.6)	42.3 (24.7)	84.0 (26.4)	76.2 (42.3)	66.2 (33.2)	69.9 (37.9)
TAMP time per task	13.4 (4.64)	9.26 (2.95)	8.39 (1.93)	9.59 (4.04)	16.9 (2.81)	12.2 (3.50)	10.8 (2.25)	11.3 (2.65)	21.0 (2.40)	15.9 (2.59)	14.7 (2.13)	15.1 (2.45)



(a) switching rate (%)
(b) Task and motion planning time (sec)

Fig. 7: Important results from the experiment of task and motion planning



(a) Execution time (sec)
(b) switching rate (%)

Fig. 8: Important results from dynamic simulations

TABLE II: Results of dynamic simulation in the Unity-ROS environment (30 repetitions and $N = 20$)

Measure	SEARCH	GREEDY	DISTANCE	RANDOM
#switching	1.63 (1.65)	1.87 (2.03)	0.600 (1.163)	1.23 (1.33)
Switching rate (%)	50.4 (37.4)	42.8 (41.5)	14.7 (30.3)	33.1 (30.9)
#objects relocated	4.00 (1.17)	4.63 (2.09)	4.40 (1.69)	4.47 (1.83)
Success rate (%)	93.3	93.3	96.7	96.7
#replanning	0.0667	0.933	0.633	0.800
TAMP time	84.0 (26.4)	76.2 (42.3)	66.2 (33.2)	69.9 (37.9)
Total execution time	72.4 (24.3)	90.6 (46.4)	99.6 (47.3)	93.9 (45.7)

instance set ($N = 20$), SEARCH reduces the switching rate at most 70.8% and at least 34.3% compared to DISTANCE and RANDOM, respectively. As a consequence, more switches will lead to a more time-efficient task plan for the robots to execute because more actions can be parallelized, resulting in a shorter makespan. GREEDY also shows high switching rates than the compared results, although it does not outperform SEARCH. Notice that the number of switches (not the rate) of GREEDY is larger than SEARCH. This result does not contradict the fact that SEARCH is optimal because the chances of switching are greater with GREEDY as it relocates more objects on average. This explains why we should focus on the switching rate.

Despite the exponential time complexity of SEARCH, its practical planning time (TAMP time in the table) is not prohibitively long. In comparison with others (shown in Fig. 7b), its planning time is at most 21.2% longer (when $N = 20$), which is about 18 seconds longer. One reason for the bounded planning time is that each node in the search tree has a branching factor of two (i.e., has at most two children) since each branching is for each robot performing a task. Also, the number of objects to be relocated does not exceed four in average with SEARCH so the search depth is also bounded. Therefore, SEARCH can run efficiently even with its poor theoretical time complexity. The breakdowns in Fig. 7b indicate that motion planning time is the shortest with SEARCH. The main reason is that SEARCH does not request frequent replannings leading not to generate many motion planning queries, which are computationally expensive. The success rate measures the chance of returning a solution within the time limit (200 seconds). All methods show high success rates. The last measure to note is the number of relocated objects where SEARCH outperforms others in general. GREEDY runs faster than SEARCH, but the difference in time is not impressive since SEARCH does not replan much in our experiments. If the environment is more dynamic so replanning occurs frequently, the difference will become significant.

B. Dynamic simulations

In this set of experiments, we run the relocation plan in the dynamic simulation environment to show the feasibility of running our methods with real robots. The result from 30 instances where $N = 20$ is shown in Table II and Fig. 8. As expected, SEARCH shows the shortest execution time with the highest number of switches. It indicates that SEARCH is capable of generating efficient plans for execution at runtime. The reduction in execution time is at most 27.3% and at least 22.9% compared to DISTANCE and RANDOM, respectively. Although it is not satisfactory, GREEDY shows a slightly better performance. The low reduction in execution time (up to 9%) comes from the myopic decision-making in contrast to the far-sighted SEARCH.

VI. CONCLUSION

We consider the problem of coordinating two manipulators to retrieve a target object from a cluttered and confined space where overhand grasps are not allowed. The objective is to minimize the execution time of the relocation plan through simultaneous executions of tasks by the two robots. We formulate the problem as three subproblems: (A) computing relocation tasks, (B) allocating the tasks to the robots, and (C) sequencing primitive actions and generating collision-free motions for the actions. We present an approach that solves the three subproblems sequentially. We propose two methods where one computes the entire task plan and allocation before execution, and the other finds a relocation task and the allocated robot for the task instantaneously after executing every task. We provide proofs for the time complexities and completeness of the methods. Our extensive experiments show that our methods significantly reduce the execution time up to 27.3% compared to other methods that do not consider coordination of the robots. In the future, we will work on developing a decentralized method for coordination and motion planning. Also, we will run the proposed methods on more than two robots to show their scalability.

REFERENCES

[1] A. H. Qureshi, A. Mousavian, C. Paxton, M. C. Yip, and D. Fox, "Nerp: Neural rearrangement planning for unknown objects," in *Proc. of Robotics: Science and Systems (RSS)*, 2021.

[2] M. S. Saleem and M. Likhachev, "Planning with selective physics-based simulation for manipulation among movable objects," in *Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2020, pp. 6752–6758.

[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 Int. J. of Robotics Research*, vol. 37, no. 13-14, pp. 1775–1795, 2018.

[4] W. Bejjani, M. Leonetti, and M. R. Dogar, "Learning image-based receding horizon planning for manipulation in clutter," *Robotics and Autonomous Systems*, vol. 138, p. 103730, 2021.

[5] 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 *Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2021.

[6] C. Nam, J. Lee, S. H. Cheong, B. Y. Cho, and C. Kim, "Fast and resilient manipulation planning for target retrieval in clutter," in *Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2020.

[7] 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 *Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2020.

[8] J. Ahn, J. Lee, S. H. Cheong, C. Kim, and C. Nam, "An integrated approach for determining objects to be relocated and their goal positions inside clutter for object retrieval," in *Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2021.

[9] S. D. Han, N. M. Stiffler, K. E. Bekris, and J. Yu, "Efficient, high-quality stack rearrangement," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1608–1615, 2018.

[10] J. Lee, Y. Cho, C. Nam, J. Park, and C. Kim, "Efficient obstacle rearrangement for object manipulation tasks in cluttered environments," in *Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2019, pp. 2917–2924.

[11] C. Nam, S. H. Cheong, J. Lee, D. H. Kim, and C. Kim, "Fast and resilient manipulation planning for object retrieval in cluttered and confined environments," *IEEE Trans. on Robotics*, 2021.

[12] I. Umay, B. Fidan, and W. Melek, "An integrated task and motion planning technique for multi-robot-systems," in *Proc. of IEEE Int. Symp. Robotic and Sensors Environments (ROSE)*, 2019, pp. 1–7.

[13] B. Cohen, M. Phillips, and M. Likhachev, "Planning single-arm manipulations with n-arm robots," in *Annual Symp. on Combinatorial Search*, 2015.

[14] R. Shome and K. E. Bekris, "Anytime multi-arm task and motion planning for pick-and-place of individual objects via handoffs," in *Proc. of Int. Symp. on Multi-Robot and Multi-Agent Systems (MRS)*, 2019, pp. 37–43.

[15] ——, "Synchronized multi-arm rearrangement guided by mode graphs with capacity constraints," *arXiv preprint arXiv:2005.09127*, 2020.

[16] D. SepúLveda, R. Fernández, E. Navas, M. Armada, and P. González-De-Santos, "Robotic aubergine harvesting using dual-arm manipulation," *IEEE Access*, vol. 8, pp. 121 889–121 904, 2020.

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

[18] A. Kimmel and K. E. Bekris, "Scheduling pick-and-place tasks for dual-arm manipulators using incremental search on coordination diagrams," in *Proc. of ICAPS Workshop on Planning and Robotics (PlanRob)*, 2016.

[19] J. K. Behrens, K. Stepanova, and R. Babuska, "Simultaneous task allocation and motion scheduling for complex tasks executed by multiple robots," in *Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2020, pp. 11 443–11 449.

[20] J. K. Behrens, R. Lange, and M. Mansouri, "A constraint programming approach to simultaneous task allocation and motion scheduling for industrial dual-arm manipulation tasks," in *Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2019, pp. 8705–8711.

[21] A. M. Kabir, S. Thakar, P. M. Bhatt, R. K. Malhan, P. Rajendran, B. C. Shah, and S. K. Gupta, "Incorporating motion planning feasibility considerations during task-agent assignment to perform complex tasks using mobile manipulators," in *Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2020, pp. 5663–5670.

[22] B. P. Gerkey and M. J. Matarić, "A formal analysis and taxonomy of task allocation in multi-robot systems," *The Int. J. of Robotics Research*, vol. 23, pp. 939–954, 2004.

[23] J. Bruno, E. G. Coffman Jr, and R. Sethi, "Scheduling independent tasks to reduce mean finishing time," *Communications of the ACM*, vol. 17, pp. 382–387, 1974.

[24] J. Lee, C. Nam, J. Park, and C. Kim, "Tree search-based task and motion planning with prehensile and non-prehensile manipulation for obstacle rearrangement in clutter," in *Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2021.

[25] J. Yu, "Rearrangement on lattices with pick-n-swaps: Optimality structures and efficient algorithms," in *Proc. of Robotics: Science and Systems (RSS)*, 2021.

[26] E. D. Demaine, S. P. Fekete, P. Keldenich, H. Meijer, and C. Scheffer, "Coordinated motion planning: Reconfiguring a swarm of labeled robots with bounded stretch," *SIAM J. on Computing*, vol. 48, pp. 1727–1762, 2019.

[27] S. Russell and P. Norvig, "Artificial intelligence: a modern approach," 2002.

[28] U. Technologies, "Unity Robotics Hub," https://github.com/Unity-Technologies/Unity-Robotics-Hub, [Online].

[29] J. J. Kuffner Jr and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA)*, vol. 2, 2000.

[30] I. A. Sucan, M. Moll, and L. E. Kavraki, "The open motion planning library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, 2012.

[31] I. A. Sucan and S. Chitta, "MoveIt," http://moveit.ros.org, [Online].