Computing High-Quality Clutter Removal Solutions for Multiple Robots

Wei N. Tang Shuai D. Han Jingjin Yu

***Abstract*— We investigate the task and motion planning problem of clearing clutter from a workspace with limited ingress/egress access for multiple robots. We call the problem multi-robot clutter removal (MRCR). Targeting practical applications where motion planning is non-trivial but is *not* a bottleneck, we focus on finding high-quality solutions for feasible MRCR instances, which depends on the ability to efficiently compute high-quality object removal sequences. Despite the challenging multi-robot setting, our proposed search algorithms based on A*, dynamic programming, and best-first heuristics all produce solutions for tens of objects that significantly outperform single robot solutions. Realistic simulations with multiple Kuka youBots further confirms the effectiveness of our algorithmic solutions. In contrast, we also show that deciding the optimal object removal sequence for MRCR is computationally intractable.**

I. Introduction

This study expands the investigation of the *clutter removal problem* (CRP) [1] to the case where multiple robots are available. Specifically, we target the setting where several robots operate in a constrained workspace where an exit is shared, and the task is to remove objects that are initially scattered in the workspace. We call this the *multi-robot clutter removal* (MRCR) problem (see Fig. 1 for an example). The shift from a single robot to multiple robots brings two key challenges. First, the robots must share the free space, especially when they are close to the exit, which negatively impacts the computational efficiency when high-quality plans are sought after. Second, intricate interactions arise when there are more than one robot. Consider the scenario in which one robot is scheduled to grasp an object (say o_1), while o_1 is currently blocking the access to another object o_2. As we plan for a second robot, an optimal plan must account for the possibility that o_2 becomes available after the first robot picks up o_1, even though o_2 is not accessible at the moment.

To address these challenges, in this paper, multiple best-first and near-optimal algorithmic solutions are developed for the computation of high-quality object removal sequences for multiple robots. These solutions are empirically shown to be of high quality when they are compared with linearly scaled single-robot solutions. For example, for 15 objects, solutions computed using dynamic programming use only less than 60% the execution time required for a single robot. This closely match the theoretical lower bound of 50%, which is clearly impossible to achieve due to inevitable robot-robot interaction. In contrast to the single-robot case [1], for typical multi-robot settings, though greedy best-first methods work reasonably well and run faster, they fall significantly behind more optimal methods in terms of solution quality. V-REP [2] based simulation further confirms the removal sequences computed by our proposed methods remain effective when they are integrated into complete solutions.

W. N. Tang, S. D. Han, and J. Yu are with the Department of Computer Science, Rutgers, the State University of New Jersey, Piscataway, NJ, USA. E-Mails: {wei.tang, shuai.han, jingjin.yu} @ rutgers.edu. This work is supported by NSF awards IIS-1734419 and IIS-1845888.

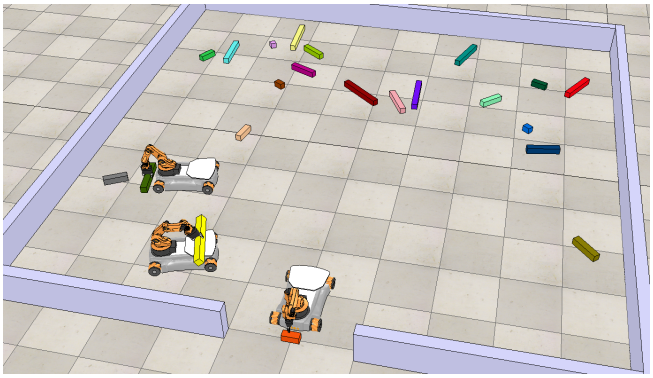


Fig. 1. A snapshot from a simulation run with three Kuka youBots solving a clutter removal task. At the moment, the robot at the bottom is placing an orange object at the drop-off location (i.e., the exit), the robot in the middle is carrying a yellow object to the exit, while the robot at the top is retrieving a dark green object. To solve the problem efficiently, careful coordination among the robots is necessary.

From the practical point of view, our study is motivated by the need of deploying mobile robot systems for disaster response tasks [3], [4]. Clearly, effective and autonomous disaster response requires the close integration of robotics hardware and advanced algorithmic solutions spanning computer vision (e.g., scene understanding), planning, among others. Our work focus at calculating the optimal object picking sequence, which is often called *task planning* in a *task and motion planning* (TaMP) problem [5]–[9].

Generally, solving a TaMP challenge requires discrete combinatorial reasoning and (continuous) motion planning, both of which can often be computationally hard [10]–[15]. Nevertheless, effective algorithmic solutions have been proposed for solving many practical settings. A problem bearing similar combinatorial challenge as MRCR is the problem of *Navigation among Movable Obstacles* (NAMO). When the problem instance has *monotone* property, i.e., a solution exists that requires moving each obstacle once, backtracking techniques may be applied to effectively solve it [16]. Probabilistic complete solutions for non-monotone settings have also been proposed [17]. In solving the single robot CRP, we further note that *dynamic programming* can be integrated into the backtracking process [1] to reduce the search complexity significantly, from $O(n!)$ to $O(n2^n)$.

Object rearrangement, e.g., [18]–[21] is a class of TaMP

problems closely related to MRCR. Most of these studies focus on the effective generation of a rearrangement sequence to reach a desired spatial order of multiple objects. Some formulations [18], [22] in this domain are much like NAMO. Whereas a search based approach is used in [18], symbolic reasoning is applied in [22]. In contrast, [19], [20] put more emphasis on taming the combinatorial explosion caused by the multiple objects involved. As it turns out, the combinatorial aspect is highly non-trivial. For example, rearranging unlabeled objects is already NP-hard if an optimal solution is sought after [20].

Our study builds on efforts aimed at developing integrated TaMP solutions [5]–[9]. However, the current work distinguish itself in that it attacks *optimality issues* under the CRP formulation but for multiple robots. In this aspect, the focus is similar to [1], [20], [23]. We note that this work does not consider other equally important aspects in TaMP including grasp planning [24], [25], high-fidelity motion planning of robot arm [26]–[29], non-prehensile manipulation [30]–[32], or uncertainty rising from perception and motion [33]–[37].

Main Constributions. First, we develop several practical combinatorial algorithms that generate high-quality object removal sequences for multiple robots (the problem has a search space of size $O(n!k^n)$ where n is the number of objects and k is number of robots). These algorithms include novel A* and dynamic programming variants which produce solutions approaching theoretical optimality limits. Second, through realistic simulations using multiple Kuka youBots, we verify that our algorithms remain effective under practical application setups, providing significant savings in execution time as compared with single robot solutions. This validates the premise that a high-quality object removal sequence is a main performance impacting factor in real-world clutter removal tasks. As a minor contribution, we also show that MRCR is NP-hard to optimally solve.

The rest of the manuscript is organized as follows. In Section II, we define MRCR and provide an overview of our MRCR solution pipeline. In Section III, we describe MRCR's structural properties, including NP-hardness if one attempts to optimally solve a part of it, and an intricate dependency between two robots and two objects. In Section IV, we provide combinatorial algorithms for deciding high-quality object removal sequences that also match objects with robots. We present experimental results in Section V and conclude in Section VI.

II. Preliminaries

A. The Multi-Robot Clutter Removal Problem

We consider the setting in which $k \geq 2$ mobile robots $\mathcal{R} = \{r_1, \ldots, r_k\}$ are to clear n rigid objects $\mathcal{O} = \{o_1, \ldots, o_n\}$ *scattered* on the ground, i.e., the objects are isolated from each other. We denote the workspace as $\mathcal{W} \subset \mathbb{R}^2$, and its boundary as $\partial\mathcal{W}$. The robots are initially placed outside of $\mathcal{W}$, and can enter $\mathcal{W}$ through an *exit* on $\partial\mathcal{W}$. Each robot is capable of grasping and transporting a single object at a time. Each object may be picked up once and must be subsequently transported outside of $\mathcal{W}$. An object is considered *cleared* after it is carried by the robot outside the exit and dropped off. The problem studied in this paper is defined as below.

Problem 1. ***Multi-Robot Clutter Removal*** **(MRCR).** *Given $\mathcal{W}, \mathcal{R}, \mathcal{O}$, find a time-optimal plan to clear all objects in $\mathcal{O}$.*

A typical MRCR instance with $k = 3$ is illustrated in Fig. 1. In this study, we assume the given MRCR instance is always feasible and focus on optimizing the solution to minimize the task completion time, also known as the *makespan*.

B. Task and Motion Planning Pipeline

As TaMP is generally computationally intractable, approximation is necessary. We apply a hierarchical approach where a task planner takes charge of the overall planning process. As shown in Fig. 2, a general solution pipeline for MRCR starts with the acquisition of environment setup including object poses, workspace boundaries, and the ingress/egress location. In the current work, we assume such information is given or can be directly retrieved from the simulator (V-REP) backend. In a real-world setting, which we plan to tackle in future work, this step would require perception techniques including object detection, scene understanding, and pose estimation, among others.

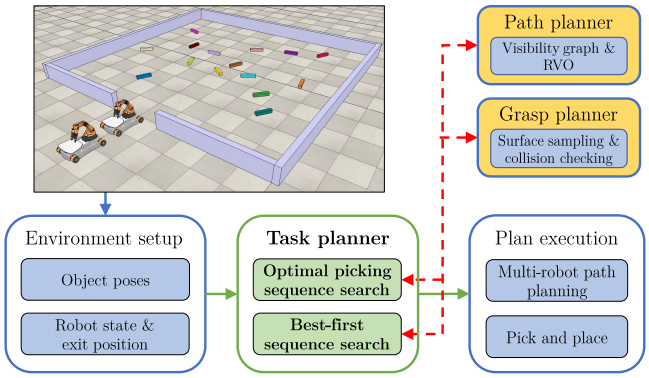
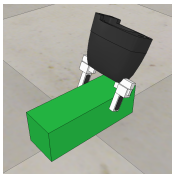


Fig. 2. The full clutter removal pipeline.

With the necessary information acquired about the problem setup, the *task planner* is called next, which handles the task of matching each robot to an ordered sequence of objects to be removed. In general, $k \ll n$ so each robot will be matched with multiple objects. Multiple algorithms are proposed (detailed in Section IV) that trade off between solution optimality and computational efficiency; a specific choice can be decided according to an application. During the task planning step, a *grasp planner* is used to find grasping configurations, and a *multi-robot motion planner* is used to evaluate the time cost for traveling to a certain target object to be removed. By integrating the grasp planner and the multi-robot motion planner into the task planner, we can compute a desirable picking sequence, the corresponding reference robot trajectories for reaching the target objects, and object grasping plan, all at the same time.

As our work does not focus on non-prehensile manipulation, we work with cuboid-like objects. For such objects,

we apply a relatively simple grasp planner: for each object, the planner first finds the top face (i.e., the one with surface normals pointing up) and samples the normals for possible grasps by a 2-finger gripper. As an illustration, in the figure on the right, the simple 2-finger gripper is placed at a sampled grasp pose over an accessible object identified by the planner. Generally, at each phase, many such grasping poses are sampled.



Given the robots' current configurations and the target objects' poses, the task for the multi-robot motion planner is to plan time-efficient, collision-free trajectories for the robots to retrieve the objects from the workspace. Unlike CRP where only a single robot appears in the scene, in MRCR, we must avoid robot-robot collisions while ensuring that the resulting paths have short makespan. In this work, we experimented with both a dRRT* [38] based planner and a planner that combines visibility graph (VG) [39] and Reciprocal Velocity Obstacles (RVO) [40]. dRRT* worked for two robots while VG-RVO worked well for two or more robots. In the two-robot case, we did not observe significant optimality difference between the dRRT* solution and the VG-RVO solution. As such, we only provide evaluation results on VG-RVO. Other multi-robot motion planners can also be used.

We note that the pipeline can be made resolution complete and asymptotically optimal for solving MRCR, e.g., by using an optimal sampling-based motion planner within the task planner. However, the approach is unlikely to be scalable.

C. Special Cases and Problem Extensions

In the study of the single robot version of MRCR [1] (i.e., $k = 1$), we have explored additional cases including multiple exits, static obstacles within the workspace, and different object placements. From that study, we have observed that internal static obstacles do not adversely affect algorithm running time holding object density unchanged. Also, cases with multiple exits and axis-aligned or overlapping objects can be handled with proper techniques and are generally simpler to solve. In the multi-robot scenario, we admit that these variations and extensions may make the motion planning part harder to solve, but they do not add much more complexity to the task planner. Therefore, in our study of MRCR, we focus on the most challenging single-exit case without considering static internal workspace obstacles, and mainly consider cases where objects are randomly placed without overlapping.

III. Structure and Hardness of MRCR

A. Unique Structural Properties of MRCR

For an MRCR instance with k robots and n objects, in the worst case, there are $n!k^n$ possible assignments of robots to objects over the time horizon: there are $n!$ possible sequences with which the n objects may be picked up over time; for each such sequence of length n, there are k^n possible ways of assigning the k robots.

Beside the combinatorial explosion induced by multiple robots, there are two more differences that distinguish MRCR from single robot CRP. First, with a single exit of limited width, robots must non-trivially coordinate their movement to avoid collisions. Second, there can be more intricate dependencies between multiple robots and objects that impact solution optimality. Fig. 3 illustrates a simple case that may happen between two robots and two objects. Essentially, when there are multiple robots, *lookahead* (i.e., simulating execution of plan into the future) is necessary to optimize a plan. Our algorithmic solutions (in Section IV) are tailored to address these unique complications induced by MRCR.

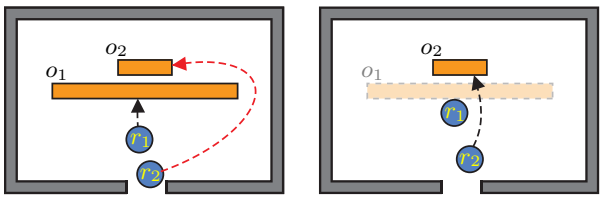


Fig. 3. An example illustrating an intricate dependency that may happen among two objects o_1, o_2 and two robots r_1, r_2. [left] If we do not look into the future, r_2 will need to take a long detour to pick up o_2. [right] However, it is clear that once r_1 lifts o_1, r_2 may readily access o_2.

To conclude this subsection, we mention that for a given environment and object set, if a single robot requires at least time T to complete the object removal task, then for k robots, the minimum possible makespan is no less than $\frac{T}{k}$, i.e.,

Proposition III.1. *For an* MRCR *instance with k robots, if the corresponding* CRP *task for a single robot can be optimally solved in time T, then k robots require at least $\frac{T}{k}$ makespan.*

Clearly, $\frac{T}{k}$ is a theoretical limit that is generally not achievable due to interactions among robots and objects. We will be comparing our algorithmic solutions to this limit.

B. Hardness of Selecting Optimal Object Removal Sequence

In an extended version [41] of [1], CRP is shown to be NP-hard to optimally solve when there is a single exit (the conference version [1] only provided hardness proof for multiple exits). In the proof, it was shown that a planar arrangement of objects (Fig. 4) can be made for which the optimal picking sequence is NP-hard to compute. The structure is constructed from a *monotone planar 3-SAT* (MPSAT) instance [42]. Based on this result, we may establish that the combinatorial part of MRCR is also hard to solve.

Fig. 4. A sketched illustration of the object arrangement used in proving that CRP is NP-hard to solve optimally.

Theorem III.1. *Deciding an optimal object removal sequence is NP-hard for* MRCR.

Proof. The reduction from MPSAT to optimal CRP is fairly complex [41]. However, for this proof, we only need the

fact that the colored pieces can be only removed in a mostly sequential manner. Moreover, an optimal removal sequence translates to a solution to the original MPSAT instance. For our reduction from MPSAT to MRCR with k robots, we essentially make k copies of the structure. For two robots, a possible reduced instance is given in Fig. 5.

Fig. 5. A sketched illustration of the object arrangement used in proving that MRCR is NP-hard to solve optimally.

In the figure, two of the reduced structures are placed on the left and right ends of the environment and there is a single exit in the middle. If this setup is to be solved with a single robot, it can be partitioned into a left task and a right task, each of which take the same amount of minimum time to complete, say T. The minimum makespan for the full problem is then $2T$ which is NP-hard to compute by [41]. If we use two robots, then the problem can be solved with a makespan of T by asking each robot to take care of one side. On the other hand, given the sequential nature in solving the individual side, having two robots working on the same side (e.g., the left one) cannot reduce the required makespan. As such, the MRCR problem is NP-hard to solve as well. The generalization to arbitrary k is straightforward. □

IV. Computing High-Quality Object Removal Sequence and Robot Assignments

As mentioned, unlike the single robot setting [1], MRCR has a unique, more complex structure that makes it much more challenging computationally. To begin to address the challenge, we first provide a description of the system state for tracking progress in a search algorithm.

Problem State Discretization. While there are many possible ways of doing this, we discretize the continuous problem at time instances when there is at least one robot at the exit (the drop-off location) and ready to start retrieving an object. With this definition, a state is essentially a crucial time point when a decision must be made in terms of which objects the free robots are going to retrieve next. We denote a state as $\{R, O_1, O_2, \ldots, O_k, O_{\text{NA}}, \mathcal{S}\}$ with R being the set of robots currently idle, O_j $(1 \leq j \leq k)$ being the sequence of objects that are already assigned to robot r_j and are (or being) retrieved by r_j. $O_{\text{NA}} \coloneqq \mathcal{O} \setminus \bigcup_{1 \leq j \leq k} O_j$ is the set of objects not yet assigned to any robot. $\mathcal{S}$ contains the low-level motion planning elements.

A Basic Search Process. Beginning from the start state where $O_{\text{NA}} = \mathcal{O}$, $R = \mathcal{R}$ and all robots are at the entrance, a discrete search can be carried out to find a picking sequence. To proceed to a next state, we assign each available robot in R an object from O_{NA}. Here, the maximum number of possible assignments is $|R|!\binom{|O_{\text{NA}}|}{|R|}$. For each possible assignment, a low-level grasp and motion planning problem is tackled, which involves two steps: *(i)* for each object assigned to retrieve, the grasp planner is called to provide a feasible configuration for the designated robot to grasp the object; *(ii)* the multi-robot path planner is called to plan the paths for the robots to move their respective grasping configuration, grasp the objects, and then move back to the entrance to drop-off. At the end of the low-level search, a robot drops off an object and is ready to start to retrieve another object. Now, the search process reaches a new state by our definition. This searching process reaches a goal state when all the objects are retrieved from the workspace.

An illustration of a search tree section is provided in Fig. 6. At the top left state, robot r_2 is on its way retrieving object o_2, while robot r_1 just finished dropping off an object and is ready to start to retrieve the next one. Since $O_{\text{NA}} = \{o_1, o_3\}$ and both objects are reachable, the current state generates two new states by assigning o_1 or o_3 to r_1. Since o_3 is close to the exit, if we assign o_3 to r_1, at the next state (the one in the middle of Fig. 6), r_1 finishes retrieving o_3 before r_2 reaches the exit, and immediately starts to retrieve o_1. The search finishes as both robots return to the exit and all objects are removed from the workspace.

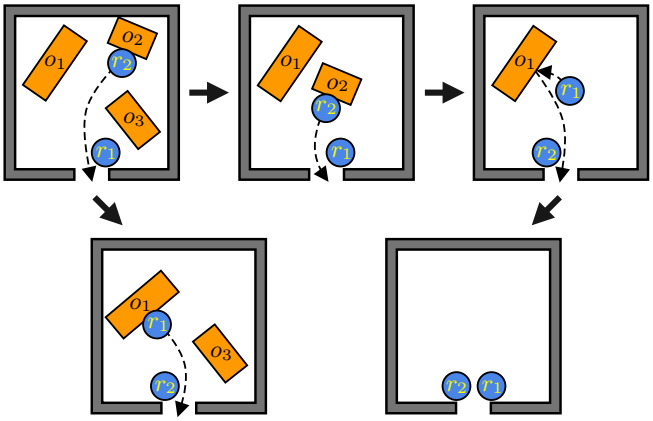


Fig. 6. Illustration of a discrete search tree section. There are totally five discrete search states in this figure.

When performing an object assignment during the search process, not all assignments seem to provide feasible low-level grasp and motion planning problems due to two reasons. First, some objects might be surrounded by the others, thus the grasp planner cannot find a collision-free grasping configuration. Second, even with a valid grasping configuration, an object might be behind some other ones and is not reachable. That is, there does not exist a collision-free path from the robot's current configuration to the designated object. As discussed in Section III-A, simply ignoring these seemingly infeasible assignments could significantly increase the solution makespan. In this work, we use a *lookahead* method to accurately determine whether an assignment is feasible.

With Lookahead. To address the optimality-impacting interactions among multiple robots and objects, at each state, lookahead is performed via simulating the execution of robot actions for robots that are already assigned. For example, for the case from Fig. 3, object o_1 is assigned to robot r_1. As the assignment to r_2 is being decided, we also consider

a future setting where o_1 is already picked up by r_1 and thus no longer blocking o_2. Similar procedure can be carried out for k robots in a recursive manner: as an assignment is being decided, we virtually remove the objects that are immediately reachable, and then see if such a removal makes the previously unreachable objects reachable. As we will show in Section V, algorithms with lookahead always provide more optimal solutions than the ones without lookahead.

Based on the state discretization and the search structure, we propose several algorithms to solve the problem near-optimally, which are introduced in the next few sub-sections. Note that we describe the algorithms as near-optimal due to the near-optimality of the motion planner.

A. Near-Optimal A* Search with an Admissible Heuristic

With the definition of the system state and the established tree structure, the A* framework can be applied to find a near-optimal solution. For estimating the cost-to-goal, we propose an admissible heuristic which calculates an underestimated makespan, i.e., the minimum distance that the robots must travel to retrieve all remaining objects over the product of the number of robots k and the maximum speed of a robot. Given a search state s, we denote o_j as the object robot r_j is currently retrieving. The heuristic value for s is calculated as

$$H(s) = \frac{\sum_{1\leq j\leq k} d(o_j) + \sum_{o_i \in O_{\text{NA}}} d(o_i)}{k v_{\max}},$$

where $v_{\max}$ denotes the maximum speed of the robots, and the function d returns the shortest distance for a robot to retrieve an object. For the former part of the numerator, $d(o_j)$ calculates the remaining straight line distance for robot r_j to finish retrieving o_j; for the latter part, $d(o_i)$ is two times the Euclidean distance between o_i and the drop-off location. Note that since the heuristic ignores robot interaction and acceleration, its value never exceeds the true cost-to-goal. In a real system with robot's specifications available, the heuristic may use more accurate metrics rather than distance over maximum speed, to account for robot dynamics.

As shown in Section V, the A* algorithm generates high-quality solutions for all test cases. However, this method only scales up to around eleven robots. To improve the scalability while maintain a high level of solution optimality, we propose an approximate dynamic programming (DP) algorithm.

B. Approximate Dynamic Programming

The dynamic programming (DP) recursion is based on the assumption that the near-optimal solution of the entire problem can be built on top of the near-optimality of its sub-problems. A sub-problem of `MRCR` is to simply retrieve a subset of objects $O_{\text{sub}} \subseteq \mathcal{O}$, while ignoring the other objects for both task completion and collision check. Let $C(O_{\text{sub}})$ denote the near-optimal makespan of clearing all objects in O_{sub}, the DP framework for calculating a near-optimal makespan for the entire problem is demonstrated in Alg. 1.

Algorithm 1: Using DP to get a near-optimal makespan.

```
1 C = {∅ : 0}
2 for 1 ≤ m ≤ n do
3     for O_sub ∈ all m-subsets of O do
4         C(O_sub) =
            min_{o_i ∈ O_sub} { min_{1≤j≤k} {C(O_sub \ {o_i}) + c_ij} }
5     end
6 end
7 return C(O)
```

We start from the base case where no objects need to be picked up and gradually increase the number of objects m in O_{sub} (line 2). For each $1 \leq m \leq n$, we iterate through all possible O_{sub} (line 3) and use the recursive function in line 4 to calculate a near-optimal makespan to remove all objects in this object subset. In the recursive function, c_{ij} is the additional cost of using robot r_j to remove object o_i as compared to the makespan of removing all objects in $O_{\text{sub}} \setminus \{o_i\}$. Here, $C(O_{\text{sub}} \setminus \{o_i\})$ is already calculated in the $(m-1)$-th iteration, and c_{ij} can be computed by calling the multi-robot motion planner. It is straightforward that with proper bookkeeping, such a DP structure also provides the near-optimal task and motion plan associated with the near-optimal makespan.

The time complexity of the DP approach is as follows. Suppose $|O_{\text{sub}}| = m$, there are $\binom{n}{m}$ possible O_{sub} and for each, computing the recursion requires a cost of $O(kn)$. This yields a total computational cost of

$$O(kn)\left[\binom{n}{0} + \binom{n}{1} + \cdots + \binom{n}{n-1}\right] = O(kn2^n),$$

which grows much slower than the naive $O(n!k^n)$.

C. Greedy Best-First Search

To further improve the scalability of the discrete search-based method, we have developed a greedy best-first search algorithm to reduce the number of state visited in the entire search process. During the object assignment stage, for each robot, the greedy algorithm always selects the closest available object as its next target object.

D. Monte Carlo Tree Search

Trying to find a balance between scalability and optimality, we also implemented a search algorithm based on *Monte Carlo tree search* (MCTS) [43], [44]. Here in this sub-section, we first provide a brief review of the MCTS framework and then focus on explaining how we adapt it for `MRCR`.

MCTS is a search algorithm that expands the search tree based on the analysis of the most promising states. The method's essential components are *selection*, *expansion*, *simulation*, and *back-propagation*. In a basic iteration of MCTS, first, the selection stage picks a search node based on a selection function. Then, the expansion stage creates child nodes of the selected node. After that, the *reward values* of the new child nodes are determined by a simulation from the node to an end state. Finally, the back-propagation stage updates the tree to prepare for the next selection stage.

In our implementation, the selection phase uses an Upper Confidence Bound (UCB) formula $\bar{T} + \sqrt{\frac{\ln N}{n}}$ to select the next node to explore. Here, $\bar{T}$ is the average simulated child node makespan, n is the number of times the node is expanded, and N is the number of parent expansions. As a node is expanded, we visit all child nodes and estimate their makespan as follows: for high-level decision making, we use the greedy best-first strategy; for low-level execution time prediction, instead of calling the motion planner, we simply use the picking distance over robot speed, and ignore interactions between robots. The reward of each simulation is back-propagated to inform further expansion. Every time after we expanded three levels of the search tree, the most explored node is chosen as the next action.

V. Experimental Studies

We evaluate the performance of the proposed near-optimal and heuristic search algorithms under varied object density and drop-off location setups. The algorithms were implemented in C++ and executed on a quad-core Intel CPU at 3.3GHz with 32GB RAM. A video of simulated Kuka youBots carrying out the tasks is provided that supplements the evaluation described in this section.

The testing environments we used for evaluation can be categorized into two general types: one with objects sampled close to each other in a *clutter*, and the other one with objects *scattered* inside the workspace. An illustration of these two typical environments are shown in Fig. 7. For evaluation, we use disc robots with identical radius; some randomly selected solutions are then executed in the V-REP simulator [2] with an accurate Kuka youBot model, and the execution results are consistent with the evaluation. We assume that the picking and the unloading time are the same and equal the time it takes for a robot to travel half of the square workspace side length at maximum speed. This adjustable picking/unloading time is included in the total execution time; our choice is based on our actual experience working with Kuka youBots.

For multi-robot motion planning, in the evaluation, a shortest path connecting a robot to the assigned object is computed using visibility graph [39] which treats objects as static obstacles but does not consider interactions among the robots. Then, to generate feasible robot trajectory, collisions among robots are resolved using RVO [40].

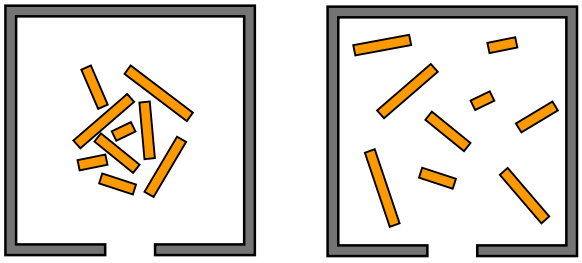


Fig. 7. Two different categories of test environments. [left] A cluttered setting. [right] Objects are more dispersed in the environment.

Our first set of experiments evaluates the performance of all proposed algorithms using both cluttered and scattered setups (Fig. 7), with the number of robots $k = 2$ and the number of objects n ranging from 5 to 15. Fig. 8 shows the result on optimality ratio and computation time. The optimality ratio is the solution makespan of the proposed algorithms divided by the optimal makespan of *single* robot clutter removal for the same setup. The results show that for all test cases, the near-optimal (A* and DP based) algorithms achieve an optimality ratio as low as 0.6, which is close to the underestimated theoretical lower bound 0.5. As the number of objects increases, the difference in optimality ratio between near-optimal algorithms and heuristic search algorithms can become as large as 5%. In terms of scalability, the A* search method scales up to about 11 objects with the computation time limited under 400 seconds, while DP can handle 15 objects. The heuristic search methods can solve a problem with 15 objects in under 30 seconds.

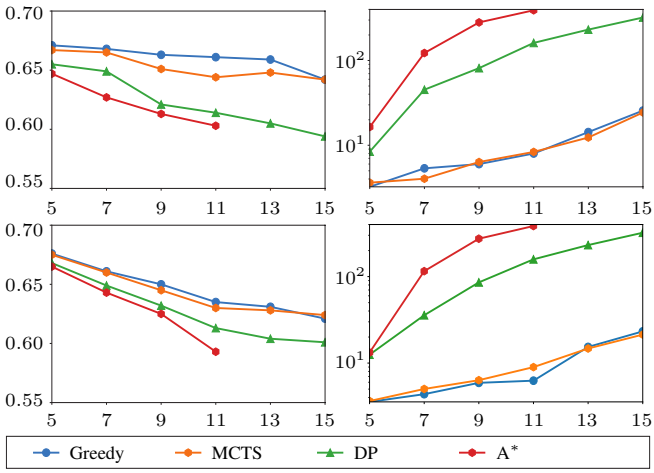


Fig. 8. Optimality ratio and computation time comparison of all the removal sequence search methods for MRCR with two robots. The x axis on all figures are the number of objects in a given environment. The y axes for the figures on the left refer to optimality ratio as compared with the single-robot case. The y axes for the figures on the right are the computation time in seconds. [top] Settings where objects are more cluttered. [bottom] Settings where objects are more scattered.

With a closer look, we can observe that, when the cluttered case is compared with the scattered case, A* and DP generally does slightly better in the former, except at one or two outlier points. On the other hand, the greedier methods show an opposite trend and do better in the scattered case. By examining the object picking sequences generated in these settings, it appears that the difference can be explained as follows. From the earlier study of CRP [1], we know that greedy methods work well for a single robot, producing solutions generally indistinguishable from the optimal. As we move to two robots, for the scattered case, greedy methods are likely to cause the robots to pick objects in different part of the workspace, essentially running a single robot greedy solution with some minor coordination. On the other hand, since the cluttered case is harder than the scattered one, there are generally more opportunities for multiple robots to optimize the picking sequence. However, greedy methods will not exploit such optimization as much given its short

horizon. Due to the closeness of the objects to be removed, the greedy method actually spend more time coordinating the robots' motion.

In the second set of experiments, we evaluate the heuristic search algorithms (i.e., greedy search and MCTS) in the scattered environment as the number of robots ranging from 1 to 5 and the number of objects ranging from 10 to 25. Since the number of robots is increased, the exit width is also increased so that it potentially allows two robots to unload at the same time, to avoid making the exit an artificial bottleneck. The experimental result is summarized in Fig. 9. As we can observe, first, as expected, with a larger exit, the two-robot case does better than the previous experiment. Second, whereas adding more robots shortens the makespan, the amount of gain is quickly diminishing (it does so even faster if we use a narrower exit, which we have attempted). Nevertheless, with five robots, the greedy methods can readily handle 25 objects and does so about 2.5 times faster than if a single robot is used. We note that the setup is chosen to be somewhat constrained; if multiple exits are available, additional execution time speedup can be expected.

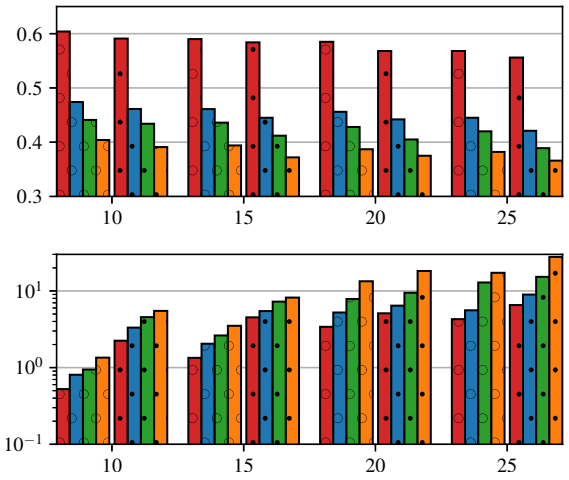


Fig. 9. Performance of greedy and MCTS methods for 2-5 robots and 10-25 objects. The x axes for both figures denote the number of objects. The top figure is the makespan as compared with a single robot case. The bottom figure are the computation time in seconds. For each number of objects, there are eight bars. The left (resp., right) four bars correspond to greedy search (resp., MCTS search) for $2, 3, 4, 5$ robots, from left to right.

In a third set of experiments presented here, we work with a cluttered setting to evaluate the impact of the complex multi-robot multi-object dependency in MRCR, as explained in Section III-A. In the experiment, we run each of the four search methods with lookahead (Section IV) enabled or disabled (by default lookahead is enabled). The result is plotted in Fig. 10. The solid lines are the same as that from the first plot in Fig. 8. As expected, the interaction contributes positively to optimality, yielding a gain up to about 5%.

Lastly, as mentioned, in V-REP based simulation with accurate Kuka youBot models, the computed object removal sequence and the associated robot assignments returned from our task planner directly carry over; the amount of execution

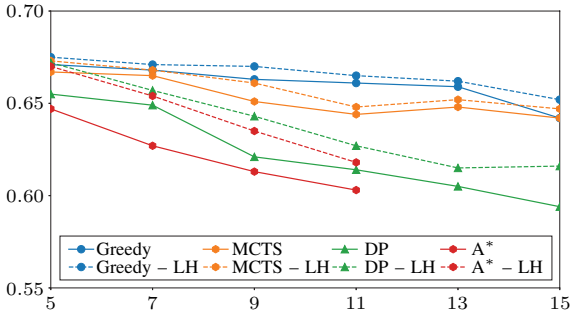


Fig. 10. Comparing methods with and without performing look-ahead to address the multi-robot multi-object dependency. The x axis is the number of objects in a given environment. The y axis is the optimality ratio as compared with the single-robot case. "method – LH" means a given method without look-ahead.

time that is saved is also largely the same as those shown in Fig. 8 and Fig. 9. This further validates the effectiveness of our pipeline design. Selected V-REP simulations can be found in the accompanying video.

VI. Conclusion and Discussions

In this work, we have explored the combinatorial challenge present in the task and motion planning problem of removing clutter from an environment with limited ingress/egress access using multiple robots. We call the formulation the MRCR problem. In contrast to the single robot case [1], for multiple robots, in addition to having a much larger search space due to the choice of more robots, unique constraints also arise that make (near-) optimal task planning more computationally demanding. Toward addressing these combinatorial challenges in MRCR, we have proposed an extendable solution pipeline and within it multiple principled search algorithms (greedy, MCTS, DP, and A*) that balance between scalability and solution optimality. In general, however, these search methods are all reasonably practical and produce significant savings in task execution time when the solutions are compared with those from the optimal single robot setting. For example, using two robots, the execution time can be as little as less than 60% of the optimal single robot case, approaching the theoretical lower bound of 50%. Moreover, when integrated into physics based simulation in realistic settings, the performance of our algorithms hold unchanged. Strengthening the result, we also show that computing the optimal object removal sequence remains NP-hard for multiple robots. In conclusion, we have developed several effective search algorithms for computing high-quality object removal sequence for solving MRCR.

References

[1] W. N. Tang and J. Yu, "Taming combinatorial challenges in clutter removal," in *The 2019 International Symposium on Robotics Research*, 2019.

[2] M. F. E. Rohmer, S. P. N. Singh, "V-rep: a versatile and scalable robot simulation framework," in *Proceedings IEEE/RSJ International Conference on Intelligent Robots & Systems*, 2013.

[3] G. Pratt and J. Manzo, "The darpa robotics challenge [competitions]," *IEEE Robotics & Automation Magazine*, vol. 20, no. 2, pp. 10–12, 2013.

[4] R. R. Murphy, *Disaster robotics*. MIT press, 2014.

[5] S. Cambon, R. Alami, and F. Gravot, "A hybrid approach to intricate motion, manipulation and task planning," *The International Journal of Robotics Research*, vol. 28, no. 1, pp. 104–126, 2009.

[6] E. Plaku and G. D. Hager, "Sampling-based motion and symbolic action planning with geometric and differential constraints," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 5002–5008.

[7] S. Srivastava, E. Fang, L. Riano, R. Chitnis, S. Russell, and P. Abbeel, "Combined task and motion planning through an extensible planner-independent interface layer," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 639–646.

[8] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling, "Ffrob: An efficient heuristic for task and motion planning," in *Algorithmic Foundations of Robotics XI*. Springer, 2015, pp. 179–195.

[9] N. T. Dantam, Z. K. Kingston, S. Chaudhuri, and L. E. Kavraki, "Incremental task and motion planning: A constraint-based approach." in *Robotics: Science and Systems*, 2016, pp. 1–6.

[10] G. Wilfong, "Motion planning in the presence of movable obstacles," *Annals of Mathematics and Artificial Intelligence*, vol. 3, no. 1, pp. 131–150, 1991.

[11] P. C. Chen and Y. K. Hwang, "Practical path planning among movable obstacles," in *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*. IEEE, 1991, pp. 444–449.

[12] E. D. Demaine, M. L. Demaine, and J. O'Rourke, "Pushpush and push-1 are np-hard in 2d," *arXiv preprint cs/0007021*, 2000.

[13] D. Nieuwenhuisen, A. F. van der Stappen, and M. H. Overmars, "An effective framework for path planning amidst movable obstacles," in *Algorithmic Foundation of Robotics VII*. Springer, 2008, pp. 87–102.

[14] 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.

[15] L. Kavraki, J.-C. Latombe, and R. H. Wilson, "On the complexity of assembly partitioning," *Information Processing Letters*, vol. 48, no. 5, pp. 229–235, 1993.

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

[17] J. Van Den Berg, M. Stilman, J. Kuffner, M. Lin, and D. Manocha, "Path planning among movable obstacles: a probabilistically complete approach," in *Algorithmic Foundation of Robotics VIII*. Springer, 2009, pp. 599–614.

[18] J. Ota, "Rearrangement planning of multiple movable objects by a mobile robot," *Advanced Robotics*, vol. 23, no. 1-2, pp. 1–18, 2009.

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

[20] 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.

[21] S. D. Han, S. W. Feng, and J. Yu, "Toward Fast and Optimal Robotic Pick-and-Place on a Moving Conveyor," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 446–453, 2020.

[22] G. Havur, G. Ozbilgin, E. Erdem, and V. Patoglu, "Geometric rearrangement of multiple movable objects on cluttered surfaces: A hybrid reasoning approach," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 445–452.

[23] W. Vega-Brown and N. Roy, "Asymptotically optimal planning under piecewise-analytic constraints," in *The 12th International Workshop on the Algorithmic Foundations of Robotics*, 2016.

[24] M. T. Ciocarlie and P. K. Allen, "Hand posture subspaces for dexterous robotic grasping," *The International Journal of Robotics Research*, vol. 28, no. 7, pp. 851–867, 2009.

[25] J. Bohg, A. Morales, T. Asfour, and D. Kragic, "Data-driven grasp synthesis—a survey," *IEEE Transactions on Robotics*, vol. 30, no. 2, pp. 289–309, 2014.

[26] T. Siméon, J.-P. Laumond, J. Cortés, and A. Sahbani, "Manipulation planning with probabilistic roadmaps," *The International Journal of Robotics Research*, vol. 23, no. 7-8, pp. 729–746, 2004.

[27] D. Berenson, S. Srinivasa, and J. Kuffner, "Task space regions: A framework for pose-constrained manipulation planning," *The International Journal of Robotics Research*, vol. 30, no. 12, pp. 1435–1460, 2011.

[28] 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.

[29] B. Cohen, S. Chitta, and M. Likhachev, "Single-and dual-arm motion planning with heuristic search," *The International Journal of Robotics Research*, vol. 33, no. 2, pp. 305–320, 2014.

[30] A. Cosgun, T. Hermans, V. Emeli, and M. Stilman, "Push planning for object placement on cluttered table surfaces," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*. IEEE, 2011, pp. 4627–4632.

[31] M. Dogar and S. Srinivasa, "A framework for push-grasping in clutter," *Robotics: Science and systems VII*, vol. 1, 2011.

[32] W. C. Agboh and M. R. Dogar, "Pushing fast and slow: Task-adaptive mpc for pushing manipulation under uncertainty," *arXiv preprint arXiv:1805.03005*, 2018.

[33] L. Chang, J. R. Smith, and D. Fox, "Interactive singulation of objects from a pile," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 3875–3882.

[34] H. Van Hoof, O. Kroemer, and J. Peters, "Probabilistic segmentation and targeted exploration of objects in cluttered environments," *IEEE Transactions on Robotics*, vol. 30, no. 5, pp. 1198–1209, 2014.

[35] A. Eitel, N. Hauff, and W. Burgard, "Learning to singulate objects using a push proposal network," *arXiv preprint arXiv:1707.08101*, 2017.

[36] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg, "Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics," *arXiv preprint arXiv:1703.09312*, 2017.

[37] M. Moll, L. Kavraki, J. Rosell, *et al.*, "Randomized physics-based motion planning for grasping in cluttered and uncertain environments," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 712–719, 2018.

[38] R. Shome, K. Solovey, A. Dobson, D. Halperin, and K. E. Bekris, "drrt*: Scalable and informed asymptotically-optimal multi-robot motion planning," *Autonomous Robots*, pp. 1–25, 2019.

[39] 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.

[40] J. Van Den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Robotics research*. Springer, 2011, pp. 3–19.

[41] W. N. Tang and J. Yu, "Taming combinatorial challenges in optimal clutter removal tasks," *arXiv preprint arXiv:1905.13530*, 2019.

[42] M. de Berg and A. Khosravi, "Optimal binary space partitions in the plane," in *International Computing and Combinatorics Conference*. Springer, 2010, pp. 216–225.

[43] R. Coulom, "Efficient selectivity and backup operators in monte-carlo tree search," in *International conference on computers and games*. Springer, 2006, pp. 72–83.

[44] L. Kocsis and C. Szepesvári, "Bandit based monte-carlo planning," in *European conference on machine learning*. Springer, 2006, pp. 282–293.