Efficiently Solving General Rearrangement Tasks: A Fast Extension Primitive for an Incremental Sampling-based Planner

Athanasios Krontiris and Kostas E. Bekris

***Abstract*— Manipulating multiple movable obstacles is a hard problem that involves searching high-dimensional C-spaces. A milestone method for this problem was able to compute solutions for monotone instances. These are problems where every object needs to be transferred at most once to achieve a desired arrangement. The method uses backtracking search to find the order with which objects should be moved. This paper first proposes an approximate but significantly faster alternative for monotone rearrangement instances. The method defines a dependency graph between objects given minimum constraint removal paths (MCR) to transfer each object to its target. From this graph, the approach discovers the order of moving objects by performing topological sorting without backtracking search. The approximation arises from the limitation to consider only MCR paths, which minimize, however, the number of conflicts between objects. To solve non-monotone instances, this primitive is incorporated in a higher-level incremental search algorithm for general rearrangement planning, which operates similar to Bi-RRT. Given a start and a goal object arrangement, tree structures of reachable new arrangements are generated by using the primitive as an expansion procedure. The integrated solution achieves probabilistic completeness for the general non-monotone case and based on simulated experiments it achieves very good success ratios, solution times and path quality relative to alternatives.**

I. Introduction

Manipulation challenges are not always directly solvable but may require a robot to first change the environment as shown in Fig. 1. For instance, a target object may be obstructed by other movable objects. This means that a robot must first rearrange the obstructing bodies.

Rearrangement is a hard problem computationally as it depends combinatorially on the number of objects in the environment. Certain instances, however, are easier to tackle. This was the realization behind a motivating work for manipulation among movable obstacles [1]. The adaptation of this method to rearrangement problems, referred here as "monotone Rearrangement Solver" (mRS), addresses monotone instances, where each object needs to be moved at most once for a solution. The harder non-monotone instances require some objects to be placed in intermediate positions before moved to their target. Furthermore, mRS depends on exhaustive backtracking search to detect the right order with which objects need to be transferred. This search can be slow when many of the potential orderings result in failure as an exponential number of them need to be considered.

Work by the authors has been supported by NSF CCF:13307893 and IIS:1451737. Any conclusions expressed here are of the authors and do not reflect the views of the sponsors.

The authors are with the Computer Science Department of Rutgers, the State University of New Jersey, 110 Frelinghuysen Road, Piscataway, NJ, USA. Email: {kostas.bekris@cs.rutgers.edu}.

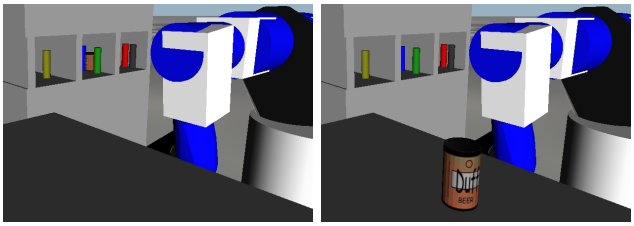

Fig. 1. In this problem, referred to as the "beer challenge" in the experiments, the manipulator has to remove a beer can from a shelf, where the can is obstructed by an object. The can needs to be placed on the table while all the other objects should be placed back in their original positions.

The **first contribution** of this paper is a new method for solving monotone instances by taking advantage of recent algorithmic insights [2], [3]. The proposed method, "fast monotone Rearrangement Solver" (fmRS), avoids backtracking search given the following observation: if the paths for transferring the objects are fixed, then one can easily compute the sequence of moving objects without collisions - if one exists [2]. In particular, the sequence is the output of topological sorting on a "constraint graph". The graph's nodes correspond to objects and the directed edges indicate the ordering between objects, if there are no cycles.

For monotone rearrangement challenges there is always a set of paths, for which the "constraint graph" will have no cycles. Finding this combination of paths, however, requires searching over all possible orders of moving the objects one after the other, which is what the original mRS achieved via backtracking search. This paper proposes fixing the paths of the objects to the "minimum constraint removal" (MCR) paths [3] for constructing the "constraint graph". An MCR path minimizes the number of collisions with other objects. The benefit of such paths is that they introduce the minimum number of constraints in the "constraint graph". While the MCR problem itself is hard, reasonable and efficient approximations are available [3], [4].

If the "constraint graph" has no cycles given the MCR transfer paths, then a topological sorting provides the order with which the objects can be moved without collisions. While the above procedure is incomplete, even for monotone instances, it has a high success ratio and significant computational benefits. In particular, when it works it can often be one or two orders of magnitude faster than the original backtracking search. It also returns faster when it fails to find a solution. This computational advantage can be leveraged by using this approach as a primitive for finding a local rearrangement path between two similar object arrangements in the context of a higher-level planner.

Recent work [5] has shown that hard non-monotone instances can be solved using monotone solvers as a local planner in the context of a higher-level task planner. In their previous work, the authors used a task planner that builds a graph in the space of object arrangements similar to PRM* [6], [7] and used an extension of the backtracking search approach for connecting pairs of object arrangements. Building such a PRM-like roadmap requires that the local planner can frequently connect two randomly sampled arrangements. But this is often not possible given the size of the search space even with a powerful local planner.

The **second contribution** of this paper is that it adapts both the original monotone solver based on backtracking search (mRS) and the new fast monotone solver (fmRS) so that they return a partial solution when no complete solution is found. This involves moving as many objects as possible towards the target arrangement. Furthermore, in the case of a "constraint graph" with cycles, an effort is made for each strongly connected component to discover alternative paths that can lead into a linear ordering without collisions. If this process succeeds for all strongly connected components, then a monotone solution to the original problem is found.

Given the partial formulation of the mRS and fmRS planners, it is possible to use them in a high-level task planner as an expansion step from an initial arrangement instead of a connection between two arrangements. This allows the efficient use of such methods in the context of higher-level search procedures similar to Bi−RRT [8], where two tree data structures originating at the start and goal arrangements are built. The tree nodes correspond to object arrangements and are connected with edges, which are monotone rearrangement paths.

To evaluate solutions for object rearrangement, this paper utilizes a set of benchmarks involving a simulated Yaskawa Motoman manipulator and multiple objects. The benchmarks are general non-monotone rearrangement problems, which include tabletop challenges, as well as more restricted spaces, such as shelves where a single arm needs to reach a target object that is occluded by others. Methodologies considered in this process are:

- For the high-level task planner: A PRM or a Bi−RRT.
- For the local planner: A pick&place primitive for an individual object (p&p), the monotone solver based on backtracking search (mRS) and the new fast monotone solver (fmRS). For the Bi−RRT case, the versions of the solvers that return a partial solution were used.

The local planners were also evaluated individually without being used as primitives in the high-level task planner.

This paper shows that the integration of Bi−RRT with the fast monotone solver is probabilistically complete and is able to solve general non-monotone instances. The experimental evaluation shows that this combination also exhibits the highest success ratio as the number of objects increases as well as the shortest solution times.

II. Rearrangement Problem and Notation

Consider an environment with static obstacles and k movable rigid-body objects $\mathcal{O}$, where each object $o \in \mathcal{O}$ acquires a pose $p \in SE(3)$. An arrangement $\alpha \in \mathcal{A}$ defines k poses $\{p_1, ..., p_k\}$ for the placement of the objects $\mathcal{O}$ in the environment, where $\mathcal{A}$ is the space of all object arrangements. The notation $\alpha[o]$ is used to indicate the pose of an object o given the arrangement α.

Furthermore, a robotic manipulator $\mathcal{M}$ is able to move the objects $\mathcal{O}$ and acquires configurations $q \in \mathcal{Q}$, where $\mathcal{Q}$ is the manipulator's configuration space. A configuration that allows the manipulator to grasp an object o at pose $\alpha[o]$ is denoted as $q(\alpha[o])$. Such a configuration can be computed using an inverse kinematics solver. Then, the manipulator can perform the following two operations [9]:

- **Transit**: The arm is not carrying an object and all the objects are stably positioned in the environment without any force applied to them from the manipulator.
- **Transfer**: The arm is grasping an object and is transferring it inside the environment.

A manipulation path is able to switch from transit to transfer subpaths at transition configurations, where the manipulator is grasping an object, while it is stably resting on a surface.

Rearrangement Problem: Given an initial arrangement $\alpha_I \in \mathcal{A}$ and a final arrangement $\alpha_F \in \mathcal{A}$ of k movable objects $\mathcal{O}$, find a manipulation path π, which allows the manipulator to move the objects from α_I to α_F one at a time in a collision-free manner.

III. Methods

This section first summarizes the existing "monotone Rearrangement Solver" mRS and then introduces the faster but incomplete fmRS. Both methods focus on monotone rearrangement. For both approaches, a variation is proposed, able to return a partial solution. The partial solvers are integrated with a higher-level Bi−RRT planner to achieve a probabilistically complete method for the general case.

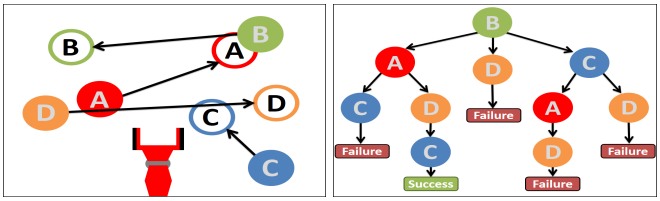

Fig. 2. (left) An example of 2 arrangements for 4 objects (initial: filled color, final: open circles) and a possible linear paths that allow for monotone rearrangement: B, A, D, C. (right) Corresponding backtracking search.

A. Backtracking Search for Monotone Rearrangement

If there is a monotone solution for moving k objects from an initial α_I arrangement to a final α_F arrangement, then there is a solution manipulation path with at most k transfer subpaths, i.e., where each object is transferred at most once (fig. 2 (left)). Between each pair of transfer paths there is a transit path that allows the arm to switch between objects. There is also an initial and final transit path that allows the manipulator to reach the first object and retract after completing the transfer of the last object.

Monotone problems can be solved by searching over all possible orders with which the objects can be moved, as in the (mRS) algorithm [1]. The original algorithm was addressing the challenge of clearing the space from movable obstacles so as to reach a desired target object. For this reason, it searched backwards in time, starting from the path that allowed to grasp the target object, figuring out the obstacles that were obstructing it and then clearing them out of the way. If an explicit current α_I and final α_F arrangement is provided, as in the problem setup for this paper, then the search can be performed forward in time.

Algorithm 1: $\texttt{mRS}(o, q, \mathcal{O}_R, \alpha_I, \alpha_F)$

1 $\pi_{\mathcal{N}} \leftarrow \texttt{TRANSIT}(q, \alpha_I[o], \alpha_I)$;
2 $\pi_{\mathcal{M}} \leftarrow \texttt{TRANSFER}(o, \alpha_I[o], \alpha_F[o], \alpha_I)$;
3 **if** $(\pi_{\mathcal{U}} \leftarrow \{\pi_{\mathcal{N}} \mid \pi_{\mathcal{M}}\})$ *is collision free* **then**
4 $\alpha_I[o] \leftarrow \alpha_F[o]$;
5 **if** $\mathcal{O}_R == \emptyset$ **then**
6 **return** $\pi_{\mathcal{U}}$;
7 **for** *each* $o_r \in \mathcal{O}_R$ **do**
8 $\pi \leftarrow \texttt{mRS}(o_r, q(\alpha_F[o]), \mathcal{O}_R \setminus o_r, \alpha_I, \alpha_F)$;
9 **if** $\pi \neq \emptyset$ **then return** $\{\pi_{\mathcal{U}} \mid \pi\}$;
10 **return** $\emptyset$;

Alg. 1 receives as input the first object o to be transferred between the initial arrangement α_I and the final arrangement α_F. It also receives the initial configuration q of the arm and the set $\mathcal{O}_R$, which represents the objects in the environment that have not yet been moved to their final poses in α_F.

The algorithm finds a transit path $\pi_{\mathcal{N}}$ from the arm configuration q to grasp the object o at its initial pose $\alpha_I[o]$ (line 1) and then transfers it to $\alpha_F[o]$ with the path $\pi_{\mathcal{M}}$ (line 2). These paths must be collision-free, including with all other objects $\mathcal{O} \setminus o$ according to α_I (line 3).

If such collision-free paths exist, then the object is moved (line 4). Once all objects have been moved to their final poses in α_F, then the method terminates (lines 5-6). If there are objects that still have not reached their target pose, the method is called recursively for each one of them as the first to be transferred next (lines 7-8). If a path π is found for the subproblem, then it is composed with paths $\pi_{\mathcal{N}}$ and $\pi_{\mathcal{M}}$ to return a solution (line 9). If all recursive calls to move the remaining objects fail to return a path, then the process returns failure (line 10), as the current branch of the backtracking search failed to give a solution.

The method exhaustively considers all possible object orders to reach α_F given a monotone solution. It is complete for monotone challenges, as long as the procedure for finding the transit and the transfer paths is complete.

B. A Faster Monotone Rearrangement Primitive

Exhaustively searching for all possible orders can result in long solution times and does not scale well with the number of objects in the scene. There is a way, however, to approximate this process, while avoiding backtracking search, by considering the method shown in Alg. 2.

First, for each pair of initial $\alpha_I[o]$ and final $\alpha_F[o]$ poses for an object o, the proposed approach computes the "minimum constraint removal" (MCR) path [3] (lines 1-2). The corresponding routine computes the path with the minimum set of constraints that need to be removed from the problem to find a good quality solution. When computing such a path for object $o \in \mathcal{O}$, the constraints correspond both to the initial and final placements of all other objects in $\mathcal{O} \setminus \{o\}$.

Algorithm 2: $\texttt{fmRS}(q, \mathcal{O}, \alpha_I, \alpha_F)$

1 **for** *each* $o \in \mathcal{O}$ **do**
2 $\Pi[o] \leftarrow \texttt{MCR}(q, \alpha_I[o], \alpha_F[o], \alpha_I[\mathcal{O} \setminus o] \cup \alpha_F[\mathcal{O} \setminus o])$;
3 $\mathcal{G}_c \leftarrow (\mathcal{V} \leftarrow \{\mathcal{O}\}, \mathcal{E} \leftarrow \{\emptyset\})$;
4 **for** *each* $o_i \in \mathcal{V}$ **do**
5 **for** *each* $o_j \in \mathcal{V} \setminus v$ **do**
6 **if** $(\alpha_I[o_i] \in \Pi[o_j])$ **then**
7 $\mathcal{E} \leftarrow \mathcal{E} \cup \{o_j \rightarrow o_i\}$;
8 **if** $(\alpha_F[o_i] \in \Pi[o_j])$ **then**
9 $\mathcal{E} \leftarrow \mathcal{E} \cup \{o_i \rightarrow o_j\}$;
10 **if** *($\mathcal{G}_c$ is DAG)* **then**
11 $\mathcal{L} \leftarrow \texttt{TOPOLOGICAL\_SORT}(\mathcal{G}_c)$;
12 **return** $\bigoplus_{o \in \mathcal{L}} \Pi[o]$;
13 **return** $\emptyset$;

Given such paths for all the objects, it is then possible to compute a "constraint graph" [2] (lines 3-9). This is a directed graph that defines the monotone execution sequence of moving objects $\mathcal{O}$ so as to solve a rearrangement problem.

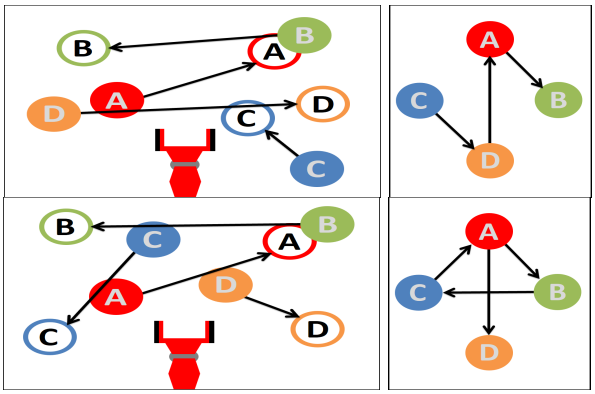

Fig. 3. (left) An example of two arrangements of four objects (Initial: filled disk, final: empty circle). (right) The constraint graph generated by these examples given the MCR paths. Top case: A monotone solution has been found. Bottom case: A cycle arises.

Definition 1 (Constraint graph): The nodes of the constraint graph correspond to objects $o \in \mathcal{O}$ and an edge $(o_i \rightarrow o_j)$ expresses the constraint that o_j needs to be moved before o_i in the execution sequence.

In particular, constraints are defined in the following way:

- If object o_i's initial pose $\alpha_I[o_j]$ collides with the MCR path $\Pi[o_j]$ for object o_j (line 6), then o_i has to move first, i.e., $o_j \rightarrow o_i$ (line 7).
- If object o_i's final pose $\alpha_F[o_i]$ collides with the MCR path $\Pi[o_j]$ for object o_j (line 8), then o_j has to move first, i.e., $o_i \rightarrow o_j$ (line 9).

If there is no cycle in the constraint graph (Fig.3 top), then it is a Directed Acyclic Graph (DAG) (line 10). Then, it is possible to compute an ordering $\mathcal{L}$ of the nodes of $\mathcal{G}$, using topological sort (line 11). For that ordering, the computed MCR paths can be used to solve the monotone problem without the need for search (line 12). If it is not a DAG (Fig.3 bottom), then it is not possible to solve the problem with MCR paths (line 13).

C. Bi-RRT using Monotone Rearrangement for Expansion

The provided algorithms can address only monotone instances. It is easy to generate problems, where the transfer path to the final pose of an object is always blocked by another object's initial pose. In these cases, the algorithm needs to move the blocking object(s) in an intermediate pose. An idea in a previous work of the authors [5] was to use such primitives like the mRS method as local planners within higher-level processes that search the space of object arrangements. The benefit of such monotone solvers is that they can provide a way to connect an arrangement to a larger set of neighboring arrangements relative to a pick and place action, which is the typical low-level primitive for manipulation task planning [10], [11].

The high-level search process can be performed in many different ways, e.g., following a general heuristic search approach, as in related work [10], or in a PRM-like fashion [5], by building a graph of object rearrangements. In the latter case, edges could be constructed by calling the monotone approach to connect pairs of arrangements. A drawback for a PRM-like task planner is that for hard problems, the probability of connection between two arbitrary arrangements using a monotone approach is still low.

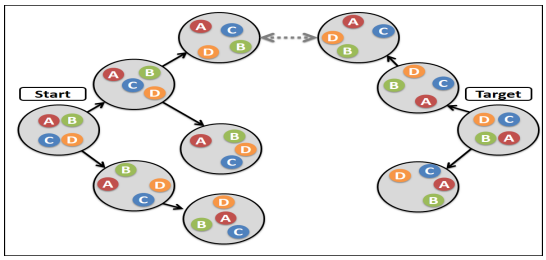

Fig. 4. A bi-directional tree in the space of object arrangements.

It is preferable, if after the approach samples an arrangement, a local path between a pair of arrangements is generated. To achieve this, instead of trying to exactly connect a new sampled arrangement with existing ones in the graph, it is better to extend an existing arrangement towards a new sampled point. This idea is closer to the operation of incremental sampling-based tree planners, such as a Bi-directional RRT (Bi − RRT) [8], which is summarized in Alg. 3 for use in rearrangement planning.

The high-level planner builds a bi-directional tree (as in Fig.4), where nodes correspond to object arrangements. The two trees start with the initial and final arrangements α_I and α_F correspondingly as their only nodes (line 1). Edges correspond to local rearrangement paths that will be computed by a monotone solver, such as mRS. While the problem is not solved (line 2), the method samples new random arrangements α_{rand} (line 3). Then, based on a distance estimate in the space of arrangements, the closest neighboring arrangement α_{near} in the starting tree ($\mathcal{T}_I$) is returned (line 4). In order to compute distances between arrangements, the sum of distances between the placement of objects in the two arrangements is used. A connection is attempted between the neighbor and the random arrangement (lines 5) with a monotone rearrangement search algorithm. Either mRS can be used here as a subroutine (as shown in Alg. 3) or fmRS.

Algorithm 3: $\texttt{Bi} - \texttt{RRT}(q, \mathcal{O}, \alpha_I, \alpha_F)$

1 $\mathcal{T}_I \leftarrow \{\mathcal{V}_I \leftarrow \{\alpha_I\}, \mathcal{E}_I \leftarrow \{\emptyset\}\}$;
  $\mathcal{T}_F \leftarrow \{\mathcal{V}_F \leftarrow \{\alpha_F\}, \mathcal{E}_F \leftarrow \{\emptyset\}\}$;
2 **while** $(\Pi \leftarrow \texttt{FIND\_PATH}(\mathcal{T}_I, \mathcal{T}_F, \alpha_I, \alpha_F)) == \emptyset$ **do**
3 $\alpha_{rand} \leftarrow \texttt{SAMPLE\_ARRANGEMENT}()$;
4 $\alpha_{near} \leftarrow \texttt{CLOSEST}(\mathcal{T}_I, \alpha_{rand})$;
5 $\{\pi, \alpha_{partial}\} \leftarrow \texttt{mRS}(o, q, \mathcal{O}, \alpha_{near}, \alpha_{rand})$;
6 $\mathcal{E}_I \leftarrow \mathcal{E}_I \cup \{(\alpha_{near}, \alpha_{partial}), \pi\}$;
7 $\alpha_{near} \leftarrow \texttt{CLOSEST}(\mathcal{T}_F, \alpha_{partial})$;
8 $\{\pi, \alpha\} \leftarrow \texttt{mRS}(o, q, \mathcal{O}, \alpha_{near}, \alpha_{partial})$;
9 $\mathcal{E}_F \leftarrow \mathcal{E}_F \cup \{(\alpha_{near}, \alpha), \pi\}$;
10 **return** Π;

In the spirit of the original $\texttt{Bi} - \texttt{RRT}$ algorithm, it is not necessary to exactly connect α_{near} with α_{rand}. Instead, it is sufficient if the method makes some progress away from α_{near} in the space of arrangements and a new $\alpha_{partial}$ is linked to the tree (line 6). The closest neighboring arrangement α_{near} is discovered in the other tree ($\mathcal{T}_F$) (line 7) and a similar extension of this tree is also performed (lines 8-9).

By extending a new arrangement from α_{near} and not connect it to α_{rand}, the algorithm is able to frequently generate edges. The first time the two trees connect, the algorithm will return a solution. It is beneficial to consider such a bi-directional solver rather than a single-tree expansion. It may be that the problem is more constrained at the initial arrangement and the tree from the target arrangement will be able to expand easier. A bi-directional tree tends to achieve better and faster coverage of the space.

D. Acquiring Partial Solutions

The mRS and fmRS methods have been defined so that they connect exactly two arrangements. Here, this requirement is relaxed so that these processes can be used as effective extension operations for the high-level $\texttt{Bi} - \texttt{RRT}$ task planner. This means it should be possible to return a best effort partial solution where just some of the objects are moved.

It is easy to make mRS return a partial solution by remembering the maximum depth the algorithm has managed to achieve during the backtracking search. In each iterative call, the algorithm needs to keep track of the maximum depth and the corresponding sequence. At the end of the search process, the algorithm either succeeds to connect the two arrangements or a partial solution corresponding to the best effort to connect the two arrangements is returned together with the resulting arrangement.

There is also a way to get a partial solution from fmRS. For instance, in Fig. 3 (bottom), object D can be moved to its target regardless of the cycle (A, B, C). The proposed partial_fmRS starts from sink nodes of the constraint graph, moves the corresponding objects to their target, removes the nodes from the graph and continues for as long as it is possible to move objects. Furthermore, it tries to decouple cycles whenever possible given that higher priority objects have moved and different MCR paths can be computed. Alg. 4 describes the algorithm, which extends the previous fmRS and operates over the resulting constraint graph $\mathcal{G}_c$.

Algorithm 4: partial_fmRS$(q, \mathcal{G}_c, \mathcal{O}, \alpha_C, \alpha_F)$

1 $\mathcal{SC} \leftarrow$ STRONG_COMPONENTS$(\mathcal{G}_c)$;
2 $\mathcal{G} \leftarrow (\mathcal{V} \leftarrow \{\mathcal{SC}\}, \mathcal{E} \leftarrow \{\emptyset\})$;
3 **for** *each strong component* $w \in \mathcal{V}$ **do**
4 **for** *each strong component* $m \in \mathcal{V} \setminus w$ **do**
5 **if** *(any of* $\alpha_C[m.\mathcal{O}_m] \in w.\Pi$*)* **then**
6 $\mathcal{E} \leftarrow \mathcal{E} \cup \{w \rightarrow m\}$;
7 **if** *(any of* $\alpha_F[m.\mathcal{O}_m] \in w.\Pi$*)* **then**
8 $\mathcal{E} \leftarrow \mathcal{E} \cup \{m \rightarrow w\}$;
9 $\mathcal{L} \leftarrow$ TOPOLOGICAL_SORT$(\mathcal{G})$;
10 $\pi \leftarrow \emptyset$;
11 **for** *each* $w \in \mathcal{L}$ **do**
12 **if** *(*$|w| == 1$*)* **then**
13 $\pi' \leftarrow$ MCR$(q, \alpha_C[w.\mathcal{O}_w], \alpha_F[w.\mathcal{O}_w], \alpha_C[\mathcal{O} \setminus w.\mathcal{O}_w])$;
14 **else**
15 $\alpha'_F[\mathcal{O} \setminus w.\mathcal{O}_w] \leftarrow \alpha_C[\mathcal{O} \setminus w.\mathcal{O}_w]$;
16 $\alpha'_F[w.\mathcal{O}_w] \leftarrow \alpha_F[w.\mathcal{O}_w]$;
17 $\pi' \leftarrow$ fmRS$(q, \mathcal{O}, \alpha_C, \alpha'_F)$;
18 **if** *(*$\pi' \neq \emptyset$*)* **then**
19 $\pi \leftarrow \pi \oplus \pi'$;
20 $\alpha_C[w.\mathcal{O}_w] \leftarrow \alpha_F[w.\mathcal{O}_w]$;
21 **return** π;

Given a constraint graph $\mathcal{G}_c$ with cycles, depth-first search can be used to detect its strongly connected components (line 1). This allows to turn $\mathcal{G}_c$ into a directed acyclic graph (DAG), where each strongly connected component may includes many interdependent objects (line 2), e.g., objects (A,B,C) in Fig. 3 (bottom). It is then possible to apply topological sorting over the resulting DAG (lines 3-9). Then, each node w of the graph $\mathcal{G}$ based on the topological order $\mathcal{L}$ can be one of the following:

An individual node: When $|w| = 1$ (line 12), a new MCR path is computed for the corresponding object (line 13). This path respects the current poses of all objects. Objects with higher priority in the DAG's topological order can be in either their initial or target poses, depending on whether the partial solution managed to transfer them or not. All the objects with lower priority must be at their initial poses.

A coupled node: To compute paths for the objects on a strongly connected component, the method first sets as the target pose of all objects not belonging to the component their current pose (line 15). For all the objects in the component, the final pose remains the same (line 16). For the objects inside the coupled node, new minimum constraint removal paths are generated (line 17). These paths are less constrained than those computed by the original fmRS (where paths are constrained both by the initial and target object poses) because the algorithm considers as obstacles only the current positions of those objects not in the component.

In either case, if a path is found for the current node (line 18), the path is appended to the current solution (line 19) and the objects are transferred to their targets (line 20). Otherwise, the objects corresponding to w remain at their current pose. If there is at least one object moved to its target, then the method returns the corresponding partial solution.

IV. Properties

Consider a path π that is an arbitrary solution to the problem and will bring all objects $\mathcal{O}$ from an initial arrangement α_I to a final arrangement α_F.

Lemma 1: The solution path π can be decomposed into a sequence of segments $\{\pi_1, ..., \pi_k\}$, where each π_i corresponds to a transit motion to grasp an object and a transfer motion to move the object to an intermediate position. Denote the object moved during path segment π_i as o_i.

The path π is a solution to the problem. It is easy to define all the corresponding intermediate arrangements after the application of each π_i, denoted as α_i. All of these intermediate arrangements can be eventually sampled from the high-level task planner. Assume that α_i has already been reached from the initial arrangement α_I and is selected for propagation. Then eventually the arrangement α_{i+1} will be sampled and it will be generated using any of the considered methods as extension primitives (mRS, fmRS, their partial variants or even an individual pick&place action). The new arrangement α_{i+1} can be connected to the initial arrangement α_i, because all of these primitives can successfully connect two arrangements that differ by only one object pose where the transfer is feasible. If it wasn't feasible, the path π_{i+1} would not exist.

Lemma 2: A new arrangement α_i can be acquired from a previous arrangement α_{i-1} along the solution path π.

Theorem 1: The high-level task planner $\mathtt{Bi-RRT}$, using any of the considered primitives (p&p, mRS or fmRS), will find a solution to the rearrangement problem, if a solution exists, with probability reaching 1 as the number of samples increases.

Since all arrangements α_i along a solution path π can eventually be generated and they can be pair-wise connected with any of the primitives, the methods will eventually generate the solution path π in every case.

V. Experimental Evaluation

Experimental Setup: Using a software package for robot planning [12], [13], the methods have been tested in 3 setups: "grid@tabletop" (Fig. 6 (top)), "grid@shelf" (Fig. 6 (bottom)) and "Beer challenge" (Fig. 1). A model of a 7-DOF Yaskawa Motoman arm is used for manipulation.

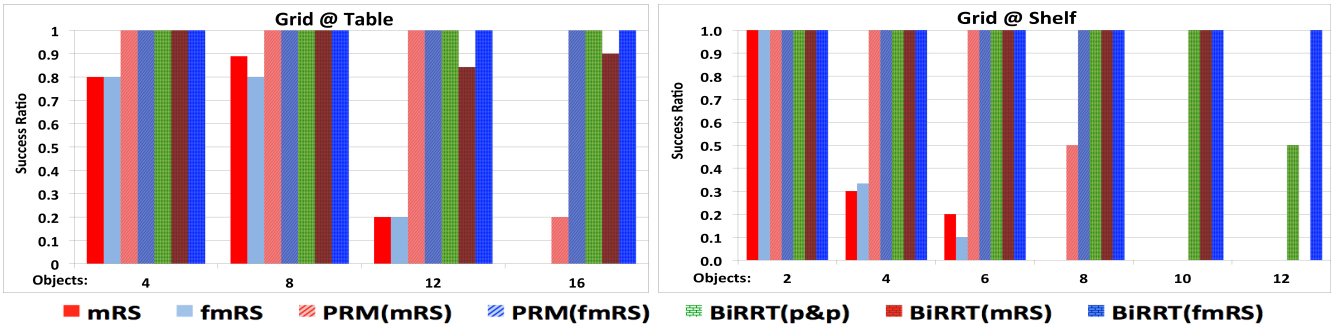

Fig. 5. Left: Success ratios for different methods in the "grid@tabletop" benchmark. Both approaches using fmRS managed to solve all the problems. Right: Success ratios for different methods in the "grid@shelf" benchmark. Bi − RRT with fmRS managed to solve all the problems.

The "grid@tabletop" benchmark places 2 to 16 objects on a tabletop, while the "grid@shelf" benchmarks has 2 to 12 cylinders placed in a shelf that does not allow overhand grips. In both of these scenarios, the objective is to rearrange the objects from a random to a grid arrangement. The "Beer challenge" involves 6 objects in a shelf with multiple bins. Initially the objects and the beer can are randomly placed inside the bins and the objective is to bring the "beer" (a can in the visualization) on the table next to the shelf, while all the other objects keep their initial pose.

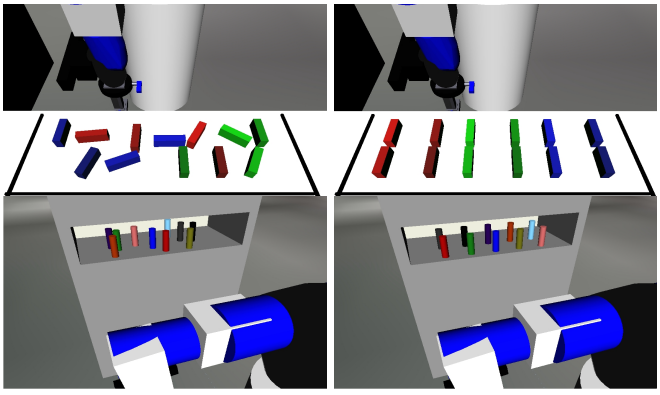

Fig. 6. The "grid@tabletop" (top) and "grid@shelf" (bottom) problems.

Seven different methods are tested: a) the original mRS [1], b) the proposed faster but incomplete fmRS, c) a high-level PRM task planner with mRS as the local connection primitive [5], d) the PRM with fmRS as the local primitive, e) the Bi − RRT high-level task planner with Pick&Place (p&p) as the expansion primitive, f) the Bi − RRT with mRS as the expansion primitive and g) the Bi − RRT with fmRS as the expansion primitive. 15 experiments were performed for each combination of method and environment. A time limit of 30 minutes was provided to the methods.

Tabletop Results: Fig. 5 (left) provides success ratios of the methods for "grid@tabletop", which drop quickly for the monotone primitives as the number of objects increases. Both high-level task planners using mRS also suffer from lower success ratios as the number of objects increases. The rest of the approaches were able to provide solutions for all instances. Fig. 7 (left) provides the running times for each algorithm for different numbers of objects for the same benchmark. The fast monotone solver runs faster than the original monotone one, especially in larger-scale examples.

Overall, the task planners using fmRS, as well as Bi − RRT with p&p, manage to solve all the problems. The Bi − RRT using fmRS manages to solve all the hard problems, with 16 objects on the table, in less than 2.7 seconds on average and is the best alternative.

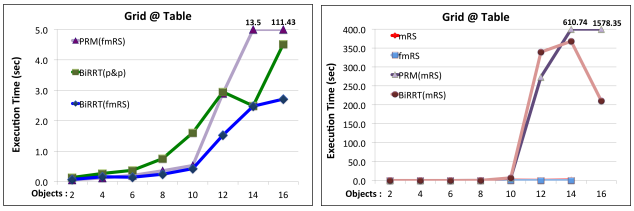

Fig. 7. Average execution time for 15 runs for 2 to 16 objects on a tabletop. Average times include examples that failed to find a solution, either due to the time limit or because the problem is not monotone. Left: The best 3 performing algorithms with 100% success. Right: The algorithms with failures and slow running times.

Shelf Results: Fig. 5 (right) provides the results for "grid@shelf ". Here the arm has reduced reachability and is not able to use overhand grasps. As the number of objects increases the challenges are no longer monotone and the monotone solvers' success ratios quickly decrease. All the high-level task planners manage to solve problems up to 6 objects. When the problems become harder, the connectivity between arrangements is challenging and both PRM approaches failed to find solutions when the shelf was cluttered. These scenarios show the benefits of using an expansion step instead of connection primitive. The Bi − RRT method using mRS has good success ratio for easy problems but its success ratio decreases as the problems become harder. The Bi − RRT with pick&place managed to find solutions for problems with 10 objects but not beyond. The Bi − RRT approach using fmRS could explore the space more efficiently, and as a result managed to solve all the problems in all cases.

Fig. 8 (top) shows the average time each algorithm needs to solve the "grid@shelf" challenge. The averages include the examples where the algorithm could not find a solution either because of the time limit or in the case of the monotone algorithms, incapability of the method to solve the problem. Fig. 8 (bottom) depicts the number of object moves the manipulator will perform to solve the problem. The Bi − RRT method using fmRS outperforms all the algorithms in both running time and number of objects that have to be moved. The differences in the shelf environment, which is more challenging, are more pronounced relative to the tabletop.

In terms of running time, there is the expected improvement from the monotone to the fast-monotone solver. The Bi − RRT method with mRS requires more time to expand between two nodes, especially in the examples with the higher number of objects. On the other hand, the Bi − RRT with fmRS requires less time to connect two nodes. For that reason, the Bi − RRT with fmRS manages to expand more arrangements and explore the space faster and more efficiently. The Bi − RRT method using pick&place can also search the space fast, due to the fact that the primitive can quickly try to connect two arrangements. Nevertheless, this method needs more time to search the arrangement space, because it can change only one object at a time. As result, it needs to expand more arbitrary arrangements before it finds a solution to a problem.

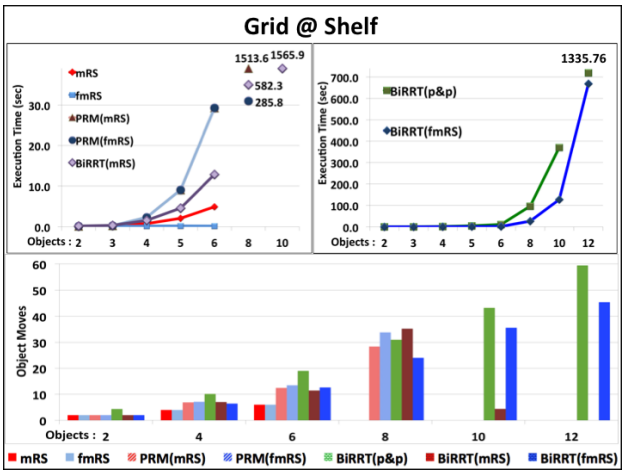

Fig. 8. (top) Average execution time for 15 runs for 2 to 12 objects inside the shelf. The averages include examples where the algorithm failed to find a solution. (bottom) The number of objects the manipulator moves on average for each problem.

Average path quality of solutions returned by the methods for the "grid@shelf" are shown in Figure 8 (bottom). The quality of the path is measured by the number of objects the manipulator moves in order to find a solution. When all the methods manage to solve the problems, the path quality is equivalent. The worst quality comes from the method using the simple pick&place as a low level primitive. For larger-scale problems, the Bi − RRT approach using fmRS provides solution to all the problems while maintaining the best path quality out of all the methods. In the case of 10 objects the approach using mRS appears to have better path quality, but the success ratio of this algorithm is just 20%. This means that the method solved only the simpler problems that required the minimum number of object moves.

"Beer Can Retrieval" Results: For the last experiment the algorithms had to find a solution where the manipulator will remove a beer can from the shelf and place it on the table. Meanwhile, the rest of the objects had to maintain their initial positions. Fig. 9 (left) shows the average running time of 50 runs. About half of the examples could be solved by the monotone primitives. Due to randomness, the beer can is sometimes in a position where the manipulator can reach and move it without moving any other object. For all the other cases, the Bi − RRT using fmRS or the mRS outperforms all the other methods, with fmRS slightly faster than the latter.

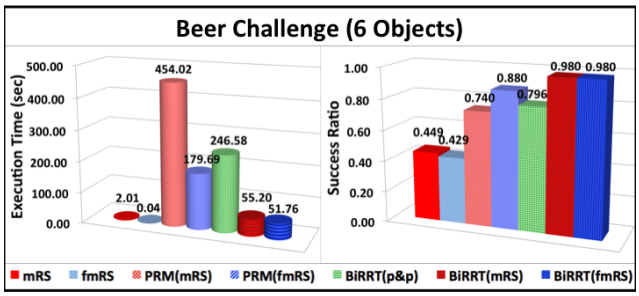

Fig. 9. (left) Average execution time and (right) success ratio for 50 runs for all tested algorithms for the Beer Challenge shown in Fig.1.

VI. Related Work

Manipulation planning among multiple movable obstacles has been considered for "monotone" problems where each obstacle can be moved at most once [1]. It can be seen as an extension of navigation among movable obstacles (NAMO) [14], [15], which is already hard [16], so completeness can be achieved only for certain problem subclasses [17], [18], [19]. These challenges relate to the minimum constraint removal problem, which seeks to minimize the number of constraints that must be displaced to find a path [20], [3].

Non-monotone instances are recognized as hard rearrangement challenges. There are recent approaches for rearrangement that can deal with non-monotone instances under certain conditions [21], [11], [10], [22], [23]. One method simplifies the problem by requiring that all objects are unlabeled, i.e., interchangeable and can occupy any pose in a final arrangement [22]. An alternative method imposes a grid for placing the objects and then uses techniques, such as answer-set programming [21]. Other approaches view the rearrangement problem as an instance of general manipulation task planning [11] and evaluate good heuristics for integrated planning [10] or discover a symbolic language for manipulation on the fly [24]. Many approaches employ a high-level symbolic planner [25], [26]. For instance, geometric constraints can be incorporated into the high-level language [27], or it is possible to plan in the cross product of the high-level symbolic reasoning and the low-level configurations [28]. Manipulation planning has been successfully approached in the literature with tree sampling-based planners [29], [30].

This paper deals with the type of problems where the start and the goal poses for the objects are fully specified. Most of the related work though [17], [18], [11], [10], [31] deal with a more abstract problem, where the goal pose is specified only for the target object and the rest of the objects can be placed anywhere. The proposed methods could also solve these kind of problems, however, this may not be the fastest way and the other methods that have been designed to solve these problems are more amenable to under-specified problems.

VII. Discussion

This work proposes a faster approach for computing solutions for monotone rearrangement relative to an existing influential algorithm [1]. The proposed solution trades completeness for efficiency by avoiding exhaustive search. Instead, it commits to using only "minimum constraint removal" paths for the objects, based on which it is possible to quickly figure out whether a monotone solution exists. This trade-off proves to be beneficial when using monotone rearrangement as a subroutine in a `Bi-RRT` task planner in the space of general object arrangements. The integration of the fast primitive with this task planner results in a method that a) is probabilistically complete, b) exhibits higher success ratios and computational performance relative to the alternatives, as well as c) better solution paths. While not demonstrated here, objects with different geometries and in general poses can be easily considered by the methods, as they are agnostic to these parameters.

It is interesting to consider whether more powerful rearrangement primitives can be considered as subroutines for manipulation task planning, especially methods that already address non-monotone challenges. Other interesting directions include: a) multiple arms collaborating to rearrange objects [32], b) non-prehensile actions [33], [34], c) mobile manipulation [10] and d) dealing with symbolic goal conditions, where conditions should be satisfied before the robot is able to find a solution. This can involve problems where the robot has to place objects on top of each other. Future efforts should also aim towards robust rearrangement trajectories under the presence of uncertainty, which can arise from pose estimation processes [35]. It is interesting to evaluate how different object placements may result in different probability of success during real-world execution. From a system's point of view it is interesting to investigate how to quickly compute such rearrangement paths through the use of cloud-based computation [36].

References

[1] M. Stilman, J. Schamburek, J. J. Kuffner, and T. Asfour, "Manipulation Planning Among Movable Obstacles," in *ICRA*, 2007.

[2] J. van den Berg, J. Snoeyink, M. Lin, and D. Manocha, "Centralized Path Planning for Multiple Robots: Optimal Decoupling into Sequential Plans," in *Robotics: Science and Systems (RSS)*, 2009.

[3] K. Hauser, "Minimum Constraint Displacement Motion Planning," in *Robotics: Science and Systems (RSS)*, 2013.

[4] A. Krontiris and K. E. Bekris, "Computational tradeoffs of search methods for minimum constraint removal paths," in *Symposium on Combinatorial Search (SoCS)*, Dead Sea, Israel, 06/2015 2015.

[5] ——, "Dealing with difficult instances of object rearrangement," in *Robotics: Science and Systems (RSS)*, Rome, Italy, July 2015.

[6] S. Karaman and E. Frazzoli, "Sampling-based Algorithms for Optimal Motion Planning," *IJRR*, vol. 30, no. 7, pp. 846–894, June 2011.

[7] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces," *IEEE TRA*, vol. 12, no. 4, pp. 566–580, 1996.

[8] S. M. LaValle and J. J. Kuffner, "Rapidly-exploring random trees: Progress and prospects," in *WAFR*, 2000.

[9] T. Siméon, J.-P. Laumond, J. Cortés, and A. Sahbani, "Manipulation Planning with Probabilistic Roadmaps," *IJRR*, no. 23, 2004.

[10] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling, "Ffrob: An efficient heuristic for task and motion planning," in *International Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2014.

[11] 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 *ICRA*, 2014.

[12] Z. Littlefield, A. Krontiris, A. Kimmel, A. Dobson, R. Shome, and K. E. Bekris, "An Extensible Software Architecture For Composing Motion And Task Planners," in *Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*, Bergamo, Italy, 2014.

[13] A. Kimmel, A. Dobson, Z. Littlefield, A. Krontiris, J. Marble, and K. E. Bekris, "Pracsys: An Extensible Architecture For Composing Motion Controllers And Planners," in *Simulation, Modeling and Programming for Autonomous Robots (SIMPAR)*, Tsukuba, Japan, 2012.

[14] D. Nieuwenhuisen, A. Frank van der Stappen, and M. H. Overmars, "An Effective Framework for Path Planning amidst Movable Obstacles," in *WAFR*, 2006.

[15] M. Levinh, J. Scholz, and M. Stilman, "Hierarchical Decision Theoretic Planning for Navigation Among Movable Obstacles," in *WAFR*, 2012.

[16] G. Wilfong, "Motion Planning in the Presence of Movable Obstacles," in *Proc. of the 4^{th} Annyal Symp. of Computational Geometry*. New York City, NY, USA: ACM, 1988, pp. 279–288.

[17] M. Stilman and J. Kuffner, "Navigation among Movable Obstacles: Realtime Reasoning in Complex Environments," in *Journal of Humanoid Robotics*, 2004, pp. 322–341.

[18] M. Stilman and J. J. Kuffner, "Planning Among Movable Obstacles with Artificial Constraints," in *WAFR*, 2006.

[19] J. van den Berg, M. Stilman, J. J. Kuffner, M. Lin, and D. Manocha, "Path Planning Among Movable Obstacles: A Probabilistically Complete Approach," in *WAFR*, 2008.

[20] K. Hauser, "The Minimum Constraint Removal Problem with Robotics Applications," in *WAFR*, 2012.

[21] G. Havir, G. Ozbilgin, E. Erdem, and V. Patoglu, "Geometric Rearrangement of Multiple Moveable Objects on Cluttered Surfaces: A Hybrid Reasoning Approach," in *ICRA*, 2014.

[22] A. Krontiris, R. Shome, A. Dobson, A. Kimmel, and K. E. Bekris, "Rearranging similar objects with a manipulator using pebble graphs," in *IEEE Humanoids*, Madrid, Spain, 2014.

[23] J. Ota, "Rearrangement Planning of Multiple Movable Objects," in *ICRA*, 2004.

[24] G. Konidaris, L. Kaelbling, and T. Lozano-Pérez, "Constructing Symbolic Representations for High-Level Planning," in *AAAI*, 2014.

[25] L. P. Kaelbling and T. Lozano-Pérez, "Hierarchical Task and Motion Planning in the Now," in *ICRA*, 2011.

[26] E. Erdem, K. Haspalamutgil, C. Palaz, V. Patoglu, and T. Uras, "Combining High-Level Causal Reasoning with Low-Level Geometric Reasoning and Motion Planning for Robotic Manipulation," in *ICRA*, 2011, pp. 4575–4581.

[27] C. Dornhege, M. Gissler, M. Teschner, and B. Nebel, "Integrating Symbolic and Geometric Planning for Mobile Manipulation," in *SSRR*, 2009.

[28] S. Cambon, R. Alami, and F. Gravot, "A Hybrid Approach to Intricate Motion, Manipulation, and Task Planning," *IJRR*, no. 28, 2009.

[29] D. Berenson, S. S. Srinivasa, D. Ferguson, and J. J. Kuffner, "Manipulation Planning on Constraint Manifolds," in *Proc. of the IEEE Intern. Conf. on Robotics and Automation (ICRA)*, 2009.

[30] D. Berenson, S. S. Srinivasa, and J. J. Kuffner, "Task Space Regions: A Framework for Pose-Constrained Manipulation Planning," *IJRR*, vol. 30, no. 12, pp. 1435–1460, 2012.

[31] J. Barry, L. Kaelbling, and T. Lozano-Pérez, "A hierarchical approach to manipulation with diverse actions," in *ICRA*, 2013.

[32] A. Dobson and K. E. Bekris, "Planning Representations And Algorithms For Prehensile Multi-Arm Manipulation," in *IROS*, 2015.

[33] A. Cosgun, T. Hermans, V. Emeli, and M. Stilman, "Push Planning for Object Placement on Cluttered Table Surfaces," in *IROS*, 2011.

[34] M. R. Dogar and S. S. Srinivasa, "A Framework for Push-Grasping in Clutter," in *Robotics: Science and Systems (RSS)*, 2011.

[35] C. Rennie, R. Shome, K. E. Bekris, and A. F. De Souza, "A Dataset For Improved Rgbd-Based Object Detection And Pose Estimation For Warehouse Pick-And-Place," *IEEE RA-L - also at ICRA*, 2016.

[36] K. E. Bekris, R. Shome, A. Krontiris, and A. Dobson, "Cloud Automation: Precomputing Roadmaps For Flexible Manipulation," in *IEEE Robotics and Automation Magazine*, vol. 22, no. 2, 2015.