Rearranging Similar Objects with a Manipulator using Pebble Graphs

Athanasios Krontiris, Rahul Shome, Andrew Dobson, Andrew Kimmel and Kostas Bekris

***Abstract*— This work proposes a method for efficiently computing manipulation paths to rearrange similar objects in a cluttered space. Rearrangement is a challenging problem as it involves combinatorially large, continuous configuration spaces due to the presence of multiple bodies and kinematically complex manipulators. This work leverages ideas from multi-robot motion planning and manipulation planning to propose appropriate graphical representations for this challenge. These representations allow to quickly reason whether manipulation paths allow the transition between entire sets of object arrangements without having to explicitly store these arrangements. The proposed method also takes advantage of precomputation given a manipulation roadmap for transferring a single object in the space. The approach is evaluated in simulation for a realistic model of a `Baxter` robot and executed on the real system, showing that the method solves complex instances and is promising in terms of scalability and success ratio.**

I. Introduction

Humanoid manipulators can benefit from the ability to rearrange objects in constrained, cluttered human environments. Such a skill can be useful in manufacturing, where multiple products need to be arranged in an orderly manner, or when a robotic assistant needs to rearrange items in a fridge to retrieve an object. This paper proposes a methodology for solving such tasks in geometrically complex and constrained scenes using a humanoid arm. The focus is on the case the target objects are geometrically similar and interchangeable. A key challenge in developing practical algorithms for such problems is the size of the search space. A complete method must operate in the Cartesian product of the configuration spaces of all the objects and the robot. Problems also become hard when the objects are placed in tight spaces, coupled with limited manipulator maneuverability. This paper deals primarily with these combinatorial and geometric aspects and proposes motion planning methods that return collision-free paths for manipulating rigid bodies.

The approach reduces the continuous, high-dimensional rearrangement problem into several, discrete rearrangement challenges on "manipulation pebble graphs" (`MPGs`). The inspiration comes from work in algorithmic theory on "pebble graphs" [2] and related recent contributions in multi-robot motion planning [22]. The transfer of the idea of pebble graphs to the problem of rearrangement is not trivial. The presence of a manipulator in rearrangement planning induces additional constraints relative to the previous work on multi-robot motion planning [22], which required the development of different solutions for the connection of `MPGs`. The current paper builds on top of work in manipulation planning [21] so as to allow the efficient use of "pebble graphs" in the context of a manipulator rearranging similar objects.

Work by the authors has been supported by NSF CCF:13307893. 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 at Rutgers University, 110 Frelinghuysen Rd., Piscataway, NJ, 08854 `kostas.bekris@cs.rutgers.edu`.

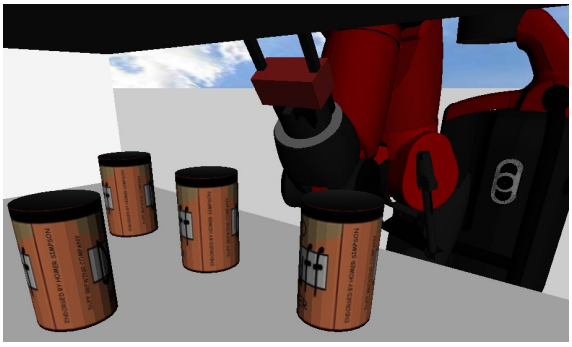

Fig. 1. In order for `Baxter` to grasp the object at the back of the shelf, the other objects need to be rearranged.

The nodes of an `MPG` correspond to potential stable poses for the objects. Edges indicate that there is a path for the manipulator to transfer an object between the poses, even if all other nodes are occupied. For most interesting queries, it is difficult to find a single pebble graph that contains both the start and the goal arrangement and solves the problem. It is helpful to generate multiple such graphs and identify transitions between them to find a solution. `MPGs` have the benefit that they can implicitly represent an entire set of object arrangements over the set of stable poses that they correspond to. Only once a sequence of pebble graph transitions that solves the problem has been found, does the algorithm need to extract the sequence of object placements and the corresponding arm paths. The encapsulation of multiple object arrangements within an individual graph helps with the combinatorial aspects of rearrangement. Furthermore, the method can effectively utilize precomputation. For a known workspace and geometry for the objects, it is possible to precompute a manipulation graph and perform many expensive collision checking operations offline.

The approach can efficiently address many rearrangement challenges, albeit with certain concessions. For instance, it must be possible to retract the arm to a safe configuration from every stable grasped pose in a solution sequence. Furthermore, this work deals only with similar geometry, interchangeable objects. Nevertheless, given the motivating work in multi-robot motion planning [22], it is possible to extend the proposed framework to dissimilar objects.

Related Work: Rearrangement [3], [20] relates to many different lines of work in robotics.

Planning among Movable Obstacles: Navigation among movable obstacles (NAMO) is NP-hard [27] and most efforts

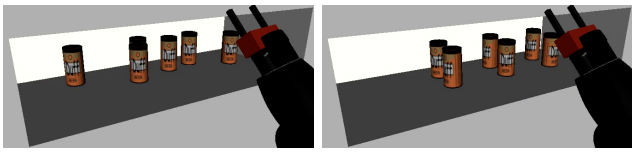

Fig. 2. Initial and final positions for the randomized grid problem with 6 objects. The manipulator is in its "safe configuration".

have dealt with efficiency [19] and provide completeness results only for problem subclasses [23]. A probabilistically complete solution was proposed for simple robots (2-3 DOFs) [5]. A decision-theoretic NAMO framework was presented to deal with uncertainty [17]. The related "minimum constraint removal" problem minimizes the number of constraints that must be displaced to find a path [12], but does not capture negative interactions between obstacles.

Manipulation: Manipulation problems can be approached with a multi-modal "manipulation graph" that contains "transit" and "transfer" paths, which can be built with sampling-based planners [1], [21]. This paper uses this approach as well as asymptotically optimal planners, i.e., PRM* [15], to construct the manipulation graph. Tree sampling-based planners have been used successfully for manipulation planning [4], as well as a variety of other approaches, such as heuristic search [7] or optimization [29]. The benefit of the proposed approach is that it naturally reduces the original problem to discrete rearrangement challenges.

Manipulation among movable objects has been considered for "monotone" problems where each obstacle can be moved at most once [24]. The current work can resolve more complex challenges. Assembly planning similarly solves multi-body problems but focuses on separating a collection of parts and the robot path is ignored [11], [25]. Another paradigm involves non-prehensile manipulation, such as pushing [8], [9], which is not the focus of this work.

Task and Motion Planning: Rearrangement planning can be seen as an instance of integrated task and motion planning [13], [14]. Manipulation problems have been approached via multi-modal roadmaps to deal both with discrete and continuous parameters [6], [13]. These methods emphasize the key insight that methods should reason over the properties of states without enumerating them.

Multi-robot Motion Planning: This work is motivated by progress in discrete solvers for multi-robot planning [16], [18], [26], [28], which is a hard problem. Related work deals with "pebble motion on graphs" problems, where pebbles need to move from initial to goal vertices [2], [10]. Determining the feasibility of such problem instances can be done in linear time [2], [10], inspiring a recent method for continuous multi-robot motion planning [22]. The method employs sampling-based planners and reduces multi-robot planning into a sequence of discrete pebble graphs, so that pebble movements quickly translate to robot motion. This paper is motivated by this approach and defines "manipulation pebble graphs" (MPGs) for manipulation challenges.

Summary of Contribution: This paper details the extension of the pebble graph framework to rearrangement planning with a high-DOF humanoid arm, and evaluates its performance in experiments. [1] In addition, the framework is able to deal with non-monotone problems, where multiple grasps of an object are needed in order to find a solution to the problem. Such problems are challenging for alternative methodologies.

II. Problem Setup and Notation

Consider a **workspace** $\mathcal{W} \subset \mathbb{R}^3$ with (a) static obstacles, (b) a set of k movable, identical geometry, unlabeled rigid body objects and (c) a manipulator $\mathcal{M}$. Each **object** $o_i \in \mathcal{O}$ can acquire a *collision-free* **pose** $p_i \in \mathcal{P}$, where $\mathcal{P} \subseteq SE(3)$.

An **unlabeled arrangement** $\alpha \in \mathcal{A}$ specifies a k-combination of poses $\{p_1, \ldots, p_k\}$, where $p_i \in \mathcal{P}$ and results in a *collision-free* placement of the objects in $\mathcal{W}$. Permutations of α are indistinguishable, since the objects are unlabeled. The **configurations** $q^{\mathcal{M}} \in \mathcal{Q}^{\mathcal{M}}$ place the **manipulator** $\mathcal{M}$ in the collision-free workspace.

The **configuration space** $\mathcal{Q} = \mathcal{Q}^{\mathcal{M}} \times \mathcal{A}$ corresponds to the Cartesian product of the manipulator's C-space and the space of unlabeled object arrangements. $\mathcal{Q}$ does not allow collisions between the manipulator or the objects with obstacles, as well as between objects, or between the manipulator and objects. The manipulation problem then has two types of valid configurations: (a) **Stable configurations:** Collision-free configurations $\mathcal{Q}^s$ where the objects are in stable poses where they rest without any forces from the arm; (b) **Grasping configurations:** These correspond to configurations $\mathcal{Q}^g$ where the manipulator is grasping an object. This paper's simulations use the set of stable poses $\mathcal{P}^s \subset \mathcal{P}$ corresponding to the placement of the objects on a horizontal surface, i.e., $\mathcal{P}^s \equiv SE(2)$.

Define a **type** $t \in \mathbb{T}$ for such manipulation problems (also called a "mode"), which has two values. In the *transit* type, the manipulator is not carrying an object and $q \in \mathcal{Q}^s$. In the *transfer* type, the manipulator carries an object and $q \in \mathcal{Q}^g$. The only way that the type can change is if the problem is in a *transition* configuration $q \in \mathcal{Q}^t = \mathcal{Q}^s \cap \mathcal{Q}^g$.

The problem's **state space** $\mathbb{X} : \mathcal{Q} \times \mathbb{T}$ arises from the combination of the configuration space $\mathcal{Q}$ and the type space $\mathbb{T}$. It is then easy to define the sets $\mathbb{X}^f$ (free), $\mathbb{X}^s$ (stable), $\mathbb{X}^g$ (grasping) and $\mathbb{X}^t$ (transition), e.g., ($\mathbb{X}^t = \mathcal{Q}^t \times \mathbb{T}$).

The Unlabeled Rearrangement Problem: Given an initial state $x_0 = ((q_0, \alpha_0), t_0) \in \mathbb{X}^s$ (Fig.2 left) and a final state $x_1 = ((q_1, \alpha_1), t_1) \in \mathbb{X}^s$ (Fig.2 right), compute a continuous *path* $\pi \in \Pi : [0, 1] \rightarrow \{\mathbb{X}^s \cup \mathbb{X}^g\}$ such that $\pi(0) = x_0$, $\pi(1) = x_1$. The path is an alternating sequence of transit and transfer states. The types of states along path π change only when $\pi(s) \in \mathbb{X}^t$.

III. Rearrangement Planning

A naive approach to solving unlabeled rearrangements would be to build a manipulation graph [1] in the entire $\mathbb{X}$. This is intractable, however, as the dimensionality increases with the number of objects. The idea here is to abstract out the motion of the manipulator and then reason directly

[1] Videos can be found: http://www.cs.rutgers.edu/~kb572/humanoids_14/rpg/

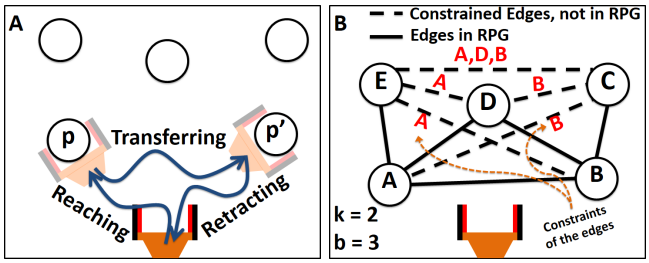

Fig. 3. Each edge on an MPG is the combination of a reaching, transferring and retracting path. Edges can be constrained by other nodes of the MPG.

about the movement of objects between different stable poses in $\mathcal{P}^s$. Reasoning about the movement of multiple objects can take place over discrete graphical representations so as to take advantage of linear-time path planning tools for rearranging unlabeled "pebbles" on a graph from an initial to a target arrangement [2].

A sampling approach can be used to define graphs where nodes correspond to stable poses in $\mathcal{P}^s$ and edges correspond to collision-free motions of the arm that transfer an object between stable poses. If such a graph is connected and contains all the poses from the initial and target arrangements, then a discrete solver can be used to define a solution in the continuous space as long as placing objects in different poses does not cause collisions [2].

It may be difficult or even impossible, however, to construct a single such graph that directly solves the problem. For example, the poses in the initial and target arrangements could be already overlapping, or it may not be possible to ensure connectivity with collision-free motions of the arm. Motivated by work in the multi-robot motion planning literature [22], the current paper considers multiple such graphs, referred to as "manipulation pebble graphs" (MPGs), as shown in Fig.3B.

Manipulation Pebble Graphs: MPGs are built so that the objects are placed in collision-free, stable poses. They have at least k poses and b additional poses as nodes (called "blanks"). The extra nodes allow the rearrangement of k objects on the MPGs. The poses used in an MPG are defined as a pumped arrangement.

Definition: A pumped arrangement $\alpha^{\mathcal{P}}$ is a set of $n = k + b$ poses where: (a) These poses are collision-free, stable poses (i.e., subset of $\mathcal{P}^s$); (b) No two objects will collide if placed on any two poses of the pumped arrangement.

Definition: An MPG is a graph $\mathcal{G}_P(\alpha^{\mathcal{P}}, \mathcal{E}^P)$, where the set of nodes is a pumped arrangement $\alpha^{\mathcal{P}}$. The set of edges $\mathcal{E}^P$ corresponds to pairs $(p, p') \in \alpha^{\mathcal{P}}$, for which the manipulator can transfer objects between poses p, p' without collisions, given that potentially every other pose of $\alpha^{\mathcal{P}}$ is occupied.

Algorithm 1 describes the construction of an MPG. The algorithm receives as input a "safe" configuration of the manipulator $q_s^{\mathcal{M}}$, which is one that does not interfere with the objects placed on any stable poses, and typically is a retracted arm configuration. The algorithm also needs the size of the MPG, i.e., the parameters k and b. The algorithm will return an MPG and a set of additional constrained edges E_c together with the poses that block them. The constrained edges are edges that are not added in the MPG.

The algorithm starts by selecting a non-intersecting set of $k + b$ poses that create a pumped arrangement (Line 1). For each pair of poses p and p', a path has to be computed that allows the manipulator to move an object from p to p'. This path consists of three segments as shown in Fig. 3:

i) A path for the arm from its safe configuration $q_s^{\mathcal{M}}$ to a transition state $x_p^t \in \mathbb{X}^t$ for pose p.

ii) A path that transfers the object from $x_p^t \in \mathbb{X}^t$ to a state $x_{p'}^t \in \mathbb{X}^t$ for p'.

iii) A retraction path from $x_{p'}^t$ back to $q_s^{\mathcal{M}}$.

Algorithm 1: CREATE_MPG$(q_s^{\mathcal{M}}, k, b)$

```
1  V_P ← Sample_Valid_Pumped_Arrangement(k + b);
2  E_P ← ∅, E_c ← ∅;
3  for p, p' ∈ V_P and p ≠ p' do
4      π ← Compute_Min_Conflict_Path(q_s^M, p, p');
5      if is_valid(π) then
6          E_P ← E_P ∪ ((p, p'), π);
7      else
8          c ← Compute_Constraints(π);
9          E_c ← E_c ∪ ((p, p'), π, c);
10 return {G_P(V_P, E_P), E_c}
```

For each edge the algorithm aims to compute a "minimum conflict path" given that objects may be placed in all poses but p'. The "minimum conflict path" is computed heuristically by searching for a path that is within a bound of the shortest path ignoring the other poses and which minimizes conflicts with the other poses. If the computed path is collision-free, then an edge (p,p') is added to the MPG (Line 6), which stores the corresponding path. In case the "minimum conflict path" π collides with objects on other poses of the MPG, the edge (p,p') is not added. The algorithm identifies these potential collisions and stores them in a set of constraints c (Line 8). Such edges with constraints are stored in the data structure E_c (Line 9). The algorithm returns the MPG $\mathcal{G}_P(V_P, E_P)$ and the set E_c of constrained edges. The use of E_c is described later on.

Signatures: Objects within each connected component of an MPG can be safely rearranged, since MPG's edges correspond to valid motions of the manipulator regardless of object placement on the graph's node. A discrete solver [2] can be used to achieve all feasible arrangements, given an MPG's connectivity. There is no need to explicitly store all the possible arrangements over an MPG, since the "signature" of an arrangement α is sufficient to describe all reachable rearrangements.

Definition: A signature, $\sigma_{\mathcal{G}_P}(\alpha)$ is the number of objects contained in each connected component of MPG $\mathcal{G}_P$ according to an arrangement α.

Super-graph: It is also possible to move objects between poses belonging to different MPGs, if the MPGs share at least k poses that can be simultaneously occupied by objects, given the corresponding signatures.

These observations give rise to a super-graph structure, where each super-node corresponds to an MPG and a sig-

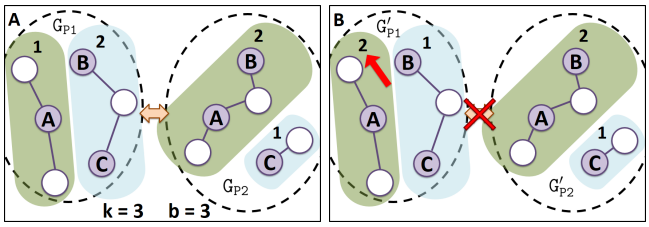

Fig. 4. CONNECT_NODE: both MPGs share the shaded positions (A,B,C). (left) The nodes are connected since both signatures allow the three common positions to contain an object. (right) The nodes cannot be connected, since on $\mathcal{G}'_{P1}$ it is not possible to place objects both on B and C.

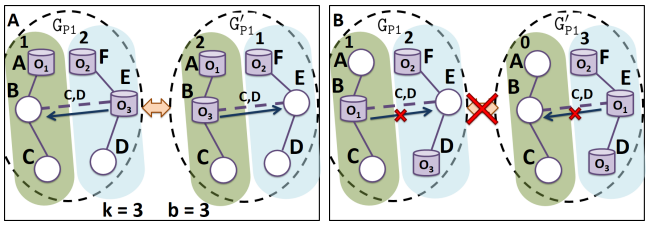

Fig. 5. CONNECT_SIBLINGS: Nodes with same MPG but different signatures. The constrained (dashed) edge is not in the MPG. (left) An edge is added, since an object can be moved along the edge given constraint (C,D). (right) The edge is not added since pose D cannot be emptied.

nature. Super-edges correspond to transitions between such super-nodes. The initial and target arrangements define two such super-nodes. Then the approach generates and connects super-nodes (i.e., MPGs and signatures) until the initial and target arrangement are connected on the super-graph. At that point, the rearrangement problem is solved and the necessary motions of the manipulator can be extracted along the path connecting the initial and target nodes on the super-graph.

Constructing super-nodes: All arrangements in an MPG with the same signature are reachable from each other, i.e., if $\sigma_{\mathcal{G}_P}(\alpha) = \sigma_{\mathcal{G}_P}(\alpha')$, then there is some sequence of transitions and a corresponding path for the manipulator, π, which brings α to α', and vice versa.

Definition: A super-node is an MPG and a signature σ shared by a set of reachable arrangements of objects on the MPG.

Definition: Sibling nodes correspond to super-nodes with the same MPG but different signature.

Algorithm 2: CREATE_SUPERNODES$(H(V_H, E_H), q_s^{\mathcal{M}}, k, b)$

1 $\{\mathcal{G}_P, E_c\} \leftarrow$ CREATE_MPG$(q_s^{\mathcal{M}}, k, b)$;
2 $\Sigma \leftarrow$ Generate_Signatures$(\mathcal{G}_P)$;
3 $V_n \leftarrow \emptyset$;
4 **for** $\sigma \in \Sigma$ **do**
5 $\quad v_H \leftarrow (\mathcal{G}_P, \sigma)$, $V_n \leftarrow V_n \cup v_H$;
6 $\quad V_H \leftarrow V_H \cup v_H$;
7 $\quad$ CONNECT_NODE(H, v_H);
8 CONNECT_SIBLINGS(H, V_n, E_c);

Algorithm 2 takes as an argument the existing super-graph $H(V_H, E_H)$ and adds new super-nodes and edges. It needs to be aware of the safe configuration $q_s^{\mathcal{M}}$, and the size of the MPGs it will construct: $n = k + b$. When this function completes, the new "sibling" nodes will be in the super-graph and appropriately connected with edges.

The algorithm begins by constructing a random, valid MPG (Line 1). Then, it computes all possible signatures that the generated MPG can attain (Line 2), i.e., for G_{P2} in Fig.4, the possible signatures are {3,0},{2,1},{1,2}. A new super-node v_H is created using the created MPG and one of the signatures (Line 5). When v_H is added in the super-graph H, the functions CONNECT_NODE and CONNECT_SIBLINGS try to connect v_H with other super-nodes (Lines 8, 9).

Connecting super-nodes: The super-nodes in H have two different ways to connect to each other.

(A) CONNECT_NODES connects super-nodes created by a new MPG with super-nodes created using previous, different MPGs. The function first identifies whether super-node v_H shares at least k common poses with an existing super-node. If true, the method checks whether both super-nodes can achieve placing k objects on the k common poses given the super-nodes' signatures (Fig. 4). If they both can, an edge is added between the two super-nodes, which represents a context switch between two arrangement sub-problems. This switch is feasible because only the b unoccupied poses of the first MPG will be replaced with the b unoccupied poses of the other MPG, while the k objects will have to stay on their current positions.

(B) CONNECT_SIBLINGS connects "sibling" super-nodes by using the constraint edges E_c to switch between different signatures on the same MPG. The algorithm connects super-nodes only if there is a motion that allows transition between signatures of the "siblings" (Fig. 5). The arm is able to follow a constrained edge in E_c if the constraining poses can be emptied given the super-node's signature. For all constrained edges, the algorithm finds the connected components of the MPG that this edge is connecting. If the connected components are different and the edge is feasible given the signature, then this edge can be used for potential connections between two "siblings". An edge is feasible, f the poses that constrain the edge and the target pose p' of the constrained edge can be emptied given the signature of v_H. Furthermore, it must be possible to bring an object to the source pose p of the edge. Moving along such edges results in a change of signature relative to that of v_H and the new signature σ_n can be computed. Then an edge is created that connects the two "siblings" in the super-graph.

Incremental Approach: Generating new super-nodes can take place in a fashion similar to a bidirectional sampling-based tree planner. The proposed approach first generates two super-nodes of size k for the initial and target arrangements. The b "blank" poses are unnecessary for these nodes. Then, k poses are selected from an already existing super-node to be used as poses for the new MPG. These poses can be occupied by objects given the signature of the selected super-node. An additional k poses are sampled for the new MPG. Alg.2 will generate all the "siblings" super-node for the new MPG that will be added in the super-graph and attempt connections with existing super-nodes. The process repeats until the initial and target super-nodes are in the same connected component.

Answering Queries: For each super-node, there exists a way to move between all possible arrangements on the

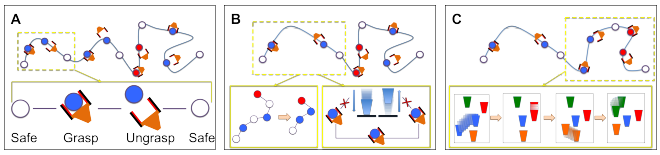

Fig. 6. A: A path decomposed into reaching, transferring and retracting motions. B: If the arm grasps the same object twice, then the intermediate stable state is removed. C: Redundant movements of objects can be removed.

corresponding MPG that have the same signature. The actual paths that accomplish these rearrangements are not explicitly stored. Only during the query phase, when a graph search returns a solution sequence of super-nodes that connects the start and target arrangements, the algorithm computes solutions to the pebble motion problem within each MPG.

The super-graph path is transformed into a manipulation path by solving discrete graph problems [2] on individual MPGs given start and final arrangements on them according to information stored on the hyper-edges. Since the MPG contains edges which have safe motions for the manipulator, any arrangement produced by the approach will be feasible for the robotic arm if it follows the sequence of actions that was used to validate the edge. The start arrangement on the MPG is simply the latest arrangement the objects have reached in the previous MPG. The end arrangement is stored on the edges of the super-graph. For connections between nodes generated from the same MPG, the edges involve a motion and have the constraints associated with this motion encoded. The end configuration is such that those constraints are satisfied.

Smoothing Solutions: Figure 6 shows examples of smoothing the solution path. The colored disks represent occupied poses. The uncolored discs represent the safe configuration $q_s^{\mathcal{M}}$. The edges connecting the nodes represent the trajectory taken by the manipulator to move the objects. A trajectory can be separated into distinct sequences of transit to grasp from $q_s^{\mathcal{M}}$, transfer an object between poses (p,p'), and retract to $q_s^{\mathcal{M}}$, as shown in Figure 6A.

Phase 1 - Consecutive, Identical Grasps: The smoothing process first looks for consecutive grasps of the same object in a solution sequence. If such pairs of grasps exist, the intermediate stable pose is removed, along with any redundant states, as shown in Figure 6B. This has the effect of removing motions, such as raising and lowering the object, which are unnecessary when consecutively grasping the same object.

Phase 2 - Standard Trajectory Smoothing: Each reaching, transferring and retracting sequence is smoothed by checking for shortcuts between pairs of states. If these shortcuts exist, they replace their corresponding intermediate trajectory.

Phase 3 - Maximizing Consecutive Grasps: The solution returned by the algorithm is conservative. This can potentially lead to redundant movements of objects. The trajectory that moves an object o_x from position p^i to p^{i+1} can be represented as $p_x^i \rightarrow p_x^{i+1}$. The trajectory sequence:

$$\{p_a^i \rightarrow p_a^{i+1}, \quad p_b^j \rightarrow p_b^{j+1}, \quad p_a^{i+1} \rightarrow p_a^{i+2}\},$$

contains an intermediate placement of o_a at pose p_a^{i+1}, before eventually moving it to p_a^{i+1} (Figure 6C). If, i) o_a can be moved from pose p_a^i to pose p_a^{i+2} with a collision free path, while o_b is at pose j and ii) the trajectory $p_b^j \rightarrow p_b^{j+1}$ is collision free given that o_a is at pose p_a^{i+2}, then the original trajectory can be replaced with $\{p_a^i \rightarrow p_a^{i+2}, \quad p_b^j \rightarrow p_b^{j+1}\}$.

These phases can be repeated, starting from phase 1, to continually improve the path quality. Once the smoothing phases are complete, phase 2 can applied over the entire trajectory, resulting in the final smoothed solution. In practice, a single application of each phase and a final application of phase 2 are sufficient.

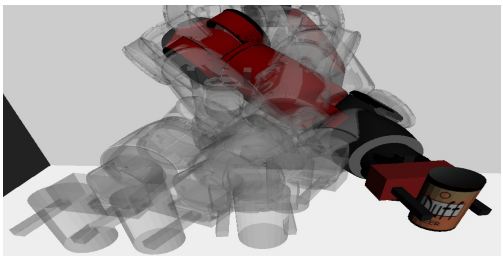

Fig. 7. Transition states $x^t \in X^t$, i.e., stable grasping configurations. An IK solution for the end effector is computed.

IV. Efficient Computation of Manipulation Paths

A manipulation planning primitive is needed during the construction of the MPGs to detect whether an edge on the MPG can be added. For the efficient online computation of this operation, this work generates a manipulation roadmap $\mathcal{R}(\mathcal{V}, \mathcal{E})$ offline, for a single object and the static geometry. In this manner, the roadmap operates over reduced collision-free, single-object states of the form $x = ((q^{\mathcal{M}}, p), t)$.

The first step in the construction of $\mathcal{R}$ is to sample transition states $x^t \in \mathbb{X}^t$, i.e., stable grasping configurations. This can be achieved by first sampling a stable collision-free object pose $p \in \mathcal{P}^s$, and then using inverse kinematics to define the corresponding manipulator's grasping configuration $q^{\mathcal{M}}$ (Fig. 7). If the inverse kinematics function returns a solution and the resulting configuration $(q^{\mathcal{M}}, p)$ is collision-free, then two states are generated with the same configuration: one of the transit type and one of the transfer type. Both of these states are inserted in $\mathcal{R}$ and an edge is added between them.

In the experiments performed for this paper, multiple grasping configurations for an individual object pose are generated. Furthermore, two extra states are inserted in $\mathcal{R}$ during this process: a) a transit state, where the arm's end-effector is slightly retracted from the object, and b) a transfer state, where the arm's end-effector is slightly raised with the object. These additional states help in the connectivity of both the transit and transfer subsets of $\mathcal{R}$.

Given the set of transition states and edges, the method proceeds to separately explore the transit and the transfer subset of the state space in a PRM* fashion. For the transit component, additional grasping configurations for the manipulator are sampled, where the object is no longer required to be in a stable pose. Then neighboring transit states according to a configuration space metric are considered for connection by interpolating between the two configurations. If the interpolated path is collision-free, the edge is added to the roadmap. The same process is repeated for transfer states.

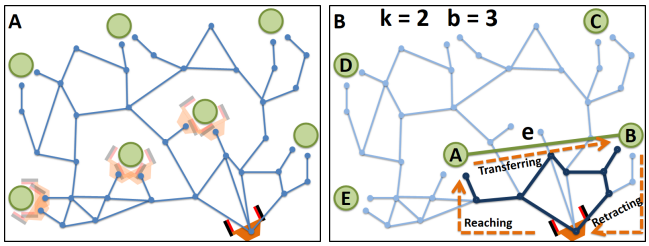

Fig. 8. MPG creation using a manipulation roadmap $\mathcal{R}$: A: The algorithm selects $k + b$ stable poses from $\mathcal{R}$ to generate a pumped arrangement. B: An edge is added if possible given $\mathcal{R}$ to transfer an object between poses A and B without collisions given that all other poses are potentially occupied. The method identifies if the "minimum conflict path" has a constraint.

The objective is to achieve a roadmap that has the minimum number of connected components, so that the manipulator is able to transfer objects among the sampled transition states.

After $\mathcal{R}$ is constructed, it is possible to store helpful information on its edges for the construction of MPGs: for each edge of $\mathcal{R}$, it is possible to compute the set of poses/nodes of $\mathcal{R}$ that would lead into collisions if the arm moved along the edge. In this way, all the necessary collision-checking for the construction of an MPG can take place offline. This information is used during the online process in order to speed up the search for the "minimum conflict path".

Fig. 8 describes the use of the roadmap $\mathcal{R}$ for the online generation of MPGs. First the poses of the MPG are selected from the poses used by $\mathcal{R}$. The roadmap stores multiple configurations $q^{\mathcal{M}} \in \mathcal{Q}^t$, which can grasp an object from a stable pose. The safe configuration $q_s^{\mathcal{M}}$ is also part of $\mathcal{R}$. To add an edge to the MPG between poses p and p', a three segment path of reaching, transferring, and retracting is evaluated on $\mathcal{R}$. A heuristic search approach for "minimum conflict path-planning" on the roadmap is used for this process in order to find the least constrained path, $q_s^{\mathcal{M}} \rightarrow x_p^t \rightarrow x_{p'}^t \rightarrow q_s^{\mathcal{M}}$ given the other poses.

V. Evaluation

The proposed algorithm was evaluated in a simulation environment to determine its scalability and showcase the class of difficult non-monotone problems it can solve. These are challenges, which require an object to be grasped multiple times in order to be solved. Such problems are challenging for state-of-the-art approaches. A 7-DOF arm of a Baxter robot is used in the simulation. The solution paths were executed on a Baxter robot in open loop trials. The identical objects being rearranged are cylinders with height $14cm$ and radius $4cm$. A brute-force approach has not been evaluated in comparison since a search on the combined configuration space of the manipulator and the objects quickly becomes intractable.

Different types of rearrangement problems are considered. These problems include a) environments with random initial configurations to a final grid rearrangement of the objects in a shelf, and b) two non-monotone problems with $k = 3, 4$ (Figure 9). The execution time and the length of the solution trajectory have been measured. These metrics are reported for different values of b, i.e., the number of "blank" poses in the MPGs. An informed pre-computed roadmap $\mathcal{R}$, as described in section IV, has been used in the implementation. A trial is deemed a failure if the method cannot find a solution within the time limit of 600 seconds. The simulations were performed on a machine running on an Intel(R) Xeon(R) CPU E5-4650 0 @ 2.70GHz.

Randomized grid evaluation: The first set of experiments populates a shelf-like environment with $2, 3, 4, 6,$ or 8 objects in mutually exclusive start poses. The goal arrangement is a uniform grid formation on the shelf. The dimensions of the shelf prevent the grasping of the objects from the top. This causes objects to be occluded by other objects in front of them and makes the rearrangement challenge non-trivial. Different values of $b = 1, 2, 3, 4$ are used for these experiments. The manipulation roadmap used for these experiments employed 20 different poses and computed 517 vertices and 6032 edges.

Non-monotone benchmarks: A second set of experiments consists of two non-monotone benchmark problems. For the first problem, 4 objects are placed on a platform which resembles the shelf with the sides removed. The goal arrangement requires the two front objects to be moved at least twice. The trials are performed for different values of $b = 1, 2, 3, 4$. The same roadmap as before is used.

Execution time: Fig. 9(left) shows that the algorithm achieves a high success ratio in the randomized grid setup for $k = 2, 3, 4, 6$. Trials with 8 objects could not finish within the stipulated limit of $600s$. $k = 8$ causes an increase in the problem complexity. The case with 8 objects is constrained by the size of the shelf in terms of the availability of free poses on the shelf and free volume required to move. This affects the motion planning complexity for pose connections. The relatively sparse roadmap used in the experiment helps the running time of the individual motion planning problems, but in a constrained space, the solution to the rearrangement problem is not always achieved within the time limit.

The value of b seems to have a significant role in the execution time for the same k, as illustrated in Fig.9. $b = 1$ seems to deteriorate the algorithm's running time for higher values of k. This is a result of a very slow rate of exploration of the poses available on the shelf for rearrangement, due to only one empty pose available in the MPG. However, the connectivity in an MPG is maximized. High values of b introduce a high combinatorial component of k objects in $k + b$ poses in every MPG and the connectivity of the MPG decreases. For every value of k, the value of b with the best performance in terms of execution time, indicates this trade-off.

Solution quality: The quality of the solution in the randomized grid experiment, is shown in Fig. 9(left). The value of $b = 1$ introduces a low exploration rate, which increases the length of the solution. The conservative solution trajectories are shortened by an average of 48%.

Benchmark evaluation: Fig. 9(right) shows that the objects need to be grasped at least twice in order to achieve the rearrangement. The algorithm is executed 10 times for different values of b. The algorithm achieves a solution in every trial of the benchmark problems.

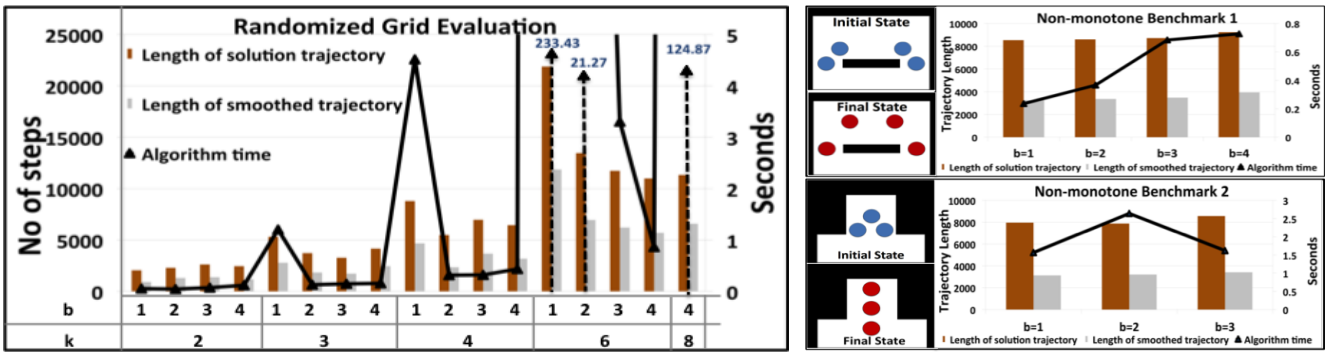

Fig. 9. Left: Average length of the trajectory before and after smoothing (bars) and average solution time (line) for the randomized grid for different values of k and b. Right: Same statistics for the non-monotone benchmarks.

VI. Discussion

The proposed method solves rearrangement problems for similar objects, using a high DOF robot arm, including non-monotone instances that are hard to address with existing methods [24]. Precomputation using a manipulation graph can be appropriately utilized to achieve fast solution times.

The probabilistic completeness properties of the method have not been discussed in this paper. Nevertheless, the authors believe that follow-up work can provide formal arguments in this direction under the following constraint: a problem can be solved with a trajectory where the arm can be retracted to a safe configuration from any grasp. Removing this assumption is an interesting challenge.

The current algorithm does not address objects with different labels or geometries. The motivating work on pebble graphs [22], however, provides a framework to extend the current approach to the case of different objects. Furthermore, the current implementation takes advantage of grasp symmetries about the Z axis for cylindrical objects placed upright. Dealing with general grasps and general resting poses for the objects needs further investigation. There are also interesting variations that can be explored in relation to different ways for connecting super-nodes that may not require the two nodes to share k poses.

The paths computed in simulation have been tested in open-loop (for known initial and target object arrangement) on a real Baxter system arranging cylindrical objects in a shelf. Long paths that involve multiple grasps were unsuccessful as the robot's path deviated due to errors from the computed one. Future efforts will focus on the computation of robust rearrangement trajectories and their robust execution given appropriate sensing input [17].

References

[1] Alami, R., Laumond, J.P., Siméon, T.: Two Manipulation Planning Algorithms. In: WAFR (1996)

[2] Auletta, V., Monti, A., Parente, D., Persiano, G.: A Linear Time Algorithm for the Feasibility of Pebble Motion on Trees. Algorithmica **23**, 223–245 (1999)

[3] Ben-Shahar, O., Rivlin, E.: Practical Pushing Planning for Rearrangement Tasks. IEEE TRA **14**(4) (1998)

[4] Berenson, D., Srinivasa, S.S., Kuffner, J.J.: Task Space Regions: A Framework for Pose-Constrained Manipulation Planning. IJRR **30**(12), 1435–1460 (2012)

[5] van den Berg, J., Stilman, M., Kuffner, J.J., Lin, M., Manocha, D.: Path Planning Among Movable Obstacles: A Prob. Complete Approach. In: WAFR (2008)

[6] Bretl, T., Lall, S., Latombe, J.C., Rock, S.: Multi-step Motion Planning for Free-Climbing Robots. In: WAFR (2004)

[7] Cohen, J.B., Chitta, S., Likhachev, M.: Search-based Planning for Manipulation with Motion Primitives. In: ICRA (2010)

[8] Cosgun, A., Hermans, T., Emeli, V., Stilman, M.: Push Planning for Object Placement on Cluttered Table Surfaces. In: IROS (2011)

[9] Dogar, M., Srinivasa, S.: A Push-Grasping Framework in Clutter. In: RSS (2011)

[10] Goraly, G., Hassin, R.: Multi-Color Pebble Motion on Graphs. Algorithmica **58**(3), 610–636 (2010)

[11] Halperin, D., Latombe, J.C., Wilson, R.H.: A General Framework for Assmbly Planning: the Motion Space Approach. Algorithmica **26**(3-4), 577–601 (2000)

[12] Hauser, K.: Minimum Constraint Displacement Motion Planning. In: RSS (2013)

[13] Hauser, K., Ng-Thow-Hing, V.: Randomized Multi-Modal Motion Planning for a Humanoid Robot Manipulation Task. IJRR (2011)

[14] Kaelbling, L., Lozano-Pérez, T.: Integrated Robot Task and Motion Planning in the Now. CSAIL Technical Report (2012)

[15] Karaman, S., Frazzoli, E.: Sampling-based Algorithms for Optimal Motion Planning. IJRR **30**(7), 846–894 (2011)

[16] Krontiris, A., Luna, R., Bekris, K.E.: From feasibility tests to path planners for multi-agent pathfinding. In: SOCS (2013)

[17] Levinh, M., Scholz, J., Stilman, M.: Hierarchical Decision Theoretic Planning for Navigation Among Movable Obstacles. In: WAFR (2012)

[18] Luna, R., Bekris, K.E.: Efficient and Complete Centralized Multi-Robot Path Planning. In: IROS (2011)

[19] Nieuwenhuisen, D., Frank van der Stappen, A., Overmars, M.H.: An Effective Framework for Path Planning amidst Movable Obstacles. In: WAFR (2006)

[20] Ota, J.: Rearrangement Planning of Multiple Movable Objects. In: ICRA (2004)

[21] Siméon, T., Laumond, J.P., Cortés, J., Sahbani, A.: Manipulation Planning with Probabilistic Roadmaps. IJRR (23) (2004)

[22] Solovey, K., Halperin, D.: k-Color Multi-Robot Motion Planning. In: WAFR (2012)

[23] Stilman, M., Kuffner, J.J.: Planning Among Movable Obstacles with Artificial Constraints. In: WAFR (2006)

[24] Stilman, M., Schamburek, J., Kuffner, J.J., Asfour, T.: Manipulation Planning Among Movable Obstacles. In: ICRA (2007)

[25] Sundaram, S., Remmler, I., Amato, N.M.: Disassembly Sequencing Using a Motion Planning Approach. In: ICRA, pp. 1475–1480 (2001)

[26] Wagner, G., Kang, M., Choset, H.: Probabilistic Path Planning for Multiple Robots with Subdimensional Expansion. In: ICRA (2012)

[27] Wilfong, G.: Motion Planning in the Presence of Movable Obstacles. In: Annual Symp. of Computational Geometry, pp. 279–288 (1988)

[28] Yu, J., LaValle, S.M.: Multi-agent Planning and Network Flow. In: WAFR (2012)

[29] Zucker, M., Ratliff, N., Dragan, A., Pivtoraiko, M., Klingensmith, M., Dellin, C., Bagnell, J.A., Srinivasa, S.S.: CHOMP: Covariant Hamiltonian Optimization for Motion Planning. IJRR (2013)