Taming Combinatorial Challenges in Clutter Removal

Wei N. Tang   Jingjin Yu

Department of Computer Science, Rutgers University
{wei.tang, jingjin.yu}@rutgers.edu

Abstract. We examine an important combinatorial challenge in clearing clutter using a mobile robot equipped with a manipulator, seeking to compute an optimal object removal sequence for minimizing the task completion time, assuming that each object is grasped once and then subsequently removed. On the structural side, we establish that such an optimal sequence can be NP-hard to compute, even when no two objects to be removed have any overlap. Then, we construct asymptotically optimal and heuristic algorithms for clutter removal. Employing dynamic programming, our optimal algorithm scales to 40 objects. On the other hand, for random clutter, fast greedy algorithms tend to produce solutions comparable to these generated by the optimal algorithm.

1 Introduction

We investigate the challenge of clearing clutter with a mobile robot, as an initial step toward the autonomous execution of clean-up tasks, e.g., the handling of the aftermath of earthquakes in urban environments or the tidying up of the daily mess in a kid's room. Specifically, the study focuses on the task and motion planning (`TAMP`) for removing scattered cuboid-like objects with known poses, in a bounded 2D region with exit(s), where each object is grasped once and subsequently removed. We call this the clutter removal problem (`CRP`) and our main goal is to design effective algorithms for computing high quality object removal sequences for minimizing the overall task completion time. A typical setting examined in this paper is illustrated in Fig. 1.

Due to the extremely high complexity of clutter removal as a general `TAMP` task, we explicitly note that the current work has a limited scope on the stated combinatorial challenge and does not consider other important issues such as uncertainties rising from perception or non-prehensile manipulation. Nevertheless, the addressed problem remains relevant when other factors are considered; therefore, the results we provided in this paper has general applicability. Reasoning about the inherent constraints associated with the challenge including objects' shapes, poses (location and orientation), and their relative placement with respect to each other, we are able to establish that finding optimal plans for `CRP` is an intractable task, even when objects assume a planar setting. On the algorithmic side, first, for the single-exit case, we develop an backtracking-based asymptotically optimal algorithm for solving `CRP`, capable of handling 40 objects, which is fairly practical. Then, multiple sub-optimal, best-first type algorithms are developed that perform very well under practical settings. Building on the single-exit solution, we further develop Voronoi-based algorithms for the case of multiple exits that achieve both high solution quality and decent computational efficiency.

Our study is mainly motivated by the need and potential of deploying autonomous robots in disaster response scenarios [19, 25]. The realization of this goal demands the

Fig. 1. A clutter removal scenario addressed in this work, where a KUKA youBot is tasked to grasp and remove all scattered objects in clutter, one at a time, in a room with four static obstacles.

efficient resolution of TAMP challenges [4,6,24]. These TAMP challenges, involving both discrete combinatorial reasoning and (continuous) motion planning, can often be notoriously hard to solve. For example, a class of problems related to this work, *Navigation among Movable Obstacles* (NAMO), are known to be computationally intractable in many forms [21, 30]. Nevertheless, practical algorithms have been proposed that effectively solve the *monotone* case (i.e., where a solution exists that requires moving each obstacle once) via standard backtracking techniques [26]. Probabilistically complete solutions for general settings have also been proposed [28]. The current study emphasizes optimality issues in CRP as a TAMP problem similar to [8, 29]. This contrasts studies with integrated TAMP solutions, e.g., [4,6,24] which do not provide optimality assurances.

Object Rearrangement is another related problem class. Some results in this area, e.g., [9, 22], can be viewed as variations of NAMO. Whereas a search based approach is used in [22], symbolic reasoning is applied in [9] which appears to be more general. In contrast, [8,15] put more emphasis on taming the combinatorial explosion caused by the sheer number of objects involved, with [8] further computing (near) optimal solutions under a metric considering both grasping costs and end-effector travel costs.

Clutter removal is also intimately linked to *(dis)assembly*, where multiple parts need to be put together to yield a product, e.g., [18,20]. The (dis)assembly problem is hard in general [11] and remains so even if the parts are put together two at a time [13]. From the algorithmic perspective, planning of (dis)assembly algorithms is studied in [31], which also proposed measures for evaluating the complexity of the resulting algorithms. Subsequently, a more general motion space approach was developed [7], which proposed a block graph abstraction for representing dependency between components.

The main contributions of the work are three-fold. First, we show that computing an optimal sequence for CRP, as a fundamental discrete combinatorial TAMP challenge independent of the geometrical grasp and motion planning components, is NP-hard. Second, we develop a sampling-based, asymptotically optimal algorithm for CRP. Employing dynamic programming and other techniques, the algorithm is capable of handling up to 40 objects given limited computation time. This is significant because there are $n!$ possible sequences to consider for n objects. Third, we continue to develop fast best-first type algorithms that are empirically shown to compute near-optimal solutions under randomized settings to be expected in real-world scenarios.

The rest of the manuscript is organized as follows. Section 2 describes the clutter removal problem (CRP) studied in this paper. Section 3 then provides some initial analysis regarding feasibility and completeness, and outlines the general algorithmic solution. Section 4 shows that the combinatorial elements of optimal clutter removal is NP-hard even when the objects do not overlap, i.e., in a planar setting. Section 5 proceeds to develop resolution-complete asymptotically optimal algorithms as well as fast best-first algorithms, and shows that there are cases where greedy algorithms yield rather sub-optimal solutions. Section 6 evaluates the performance of the proposed algorithms on computation effort and solution optimality. Section 7 concludes the work.

2 The Clutter Removal Problem

Consider the setting in which n rigid objects $O = \{o_1, \ldots, o_n\}$ are scattered on the ground of a bounded 3D workspace, with o_i representing the known pose (i.e., location and orientation) of the i-th object, $1 \leq i \leq n$. Let $\mathcal{W} \subset \mathbb{R}^2$ be the ground plane of the workspace, which may also contain static obstacles, i.e., inaccessible regions. Let $\partial\mathcal{W}$ be the boundary of $\mathcal{W}$. The workspace can be accessed through *exits* along $\partial\mathcal{W}$ by a mobile robot capable of grasping and transporting objects, one at a time. The task in a clutter removal problem (CRP) is to remove all objects from the workspace. An object is considered cleared after it is carried by the robot outside an exit. Initially, the mobile robot starts at a specific exit. The robot may travel between exits along $\partial\mathcal{W}$. We note that, due to inherent limitations of mobile robots and the placement of the objects (e.g., an object o_j close to another object o_i may prevent the robot from successfully grasping o_i), some objects may be inaccessible to the robot at any given time. Fig. 2 illustrates the top view of a problem instance with static obstacles and three exits.

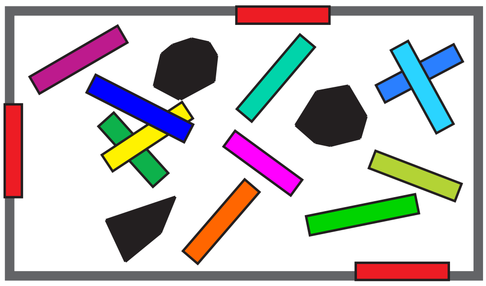


Fig. 2. An example CRP in 2D (viewed from above) where $\mathcal{W}$ is the region within the large rectangle. Three exits are marked red on $\partial\mathcal{W}$. The black polygons inside $\mathcal{W}$ are static obstacles (in additional to $\partial\mathcal{W}$). The rest of the objects, each simplified as a rectangle, are to be removed.

Focusing on the combinatorial problem of computing the optimal clutter removal sequence and given the extreme complexity of the general TAMP problem, this work does not consider non-prehensile manipulation, grasping failures, object pose uncertainty, or multiple grasps per object. Under these assumptions, we work with cuboid-like objects with known poses, for which we may assume that it takes the same amount of time to grasp an object at different poses. Subsequently, solving the CRP formulation optimally reduces to computing a clutter removal sequence to minimize the travel time of the robot. This remains highly challenging because: *(i)* grasp planning and robot base motion planning must be performed continuously to reason about object accessibility and how they can be removed, and *(ii)* given n objects, there are $n!$ possible sequences with which they may be removed; any optimal solution must consider every one of these removal sequences during its computation phase.

3 Preliminary Structural Analysis and Algorithm Design

3.1 Feasibility and Completeness

To solve a `CRP` instance and obtain an object removal sequence, one must first identify at any point the current set of graspable objects. Then, one of the accessible objects must be removed and the process repeats. One of the first issue here is whether an algorithm we design needs to be careful so that an initially feasible problem is not made infeasible. We make the observation that when non-prehensile manipulation is not considered, a feasible `CRP` instance will remain feasible regardless of the object removal order.

Proposition 1. *Adopting a proper (resolution-)complete motion planning algorithm, the clutter removal problem, in the absence of interactions among objects, can be solved with (resolution) completeness guarantees.*

Proof. No explicit requirement is placed on the feasibility of a `CRP` instance. However, we note that, to be able to remove all objects sequentially, there must exist at least one ordering of the n objects with which they *can* be removed one by one. If such an order does exist, since non-prehensile manipulation is not considered in this study, i.e., grasping an object will not make another object accessible to the robot become inaccessible, this implies the existence of a feasible solution for removing objects regardless of the actual object removal order. Therefore, if the initial problem admits a solution, then, at any stage, some object can be removed from the workspace. Subsequently, using a complete [2] (resp., resolution-complete [10, 12, 16]) motion planning algorithm can guarantee the completeness (resp., resolution completeness). □

3.2 Algorithm Structure and Common Routines

With Proposition 1 characterizing the feasibility and completeness for clutter removal, we shift the attention to algorithm design. Since the objects must be grasped and removed one by one, we need subroutines for computing the current set of graspable objects and the shortest distance to reach these objects.

Motion planning for the robot base. Motion planning for the mobile robot (base) is carried out for two purposes: to compute optimal trajectories and to identify objects within the robot's reach. In this work, these are achieved using a variant of the RRT* algorithm [10], augmented with an updating heuristic proposed as part of RRT^X [23].

The cluttered objects (with known poses) are projected over $\mathcal{W}$ (recall that $\mathcal{W} \subset \mathbb{R}^2$ is the 2D projection of the 3D work space onto the ground plane). Treating the projection and the workspace boundary (i.e. $\partial\mathcal{W}$ without the exits) as obstacles, one or more RRT* structures can be computed. An illustration of the RRT* structure (for a single exit) before and after an object removal is shown in Fig. 3. In the RRT* update,

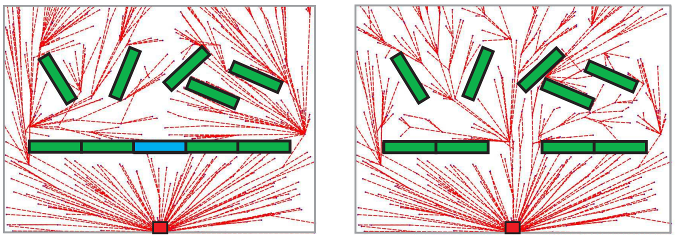
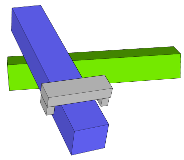


Fig. 3. An illustration of the maintenance of the RRT* structure before and after the removal of an object (the cyan one on the left). The small red rectangles at the bottom indicate the (single) exit, the green rectangles with black borders are the rest of the objects to be removed.

the idea of *cascade rewiring* with a larger radius [23] is adopted, which uses all relevant existing RRT* samples and add new ones only in the area of the newly removed object.

We mention here that for symmetric omnidirectional robot, it is also possible to use the *visibility graph* [17] to compute optimal trajectories when a polygonal approximation of the 2D projection can be obtained, which can be much faster.



Grasp planning. With the RRT* computed for the current environment, reachable objects can be identified. For all these objects, a grasp planner is invoked to compute potential grasps. Per the assumption that the objects to be removed are cuboid-like, a relatively basic grasp planner is applied: for each potential object, the planner first finds a top face (i.e., one with surface normals pointing up) and samples the normals for possible grasps by a 2-finger gripper. In the figure on the right, the gray block illustrates a possible sampled grasp for an accessible object identified by the planner.

General algorithm structure. Based on the sampled grasps returned by the grasp planner and the RRT*, travel distance costs for reaching the grasp by the robot can be computed accordingly. We note that as we increase the resolution of the two sampling process, the costs that are computed will be asymptotically optimal. With these costs, what is left is the computation of an object removal sequence. As such, all the algorithms proposed in this paper share common grasp planning and robot-base motion planning subroutines, and differ on how they use the information returned by the subroutines to compute the object removal sequence, where there are up to $n!$ choices.

4 Hardness of Optimal Clutter Removal

Before constructing full algorithms for `CRP`, we establish that computing the optimal object removal sequence is computationally intractable, even when objects to be removed do not overlap, i.e., the setting is *planar*. In this section, NP-hard is shown for cases with two or more exits. The proof for the single exit case follows similar general logic but is significantly more involved; due to the page limit, details are provided in the extended manuscript [27].

Our proof of the hardness result is via a reduction from *monotone planar* `3-SAT` [1], with the help of some special *gadgets*. In the construction, we assume that the robot is omnidirectional and powerful enough to grasp and transport large objects providing that the object has suitable graspable *handles*.

4.1 Monotone Planar 3-SAT

Monotone planar 3-SAT (MPSAT) is a variation of 3-SAT [5] with three additional restrictions: *(i)* each clause contains exclusively positive literals or exclusively negative literals, *(ii)* the graph connecting clauses to literals has a planar embedding, and *(iii)* the planar embedding can be arranged such that positive clauses and negative clauses reside on two sides of a line connecting all the variables. As an illustration, Fig. 4 provides a planar embedding for the MPSAT instance with variables x_1–x_5 and clauses $c_1 = x_1 \vee x_2, c_2 = x_1 \vee x_3 \vee x_4, c_3 = x_1 \vee x_4 \vee x_5, c_4 = \neg x_1 \vee \neg x_2 \vee \neg x_3$, and $c_5 = \neg x_3 \vee \neg x_4 \vee \neg x_5$. We will be using this example for illustrating the NP-hardness reduction to planar optimal clutter removal.

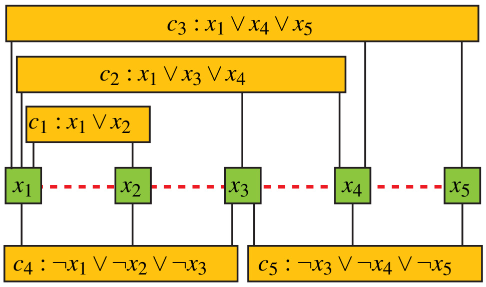


Fig. 4. The planar embedding of the MPSAT instance with variables x_1–x_5 and clauses $c_1 = x_1 \vee x_2, c_2 = x_1 \vee x_3 \vee x_4, c_3 = x_1 \vee x_4 \vee x_5, c_4 = \neg x_1 \vee \neg x_2 \vee \neg x_3$, and $c_5 = \neg x_3 \vee \neg x_4 \vee \neg x_5$.

4.2 The Variable Gadget

For each variable in a given MPSAT instance, we build a gadget for it; Fig. 5(a) illustrates such a gadget for variable x_1. The boundaries of individual objects are marked with black lines. Here, the green object o_g, representing assigning x_1 to be true, can be lifted at either its left most part or its lowest part (when fully exposed), as indicated by the red arrows. The orange object o_o mirrors o_g and represents assigning x_1 to false. After either o_g or o_o is removed, the top purple object o_p can be removed. The partially shown (three) lime objects (call these o_{l1}, o_{l2}, o_{l3}) and (one) yellow object o_y are long rectangles representing connections between the variable gadget and clause gadgets (to be detailed soon in Section 4.3), corresponding to the vertical lines shown in Fig. 4. In this particular case, they are connected to the clause gadgets for c_1–c_4.

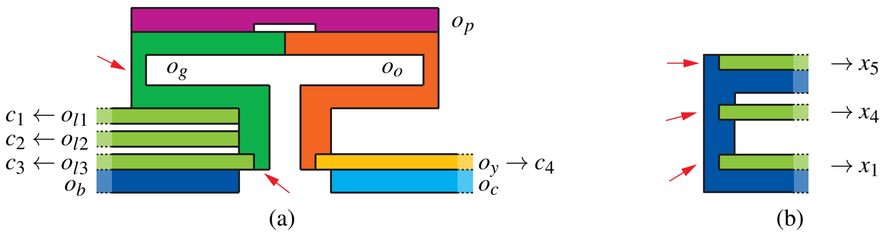


Fig. 5. (a) The variable gadget for x_1, which appears positively in c_1–c_3 and negatively in c_4. (b) The clause gadget for c_3, which connects the variable gadgets for x_1, x_4, and x_5. Notice that the figures are not drawn to scale.

On the positive side, once o_g is removed, in addition to being able to remove o_p, o_{l1}–o_{l3} can be removed by grasping them from their rightmost locations. The blue, partially shown object o_b belongs to the clause gadget for c_3. Alternatively, if o_b is removed, then o_{l3} can be removed from the right side. Following this, o_g can be lifted at its lowest point and removed. o_{l1} and o_{l2} can be subsequently removed as well. On the negative side, because x_1 only appears negatively in c_4, there is a single yellow object o_y connecting o_o to the gadget for c_4, which contains the cyan object o_c. We note that the figure is not drawn to scale. The horizontal span (i.e., the width) of the gadget is much larger than its vertical span. This will be quantified later.

4.3 The Clause Gadget

The clause gadget is fairly simple and the construction for the clause $c_3 = x_1 \vee x_4 \vee x_5$ is shown in Fig. 5(b). If any of the lime objects are removed, then the blue object will have an exposed thin handle (marked by the red arrows) that can be used for lifting and removing the (blue) clause object. A clause gadget will have an *extension* piece for a connecting variable if the rectangle (e.g., the lime piece) connecting the clause gadget and the variable gadget is the lowest one on the variable gadget side. In this example, the clause gadget for c_3 has extensions for x_1 and x_5 (the long horizontal blue extrusions). Comparing with Fig. 4, if a connection between a clause and a variable is the leftmost one for the variable, then there is an extension piece for the gadget for that connection.

4.4 Reducing `MPSAT` to Optimal Clutter Removal

The complete `CRP` instance constructed from the `MPSAT` instance is given in Fig. 6, which is a straightforward assembly of the variable and clause gadgets. The additional items are: *(i)* two extra gray objects at the bottom that can only be lifted and removed after the lowest placed positive and negative clause gadgets are removed, *(ii)* the black "cap" object o_{cap} that surrounds all other objects, and *(iii)* three exits (marked with red hexagons). $\mathcal{W}$ is not shown but can be understood as the region occupied by the construction with some padded space between the construction and $\partial\mathcal{W}$. Object o_{cap} isolates all other objects from left and right exits. The robot starts at the middle exit. The instance is not drawn to scale. The important dimensions are w_1 and w_2 as marked. The distance $2w_1$ is the horizontal span of the two symmetric objects in a variable object. $w_1 + w_2$ is the horizontal distance from the middle exit to a vertical segment of a clause gadget. We assume that all other distances are small when compared with w_1 and w_2, including the vertical span of the instance and all other horizontal distances. Vertical span being minimal means all objects are relatively "long and thin". Given the assumption, the distance between the middle exit and other exits is $w_1 + w_2$. Moreover, in Fig. 6, the lifting points within a dotted rectangle are ε-close to the corresponding exit in the same (dotted) rectangle with ε being very small when compare with w_1 or w_2. Lastly, for an `MPSAT` instance with n variables, the construction ensures $w_2 \gg nw_1$.

The `CRP` instance is feasible: all green and orange objects can be removed first, exposing the connecting rectangular objects, which can be subsequently removed. The purple objects and the black object can also then be removed. Afterward, the clause objects can be lifted and removed. Lastly, the gray objects at the bottom can be removed. On the other hand, computing an optimal solution for the problem is hard. First, we establish that the `CRP` instance requires at least a travel cost of $(2n+4)w_1 + 4w_2$.

Lemma 1. *The* `CRP` *instance requires a minimum possible cost of* $(2n+4)w_1 + 4w_2$.

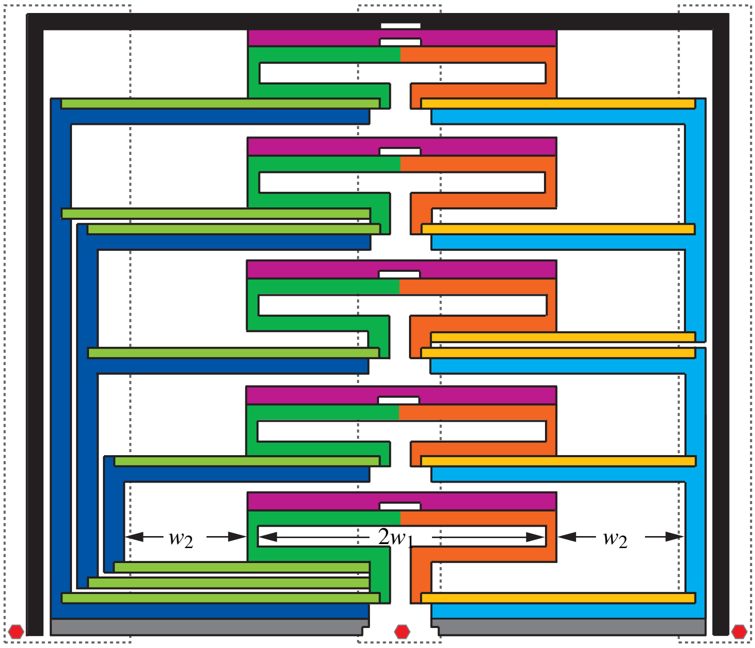


Fig. 6. The CRP instance from the MPSAT instance. In the middle are the five variable gadgets for x_1–x_5, from bottom to top. On the left side are the three (blue) positive clause gadgets for c_1–c_3, from right to left. On the right side are the two (cyan) negative clause gadgets c_4 (lower) and c_5 (upper). The three red hexagons mark the three exits. The robot is initially located at the middle exit. The figure is not drawn to scale.

Proof. First, given a feasible assignment to the MPSAT instance, we show that the CRP instance admits a solution with a total travel distance of $(2n+4)w_1 + 4w_2$. Starting in the middle, if a variable x_i is assigned to be positive (resp., negative), the robot removes the green (resp., orange) object from the i-th variable gadget from the bottom, which incurs a distance cost of $2w_1$ per variable. Then, the associated purple object and lime (resp., yellow) objects can be removed, which incurs minimum extra cost. The step ends with the removal of the black cap object. The total distance cost so far is $2nw_1$. At this point, every clause gadget object has at least one connecting rectangular (lime or yellow) object removed, allowing the clause object to be lifted. The robot then moves to leftmost and clears all blue (positive) clause gadget objects through the leftmost exit, followed by clearing all leftover connecting lime objects. The same is then performed on the right side. Lastly, the robot moves back to the middle to clear everything else. The main cost in this step is incurred by the travel from the middle to the left, then the right, then back, totaling $4(w_1 + w_2)$. The grand total is $(2n+4)w_1 + 4w_2$.

Next, we show the cost is minimal. Beside the above stated removal sequence, there are two alternatives. A first is to start with removing the black cap object from the top (e.g., through the leftmost exit to the top of the black object) . This incurs a cost of $3(w_1 + w_2)$ and the robot will be at either the leftmost exit or the rightmost exit after the removal. Suppose without loss of generality it is the leftmost exit. Then, to remove the cyan clause objects (which can only be removed from the rightmost) and subsequently the right gray object, the robot must travel another $3(w_1 + w_2)$ distance. The total by now is already $6(w_1 + w_2) > (2n+4)w_1 + 4w_2$ because $w_2 \gg nw_1$.

The second alternative is to remove some of the green/orange/purple objects but does not go all the way to the black object before moving to the leftmost or rightmost to work on clause objects. Since the two gray objects can only be removed from the middle after corresponding clause objects are removed, this means that if the robot goes to, e.g., the leftmost and then rightmost and then come back to the center, it will already incur a cost of $4(w_1 + w_2)$, which means the robot can only go to the leftmost and the rightmost once each. This is however insufficient because before the black cap object is removed, a blue or cyan clause object can only be removed from the center exit and therefore, multiple trips to the leftmost or the rightmost are necessary. □

Theorem 1. *Planar optimal clutter removal is NP-hard.*

Proof. The proof of Lemma 1 already shows that a solution to the MPSAT instance leads to a CRP solution of cost $(2n+4)w_1 + 4w_2$; we only need to prove the other direction. Assume the constructed CRP problem has an optimal solution with a total cost of $(2n+4)w_1 + 4w_2$. As has been established, the robot must travel to the leftmost and rightmost side at most once and then eventually return to the middle (with costs $4(w_1 + w_2)$), suggesting that the black cap object must be removed first before any clause objects can be removed. To be able to remove the black cap object, at least n variable objects must be removed, which incur a cost of $2nw_1$. Because this already exhausted the total cost, No more than n variable objects can be removed before the black cap object is. Afterward, we may assume without loss of generality that the robot moves to the leftmost to remove the blue clause objects through the left exit and must remove all of them before traveling back. This implies that the removed green variable objects "satisfies" all the blue (positive) clause objects. Similarly, the cyan (negative) clause objects must also be "satisfied" by the removed orange variable objects. This yields a satisfactory assignment for the MPSAT problem. □

Since it is easy to verify whether a given solution is optimal or not, planar CRP (the simplified combinatorial version without considering complex motion planning) is also in NP. Therefore, this version of optimal clutter removal is NP-complete. A corollary follows that applies to two exits.

Corollary 1. *Planar optimal clutter removal is NP-hard with two exits.*

Proof. We note that the CRP instance can be "bent" in the middle with the left and right sides bending up until they almost meet. Since the initial vertical span of the CRP instance is negligible, this causes the two exits to also be ε-close, i.e., they can be merged into a single exit. This yields a new CRP instance with two exits. The NP-hardness proof continues to work with the updated optimal cost being no more than $(2n+2)w_1 + 2w_2$. □

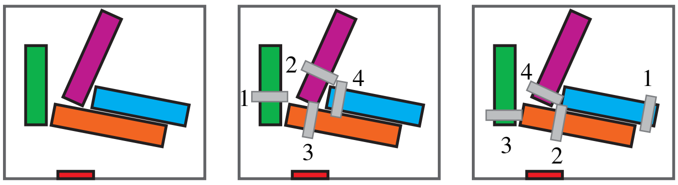
Remark. Though the robot can lift large objects by assumption, lifting a long object in the middle and then going through an exit can potentially lead to issues. In the constructed CRP instance, these are the purple objects and the black cap object; all other objects are lifted from one end (recall that the vertical span of the construction is negligible). The issue can be resolved by breaking an involved object into two equal pieces in the middle which can then be taken away separately (see Fig. 7), without incurring much additional travel cost (the vertical span of the black object is assumed to be very small). By doing this, the robot will always be holding a long object from one end.

Fig. 7. How the purple and black objects can be broken into more manageable pieces, as indicated by the yellow lines, without affecting solution structure. Objects are not drawn to scale.

5 Resolution-Complete Algorithms for CRP

By Proposition 1, applying a resolution-complete algorithm for identifying candidate objects for removal will result in a resolution-complete algorithm for CRP. In this section, we will construct several resolution-complete algorithms realizing varying levels of optimality guarantees. With the preparation done in Section 3.2, our construction assumes knowledge of currently graspable objects and the costs of reaching them, and focuses on the selection of clutter removal sequence based on these information.

Before introducing the algorithms, we present an example that illustrates additional structures of CRP in Fig. 8(a), for which a greedy removal sequence is indicated in Fig. 8(b) and an optimal one is given in Fig. 8(c). Assuming similar grasping cost, the travel time used by the greedy approach is about 1.3 times of that used by the optimal sequence. On the other hand, Theorem 1 indicates that optimal removal sequences can be hard to come by. This motivates the construction of both optimal and greedy algorithms for CRP.



(a) (b) (c)

Fig. 8. (a) A CRP instance. (b) A greedy removal sequence, with the gray bar marking the grasping position by a 2-finger gripper (top view). (c) An optimal removal sequence.

5.1 Single Exit: Exhaustive Search with Dynamic Programming

An exhaustive search approach based on backtracking [28] may be applied to derive an algorithm to search for the optimal object removal sequence. The basic idea is straightforward: all possible object removal sequences are examined and the one with the best cost is chosen. In the context of the current study, a search tree is grown and explored in a depth first manner, with each path from the root to a leaf node representing a complete object removal sequence. Since all permutations are examined, this guarantees an optimal solution is found as long as the cost estimate for grasping and transporting each object is accurate. This later part is in turn guaranteed in a resolution-complete manner in this work, because resolution-complete algorithms are used to build the common motion and grasp planning components. Complete exhaustive search is also possible.

A daunting challenge in examining all branches of a search tree with depth n is the nominal time complexity of $O(n!)$, prohibitively expensive for even small n (e.g., $n > 5$). For the CRP problem, a form of *dynamic programming* (DP) may be applied to significantly reduce this complexity as follows. Let $I = \{1, \ldots, n\}$ and $I' \subset I$. Let $J(I')$ denote the optimal cost of removing all objects with indices in I' assuming that objects with indices in $I \backslash I'$ are already removed. Then we have the key DP recursion

$$J(I') = \min_{i \in I'} \{c_i + J(I' \backslash \{i\})\}, \tag{1}$$

where c_i is the cost of removing o_i assuming that objects with indices in $I \backslash I'$ are already removed. For $|I'| = k$, there are $\binom{n}{k}$ possible I' and for each, computing (1) requires a cost of $O(n)$. This then yields a total computational cost of

$$O(n)\left[\binom{n}{0} + \binom{n}{1} + \ldots + \binom{n}{n-1}\right] = O(n2^n).$$

We note that 2^n grows much slower than $n! \sim \sqrt{2\pi n}(n/e)^n$.

In addition to DP, two additional structural properties of CRP can be exploited to further boost computational efficiency without affecting solution optimality, namely:

- **Reachability**. Objects in clutter naturally create workspace obstacles, limiting the access to other objects and thus reducing effective search branching factor. For example, the purple object in Fig. 8(a) may be inaccessible initially.
- **Object clustering**. It is possible that the objects form *clusters* that are independent in terms of the removal cost, i.e., there may be two or more isolated "piles" of objects. Note that in some cases, a pile may need to be removed first before another can be removed effectively. Clusters can be readily identified by grouping objects that are close to each other.

5.2 Single Exit: Greedy Best-First Search

The NP-Hardness of optimally solving CRP means that exhaustive search cannot run in polynomial time, which prompts the development of greedy approaches: the object with the lowest *local* removal cost is selected and removed; the same process is then recursively applied until all objects are cleared. In addition to the basic greedy best-first strategy which only looks at a single step, two more involved methods are also explored:

- **Multi-step best-first search**. This method computes cost after growing the search tree to some depth $k \geq 1$. The approach, a finite-horizon technique, balances between increased computation and better solution optimality. For example, if k is set to 3, then the case from Fig. 8 can be solved optimally using multi-step best-first search.
- **Monte Carlo Tree Search (MCTS)**. As the core complexity arises from finding a best path along a search tree, another natural choice is Monte Carlo Tree Search (MCTS) [3, 14], which performs limited search tree exploration with varying depth along different tree branches. This can be viewed as a Monte Carlo version of the multi-step best-first search strategy.

In terms of computational complexity, all greedy approaches described here have low polynomial dependency on n, the number of objects.

5.3 Multiple Exits: Extending Exhaustive Search with Dynamic Programming

Algorithms for the single-exit case generalize to multiple exits. Whereas the greedy algorithm requires little change, exhaustive search with DP requires a non-trivial extension. Let $I = \{1, \ldots, n\}$ and $I' \subset I$. Let $J_{ij}(I')$ denote the optimal cost of removing all objects with indices in I' with the robot starting from exit i, and ending at exit j, assuming that objects with indices in $I \backslash I'$ are already removed. Let E denote the set of all exits of the environment. The updated DP recursion is

$$J_{ij}(I') = \min_{e \in E} \min_{k \in I'} \{c_{ej}(k) + J_{ie}(I' \backslash \{k\})\}, \tag{2}$$

where $c_{ij}(k)$ is the cost of starting at exit i and removing o_k from through exit j, assuming that objects with indices in $I \backslash I'$ are already removed. For $|I'| = k$, if we assume the robot always start from a fixed exit, there are $|E|\binom{n}{k}$ possible I' and for each, computing (2) costs $O(|E|n)$. The total is

$$O(|E|n)\left[|E|\binom{n}{0} + |E|\binom{n}{1} + \ldots + |E|\binom{n}{n-1}\right] = O(n2^n|E|^2).$$

5.4 Multiple Exits: Voronoi Partitions

An alternative algorithm for multiple exits may look at the Voronoi partitions induced by $\mathscr{W}$ and the exits, and let the mobile robot remove objects through an exit if the object falls within the corresponding Voronoi region. After the robot finishes working with objects within a Voronoi partition, it moves to the next Voronoi partition. As an example, the Voronoi regions for the three-exit scenario in Fig. 2 is given Fig. 9. Through each exit, the robot will remove around four objects.

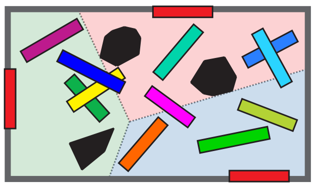


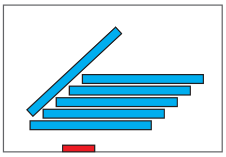
Fig. 9. The Voronoi partition of the example CRP (Fig. 2).

When objects are not heavily entangled, using Voronoi partition plus any single-exit method incurs an additional travel cost equaling at most the length of $\partial\mathscr{W}$. Indeed, simulation study shows that an Voronoi based algorithm performs quite well when compared with direct extensions of single-exit methods. Moreover, Voronoi partitions can be readily obtained based on the shape of $\mathscr{W}$ and the exits' locations, whereas direct extensions of single-exit methods may require additional sensing information to work. That is, to make estimates on which objects can be removed and the associated costs, the robot may need to physically travel through each exit to acquire that information.

6 Experimental Evaluation

A sequence of experiments were designed to evaluate the effectiveness of the algorithms for CRP. Each experiment also provides additional new insights into the structure of CRP. The algorithms were implemented in C++ and executed on a quad-core Intel CPU at 3.3GHz with 32GB RAM. A video of a simulated Kuka youBot carrying out CRP tasks is provided that corroborates the evaluation described in this section.

6.1 Single-Exit Scenarios

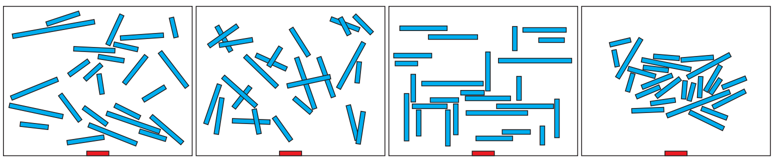


There are cases (e.g., Fig. 8) where a greedy algorithm for CRP can be rather sub-optimal when compared with the exhaustive algorithm. One may ask the natural question of whether such differences actually matter in practice. To evaluate this, we integrated the full solution pipeline with Gazebo using KUKA youBot as the mobile robot (see, e.g., Fig. 1). As a first experiment, we evaluated the execution time of plans obtained by both the exhaustive and greedy algorithm for the scene illustrated on the right. While the plan provided by the greedy algorithm has a total travel distance that is 2.1 times that from the exhaustive one, the ratio of execution time in Gazebo for the two cases is about 1.4 (see the submitted video). The difference in the two ratios (2.1 v.s. 1.4) is due to the time required for grasping/releasing the objects, which is almost the same for both. The example can be readily generalized to yield a family of "bad cases" by stacking the same pattern over and over.

CRP scenarios such as these illustrated in Fig. 6, Fig. 8, and the previous example are highly non-random. It is unlikely to encounter these in practice, where clutter tends to have a more random placement. Our second experiment focuses on different random single-exit clutter removal scenarios with the following possible opposing properties:

S/C: Whether the objects are *scattered* uniformly in the room or *centered* in the room.
R/A: Whether the objects are oriented in *random directions* or they are *axis-aligned*.
O/N: Whether the objects are *overlapping* or not.

Among the eight combinations that were attempted, we select four representative settings as illustrated in Fig. 10. The objective of the second experiment is to evaluate the



(a) (b) (c) (d)

Fig. 10. Four selected settings: (a) **SRN**. (b) **SRO**. (c) **SAN**. (d) **CRN**.

relative computational complexity of different cluttered scenes. For this, both exhaustive search and greedy search are attempted, with the results for exhaustive search (with heuristics) shown in Fig. 11. We note that all test cases are generated randomly with varying object numbers and lengths; for each setting and each number of objects (5-40), 20 test cases are created. A data point corresponds to the average over the 20 cases. For each case, a time limit of 400 seconds is placed. If one of the 20 cases exceeds the limit, no data point for that setting is included. The travel cost is unit-less. From the result the following observations can be made:

- Scenes with overlapping objects (**SRO**, Fig. 10(b)) are easier than scenes with non-overlapping objects (**SRN**, Fig. 10(a)). Axis-aligned cases (**SAN**, Fig. 10(c)) are slightly harder than cases where objects' orientations are more random (**SRN**, **SRO**).
- Centered cases (**CRN**, Fig. 10(d)) are much more challenging (notice the logarithmic scale computation time in Fig. 11). This is due to two reasons: the objects are closer and more objects are graspable, making the branching factor larger in the search tree.

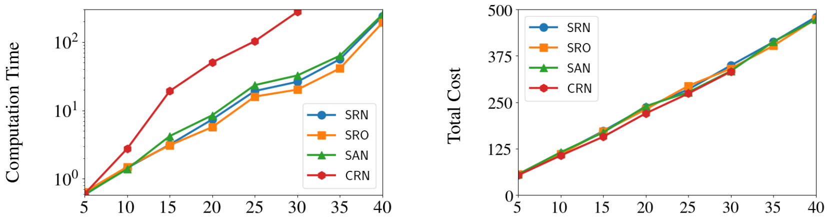


Fig. 11. The computational time and total distance cost using Exhaustive search (with heuristics) on four different settings as illustrated in Fig. 10. The x-axis legends show the number of robots.

We further observe that (from data omitted due to space constraint) somewhat surprisingly, the greedy algorithm computes solutions for all cases with nearly the same total distance costs and does so with much less computation time. To study this further, we fixate on **SRN** as we expect this to be typical and also harder than **SRO**. Multiple algorithms were tested and the result is given in Fig. 12, which clearly shows that the basic greedy approach works quite well in terms of optimality and runs much faster than other methods as the number of objects in clutter increases. Other greedy methods (multi-step, MCTS) take more time but also produce slightly more optimal solutions.

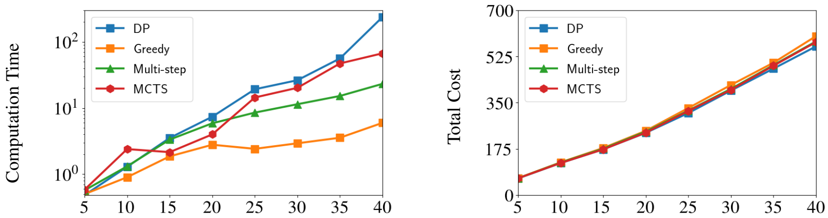
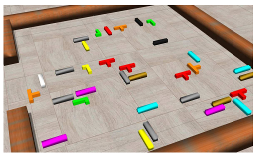


Fig. 12. Computation times and total travel costs from multiple algorithms.



Experiments were also carried out to evaluate the effect of two additional factors: obstacles and more complex shapes. A typical test case of the former is shown in Fig. 1 and a typical case for the later is illustrated on the right, with Tetris-like objects, for which grasp planning becomes more challenging. For both settings, results are highly similar to what is shown in Fig. 12.

6.2 Multi-Exit Setup

For the multi-exit setup, we also attempted a number of experiments. Given the similarity to the single-exit case, little new insights were obtained in running the experiments similar to the single-exit case, except that the branching factor becomes larger due to the availability of more exits, which makes more objects accessible at once. A new set of experiments were also created to evaluate the effectiveness of the Voronoi partition based algorithm, which is compared with exhaustive search and the basic greedy algorithm without the Voronoi heuristic. The test cases are the single-exit **SRN** ones now with three exits selected randomly along the boundary $\partial\mathscr{W}$. The robot may travel along the outside of $\partial\mathscr{W}$ between exits. The experimental result is plotted in Fig. 13.

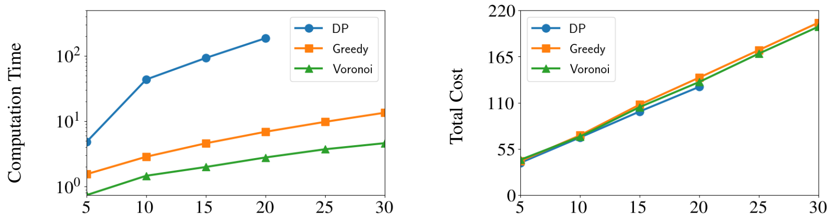


Fig. 13. Multi-exit experiments running exhaustive with dynamic programming, greedy, and Voronoi partition based greedy algorithms.

Again, we observe that the total costs exhibit little difference among the methods. However, the Voronoi based method demonstrates superior scalability, running much faster than the basic greedy algorithm and exhaustive search. At the same time, DP-based exhaustive search can effectively handle over 20 objects efficiently and provides slightly better cost than the greedy methods.

7 Conclusion and Discussion

In this paper, we investigate the clutter removal problem (CRP), performing extensive structural and algorithmic studies for both single- and multi-exit cases. After showing that the problem can be NP-hard to optimally solve, we develop resolution-complete exhaustive search algorithms for CRP. With DP, the algorithms are effective for both single and multiple exits. We also show that typical settings can be efficiently solved using greedy algorithms, which have even better scalability and produce solutions that are fairly close to being optimal. Our algorithms are capable of computing high-quality solutions in seconds for scenes with tens of objects. A key conclusion from the empirical evaluation is that greedy approaches may be applied as a first resort; when there are additional computational resources, longer horizons may also be explored using more exhaustive approaches to further enhance plan optimality.

Acknowledgement. This work is supported by NSF awards IIS-1617744, IIS-1734419, and IIS-1845888. Opinions expressed here do not reflect the views of the sponsor.

References

1. de Berg, M., Khosravi, A.: Optimal binary space partitions in the plane. In: International Computing and Combinatorics Conference, pp. 216–225. Springer (2010)
2. Canny, J.: The complexity of robot motion planning. MIT press (1988)
3. Coulom, R.: Efficient selectivity and backup operators in monte-carlo tree search. In: International conference on computers and games, pp. 72–83. Springer (2006)
4. Dantam, N.T., Kingston, Z.K., Chaudhuri, S., Kavraki, L.E.: Incremental task and motion planning: A constraint-based approach. In: Robotics: Science and Systems, pp. 1–6 (2016)
5. Garey, M.R., Johnson, D.S.: Computers and intractability. WH Freeman, New York (2002)
6. Garrett, C.R., Lozano-Pérez, T., Kaelbling, L.P.: Ffrob: An efficient heuristic for task and motion planning. In: Algorithmic Foundations of Robotics XI, pp. 179–195. Springer (2015)
7. Halperin, D., Latombe, J.C., Wilson, R.H.: A general framework for assembly planning: The motion space approach. Algorithmica **26**(3-4), 577–601 (2000)
8. Han, S.D., Stiffler, N.M., Krontiris, A., Bekris, K.E., Yu, J.: Complexity results and fast methods for optimal tabletop rearrangement with overhand grasps. The International Journal of Robotics Research **37**(13-14), 1775–1795 (2018)

9. Havur, G., Ozbilgin, G., Erdem, E., Patoglu, V.: Geometric rearrangement of multiple movable objects on cluttered surfaces: A hybrid reasoning approach. In: Robotics and Automation (ICRA), 2014 IEEE International Conference on, pp. 445–452. IEEE (2014)
10. Karaman, S., Frazzoli, E.: Sampling-based algorithms for optimal motion planning. The international journal of robotics research **30**(7), 846–894 (2011)
11. Kavraki, L., Latombe, J.C., Wilson, R.H.: On the complexity of assembly partitioning. Information Processing Letters **48**(5), 229–235 (1993)
12. Kavraki, L., Svestka, P., Overmars, M.H.: Probabilistic roadmaps for path planning in high-dimensional configuration spaces, vol. 1994. Unknown Publisher (1994)
13. Kavraki, L.E., Kolountzakis, M.N.: Partitioning a planar assembly into two connected parts is np-complete. Information Processing Letters **55**(3), 159–165 (1995)
14. Kocsis, L., Szepesvári, C.: Bandit based monte-carlo planning. In: European conference on machine learning, pp. 282–293. Springer (2006)
15. Krontiris, A., Bekris, K.E.: Dealing with difficult instances of object rearrangement. In: Robotics: Science and Systems (2015)
16. LaValle, S.M., Kuffner Jr, J.J.: Randomized kinodynamic planning. The international journal of robotics research **20**(5), 378–400 (2001)
17. Lozano-Pérez, T., Wesley, M.A.: An algorithm for planning collision-free paths among polyhedral obstacles. Communications of the ACM **22**(10), 560–570 (1979)
18. Lozano-Perez, T., Wilson, R.H.: Assembly sequencing for arbitrary motions. In: Proceedings IEEE International Conference on Robotics & Automation (ICRA), pp. 527–532 (1993)
19. Murphy, R.R.: Disaster robotics. MIT press (2014)
20. Natarajan, B.K.: On planning assemblies. In: Proceedings of the fourth annual symposium on Computational geometry, pp. 299–308. ACM (1988)
21. Nieuwenhuisen, D., van der Stappen, A.F., Overmars, M.H.: An effective framework for path planning amidst movable obstacles. In: Algorithmic Foundation of Robotics VII, pp. 87–102. Springer (2008)
22. Ota, J.: Rearrangement planning of multiple movable objects by a mobile robot. Advanced Robotics **23**(1-2), 1–18 (2009)
23. Otte, M., Frazzoli, E.: RRT^X: Real-time motion planning/replanning for environments with unpredictable obstacles. In: Algorithmic Foundations of Robotics XI, pp. 461–478. Springer (2015)
24. Plaku, E., Hager, G.D.: Sampling-based motion and symbolic action planning with geometric and differential constraints. In: Robotics and Automation (ICRA), 2010 IEEE International Conference on, pp. 5002–5008. IEEE (2010)
25. Pratt, G., Manzo, J.: The darpa robotics challenge [competitions]. IEEE Robotics & Automation Magazine **20**(2), 10–12 (2013)
26. Stilman, M., Kuffner, J.: Planning among movable obstacles with artificial constraints. The International Journal of Robotics Research **27**(11-12), 1295–1307 (2008)
27. Tang, W.N., Yu, J.: Taming combinatorial challenges in optimal clutter removal tasks. arXiv:1905.13530 (2019)
28. Van Den Berg, J., Stilman, M., Kuffner, J., Lin, M., Manocha, D.: Path planning among movable obstacles: a probabilistically complete approach. In: Algorithmic Foundation of Robotics VIII, pp. 599–614. Springer (2009)
29. Vega-Brown, W., Roy, N.: Asymptotically optimal planning under piecewise-analytic constraints. In: The 12th International Workshop on the Algorithmic Foundations of Robotics (2016)
30. Wilfong, G.: Motion planning in the presence of movable obstacles. Annals of Mathematics and Artificial Intelligence **3**(1), 131–150 (1991)
31. Wilson, R.H., Latombe, J.C.: Geometric reasoning about mechanical assembly. Artificial Intelligence **71**(2), 371–396 (1994)