Planning Representations and Algorithms for Prehensile Multi-Arm Manipulation

Andrew Dobson Kostas E. Bekris

***Abstract*— This paper describes the topology of general multi-arm prehensile manipulation. Reasonable assumptions are applied to reduce the number of manipulation modes, which results in an explicit graphical representation for multi-arm manipulation that is computationally manageable to store and search for solution paths. In this context, it is also possible to take advantage of preprocessing steps to significantly speed up online query resolution. The approach is evaluated in simulation for multiple arms showing it is possible to quickly compute multi-arm manipulation paths of high-quality on the fly.**

I. Introduction

Many robotic applications include multiple arms in the same workspace, which are centrally coordinated to transfer individual objects to desirable configurations using grasping, as in Fig. 1. The focus is to codify the topology of prehensile multi-arm manipulation problems [7], [20], [21] in a way that allows for efficient integrated task and motion planning [4], [14], [16]. This problem is challenging since the dimensionality of the space increases quickly with the number of arms. Furthermore, it is necessary to identify: (a) which arms are required and in what order are they needed; (b) when regrasping is required and where to place the object; (c) the sequence and locations of handoffs, and the necessary grasps to achieve them. For challenging instances, multiple arms, regrasps and handoffs may be needed to transfer an object across the workspace. This paper deals with these geometric and combinatorial aspects of multi-arm manipulation.

The first contribution of this paper is to provide the topology of prehensile manipulation for n static arms, where $1 \leq k \leq n$ of them are needed to transfer an object. This generalizes the "manipulation graph" of Fig. 2, which accounts for the topology of single-arm manipulation [25] and stems from earlier efforts [7], [20], [21]. Only recently has the topology of dual-arm manipulation been explicitly defined [13]. The current paper extends these efforts to the multi-arm case. The nodes of the proposed "multi-arm manipulation graph" ($\mathcal{G}_{\texttt{MAM}}$) are high-level states, where the object is grasped at a stable pose or a handoff takes place. The edges correspond to arm paths, where the object is transferred or the arms move to grasp it. The edges correspond to the "modes" of manipulation and increase combinatorially with the number of arms.

This work restricts the number of modes used, given access to: (i) the number k of arms needed to transfer an object, and (ii) the feasible handoffs given the spatial arrangement of the arms in the workspace. The size of the resulting $\mathcal{G}_{\texttt{MAM}}$ is shown to be computationally manageable for several arms and can be constructed explicitly. A solution is generated by searching this graph, where each edge transition corresponds to a motion planning challenge in a particular manipulation mode.

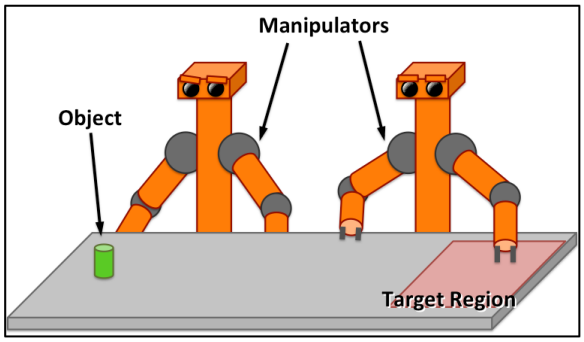

Fig. 1. An example with $n = 4$ arms. In this setup, no arm can reach both the object's starting pose and the target region.

Existing efforts generally do not explicitly construct the topology for multi-arm manipulation. Early work performed an implicit transfer-transit path search given simplifications, such as not allowing regrasps at stable poses [21]. Alternatives build a large sampling-based roadmap by composing graphs for each arm [11], [12]. A recent effort employs heuristic search to first plan an unconstrained object path, which informs the search for the arms' paths [6], but this is limited to light objects transferable by a single arm, and does not allow regrasps at stable poses. It can however be used as a heuristic in the proposed framework. Another approach deals with the multi-modal nature of manipulation, but focuses on complex plans for a single arm [2].

The benefit of constructing $\mathcal{G}_{\texttt{MAM}}$ is that it allows for useful preprocessing, which is the second contribution of this work. Given the arms' base locations, the geometry of the object and static obstacles, it is possible to precompute stable poses, arm grasps and corresponding arm paths which are invariant of specific tasks. The work computes expected path distance between high-level states, and uses it as a heuristic during query resolution. Providing good heuristics is useful primitive for manipulation task planning [10].

The evaluation of the approach includes simulations for multiple Baxter arms working in the same space. The experiments indicate that the framework solves problems with competitive running times, returning high-quality paths.

Work by the authors has been supported by NSF CCF:13307893, NSF IIS:1451737 and a DHS scholarship to Andrew Dobson. Any conclusions expressed here are of the authors and do not reflect the views of the sponsors.

Computer Science, Rutgers University, NJ, USA, {ajd223,kostas.bekris}@cs.rutgers.edu.

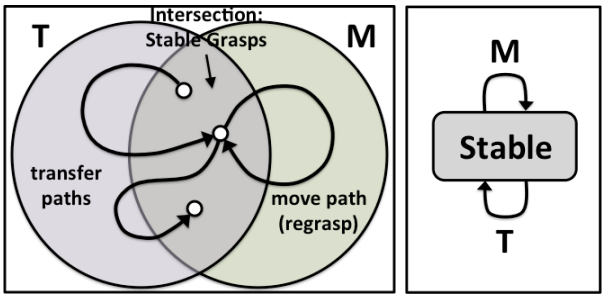

Fig. 2. Single-arm manipulation graph [25]. *Left:* For the T set, the object is transferred while grasped. For the M set, the arms move and the object rests at a stable pose. At the intersection, the object is grasped at a stable pose. *Right:* Useful paths for manipulation bring the object back to a stable grasp pose.

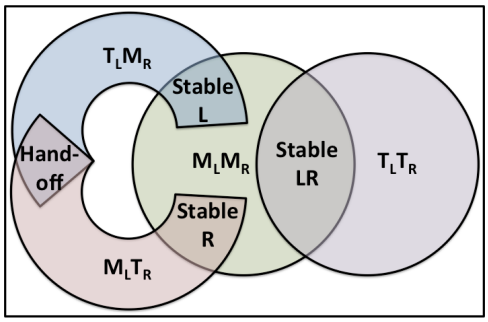

Fig. 3. The representation for dual-arm manipulation has four sets of configurations in the general case. Note that M_LM_R and T_LT_R correspond directly to M and T for single-arm manipulation (Figure 2 Left).

II. Problem Setup, Notation and Foundations

The workspace contains n robot arms $\{\mathcal{A}_1, \ldots, \mathcal{A}_n\}$, where each have their own configuration space $\mathcal{C}_{\mathcal{A}_i}$. Each manipulator has a fixed base and is assumed to move in a kinematic fashion, since the focus of the work is not arm dynamics. There is a single rigid-body object, o, with its own configuration space $\mathcal{C}_o \subset SE(3)$. Then, the configuration space for the entire problem is:

$$\mathcal{C} = \prod_{i=1}^{n} \mathcal{C}_{\mathcal{A}_i} \times \mathcal{C}_o.$$

The collision-free subset $\mathcal{C}_{free} \subset \mathcal{C}$ includes two sets, which allow contacts:

a) *stable configurations* $\mathcal{C}_s$: the object is at a stable pose, where o is in contact with static obstacle geometry, and its pose does not change while no arm acts upon o;

b) *grasping configurations* $\mathcal{C}_G$: these correspond to one or more arms grasping the object.

Let k denote the number of arms required to move the object o, where $1 \leq k \leq n$. The value of k depends on the o; for instance, o may be heavy and require multiple arms to be transferred. A valid path $\tau : [0, 1] \rightarrow \mathcal{C}_{free}$ is a continuous sequence of stable and grasping configurations, where transitions happen via stable poses or object handoffs.

Definition 1 (Multi-Arm Manipulation Problem): Given an initial configuration $q_{init} \in \mathcal{C}_s$ and a goal configuration $q_{goal} \in \mathcal{C}_s$, compute a valid path $\tau : [0, 1] \rightarrow \mathcal{C}_{free}$, such that $\tau(0) = q_{init}$ and $\tau(1) = q_{goal}$.

The traditional representation for a single arm interacting with an object consists of two modes:

a) transfer configurations T, where the manipulator is grasping and transferring the object, and

b) move configurations M, where the manipulator is moving through the space without holding the object. While typically referred to as "transit" configurations, the term moved is used to follow the M vs. T notation.

This gives rise to the graphs of Fig. 2, where a transfer is a trajectory through T connecting different stable poses, and a regrasp goes through M from a stable pose to itself.

This representation was only recently generalized for dual-arm robots [13], which is shown in Fig. 3, given the notation used here. Reasoning about the topology of the problem allows to identify whether using both hands is helpful by integrating grasp and object placement planners. In the n-arm case, the planning problem has 2^n modes, and each corresponds to different combinations of arms either moving or transferring. For example, the mode $\mathrm{M}_L\mathrm{M}_R$ corresponds to both arms moving without grasping the object, while $\mathrm{T}_L\mathrm{T}_R$ has both arms transferring the object simultaneously.

In the two arm case, a new mode corresponding to hand-offs arises. A hand-off is a transition where the arm - or, in general, a set of arms - which grasps the object changes while the object is not necessarily in a stable pose. This requires ensuring the object is not dropped; however, this is a physics problem which is not the focus of this work. Next, this representation is generalized for n arms while pruning away redundant modes.

This work assumes access to four primitives: (i) A **stable pose generation** module provides reachable stable object configurations $q_o \in \mathcal{C}_o$; (ii) A **hand-off generation** module returns object configurations where k arms can grasp the object simultaneously; (iii) A **spatial interaction** module, which is a graph that indicates which subsets of arms can simultaneously grasp an object. This graph is referred to as a Spatial Interaction Graph (`SIG`), and is detailed in section III; (iv) A **grasping** module, which computes grasps to facilitate transfers and hand-offs.

III. Proposed Representation

This section presents the reduced representation of n-arm manipulation, providing an algorithm to explicitly construct the graph with an appropriate topology for any n and k. Unsurprisingly, the topology of the graph constructed with n and k is equivalent to one where $n' = h = \binom{n}{k}$ and $k' = 1$.

Assumption 1: There is a known, fixed number of arms, k, required to transfer the object.

For $n = 2$, k can either be 1 or 2, as illustrated in Fig. 4, noting differences with Fig. 3. For $k = 1$, it is unnecessary to have a state where both arms are transferring the object. Conversely, for $k = 2$, single-arm transfer states are infeasible. Note that for $n = k = 2$, the state representation is topologically equivalent to the single-arm case of Fig. 2. A dual representation to that of Fig. 4 is shown in Fig. 5 below.

This representation is general, and Fig 6 illustrates the case of $n = 3$ and $k = 2$.

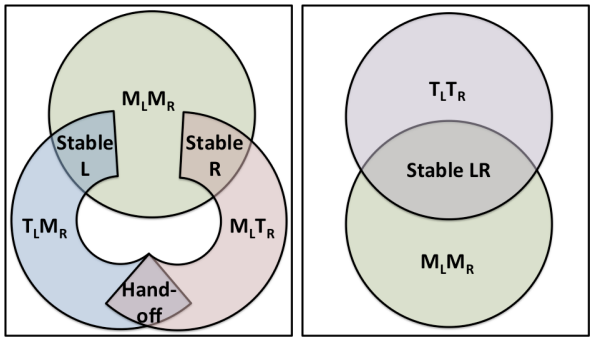

Fig. 4. The manipulation graphs for $k = 1$ (left) and $k = 2$ (right) using 2 arms: a left (L) and a right arm (R).

The general case has $h = \binom{n}{k}$ stable grasp states, and from each pair of these states, up to $\binom{h}{2}$ handoff states. This results in a total of up to $h + \binom{h}{2}$ states, with the following edges connecting them:

i) Regrasp and Transfer self-edges for each stable grasp state ($2h$ edges).

ii) Move edges between all pairs of stable grasp states ($\binom{h}{2}$ edges).

iii) Transfer edges between handoffs and their corresponding stable grasp states ($2\binom{h}{2}$ edges).

iv) Edges between pairs of handoff states which share a common subset of arms. That is, each handoff state maintains sets of arms, S_{in} and S_{out}, and there exists an edge between states which share a set ($\binom{h}{2}(h-2)$ edges).

v) Self-edges for each handoff state where (a) k arms keep the object stable and (b) k other arms perform a regrasp ($2\binom{h}{2}$ edges).

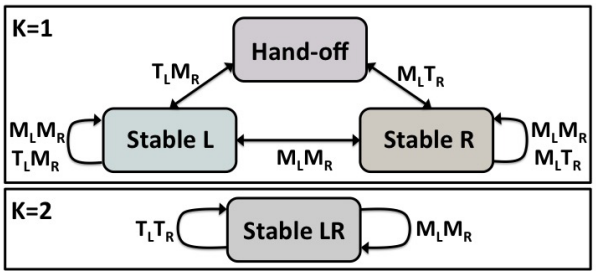

Fig. 5. *Top:* an object requiring $k = 1$ arms, corresponding to Fig. 4 (left). *Bottom:* $k = 2$, corresponding to Fig. 4 (right).

This representation significantly reduces the number of modes compared to the case that allows all combinations of arms to grasp the object. In the exhaustive case, there are $2^n - 1$ stable grasp states. Then, there are $\binom{2^n-1}{2}$ handoff states, for a grand total of $(2^n - 1) + \binom{2^n-1}{2}$ states. This complete representation would also have the same types of edges as described before. This results in a grand total of $2(2^n - 1) + 5\binom{2^n-1}{2} + \binom{\binom{2^n-1}{2}}{2}$ edges/modes. The relative benefits of the reduced representation for certain choices of n and k is provided in Fig. 7.

Assumption 1 prunes away states; however, handoffs cannot always be achieved, so another assumption is used:

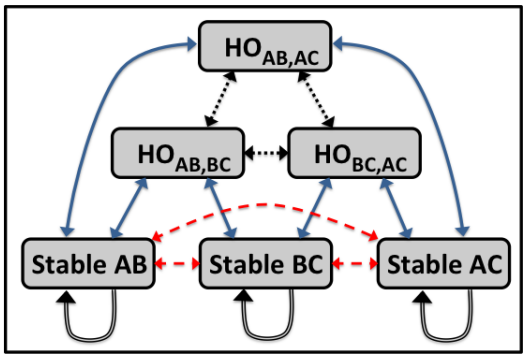

Fig. 6. Example for $n = 3$ and $k = 2$ (topologically equivalent to $n = 3$ and $k = 1$). Hallow lines represent self-transitions for stable states, light dashed lines are transitions between stable states, dotted lines are passes between different handoffs, while solid lines are transitions between stable and handoff states.

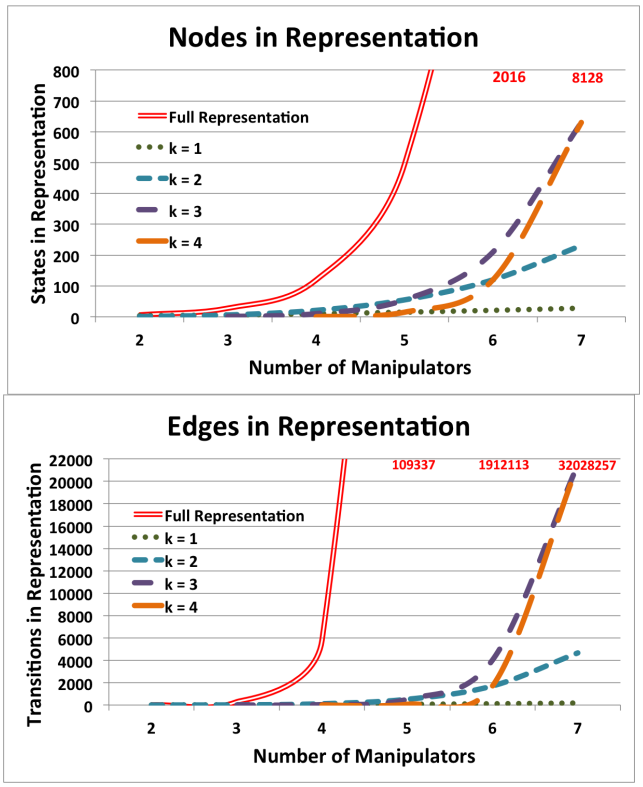

Fig. 7. A comparison between the full representation and the upper bound for the provided one. For different values of n and k, the provided graph results in considerably fewer modes.

Assumption 2: It is known which arms can potentially achieve a handoff and reach the same stable object poses.

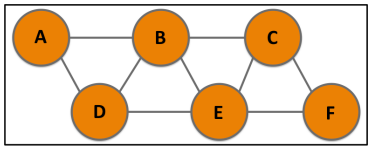

Fig. 8. An example SIG for $n = 6$ and $k = 1$.

Two input "Spatial Interaction Graphs" are assumed: a $\texttt{SIG}_h$ for feasible handoffs and a $\texttt{SIG}_s$ for transitions between stable grasps. A node in a SIG represents a set of k arms, and an edge between two nodes in $\texttt{SIG}_h$ indicates a handoff is possible, and an edge in $\texttt{SIG}_s$ indicates that there are stable object poses

reached by both sets of k arms.

An example SIG is shown in Fig. 8. Using the graph for both $\texttt{SIG}_s$ and $\texttt{SIG}_h$ with $n = 6$ and $k = 1$, there will be 15 states and 60 edges, but without the SIGs, there would be 21 states, and 117 edges, reducing the number of modes by nearly half.

An algorithm for constructing the appropriate graph is given in Alg. 1. It first generates all $h = \binom{n}{k}$ stable grasp states (Line 1). Next, the self-edges for regrasp (Line 3) and transfer (Line 4) at all stable grasp states are added.

Algorithm 1: GENERATE_AUTOMATON$(n, k, \texttt{SIG}_s, \texttt{SIG}_h)$

1 $V \leftarrow$ GENERATE_STATES(n, k); $H \leftarrow \emptyset$; $E \leftarrow \emptyset$;
2 **for** $v \in V$ **do**
3 $\quad E \leftarrow E \cup \{\texttt{EDGE}(v, v, \texttt{ALL\_MOVE})\}$;
4 $\quad E \leftarrow E \cup \{\texttt{EDGE}(v, v, \texttt{TRANS}(\texttt{ARMS}(v)))\}$;
5 **for** *pairs*$(u, v) \in V$ **do**
6 $\quad$ **if** SHARED_SURFACE$(u, v, \texttt{SIG}_s)$ **then**
7 $\quad\quad E \leftarrow E \cup \{\texttt{EDGE}(u, v, \texttt{ALL\_MOVE})\}$;
8 $\quad$ **if** HAND-OFF_POSSIBLE$(u, v, \texttt{SIG}_h)$ **then**
9 $\quad\quad h_{u,v} \leftarrow$ GENERATE_HANDOFF(u, v);
10 $\quad\quad H \leftarrow H \cup \{h_{u,v}\}$;
11 $\quad\quad E \leftarrow E \cup \{\texttt{EDGE}(v, h_{u,v}, \texttt{TRANS}(\texttt{ARMS}(v)))\}$;
12 $\quad\quad E \leftarrow E \cup \{\texttt{EDGE}(u, h_{u,v}, \texttt{TRANS}(\texttt{ARMS}(u)))\}$;
13 $\quad\quad E \leftarrow E \cup \{\texttt{EDGE}(h_{u,v}, h_{u,v}, \texttt{TRANS}(\texttt{ARMS}(v)))\}$;
14 $\quad\quad E \leftarrow E \cup \{\texttt{EDGE}(h_{u,v}, h_{u,v}, \texttt{TRANS}(\texttt{ARMS}(u)))\}$;
15 **for** *pairs*$(u, v) \in H$ **do**
16 $\quad S_u \leftarrow \texttt{ARMS}(u)$; $S_v \leftarrow \texttt{ARMS}(v)$;
17 $\quad$ **if** $S_u \cap S_v \neq \emptyset$ **then**
18 $\quad\quad E \leftarrow E \cup \{\texttt{EDGE}(u, v, \texttt{TRANS}(A_u \cap A_v))\}$;
19 return $G = (V \cup H, E)$;

Then, for every edge in $\texttt{SIG}_s$, add an edge between the corresponding stable grasp states (Lines 6-7). Next, the method adds a handoff for each edge in $\texttt{SIG}_h$ (Lines 8-10). All generated handoffs are connected to the corresponding stable grasps (Lines 11-12). The algorithm also adds self-transitions for each generated handoff (Lines 13-14). Finally, transitions between handoffs are added as long as the two handoffs include a common set of k arms (Lines 16-19).

IV. Computing a Solution Path

This work uses an informed search method, which reaps the benefit of **preprocessing**, which can be used to: (a) Produce appropriate object poses and grasps for each state of the transition diagram; (b) Construct roadmaps for each arm using generated grasps as seeds; (c) Generate heuristics for high-level search of the graph. Then, an **online search** algorithm is employed to compute plans for the arms.

A. Steps of the Preprocessing

Preprocessing is used to speed up online query resolution in a fashion that is invariant to the tasks given dring query time. There are three phases of preprocessing utilized.

Stable pose generation: A set of object poses are sampled for both stable grasps and handoffs. For each generated pose, grasps are computed using inverse-kinematics for arms which need to grasp the object at that pose. Sampling grasps for handoffs additionally requires the grasps be feasible simultaneously for all arms, and that the object is suspended so it will not be dropped.

Arm path precomputation: A roadmap is generated for each arm $\mathcal{A}_i$ in that arm's configuration space $\mathcal{C}_{\mathcal{A}_i}$. The method employed is PRM* [17], using all grasp solutions generated in the prior step as seed configurations. Each roadmap is constructed assuming no other arms are in the scene, where these roadmaps will be queried to generate plans for arms in a decoupled manner. Collision-checking for other arms and the object is deferred to the online search.

Computing heuristics: The last step generates a heuristic for the online search. A simple though inadmissible heuristic is used, which is problematic for computing bounds on search suboptimality; however, it is employed due to its simplicity. Finding appropriate heuristics is an interesting direction for future work. The cost of transitioning between high-level states in $\mathcal{G}_{\texttt{MAM}}$ is estimated and used to compute the heuristic function $h(\cdot)$. These are estimated using the precomputed roadmaps by running simplified queries which ignore other arms in the scene, averaging the longest computed paths. The transition cost is computed as:

$$c_{U,V} = avg\{\max_{S_{U,V}}(path\_cost(u, v))\}, u \in U, v \in V, u \neq v$$

where u, v are configurations in automaton states U, V.

B. Efficient Online Search of $\mathcal{G}_{\texttt{MAM}}$

This work proposes a hybrid approach to search $\mathcal{G}_{\texttt{MAM}}$ efficiently, which combines informed search principles with sampling-based planning that is tailored for continuous space search. The goal is to balance the rapid exploration properties of RRT in continuous space with the dynamic programming efficiency of A*. Searching $\mathcal{G}_{\texttt{MAM}}$ in a discrete manner is possible given the preprocessing generates a discrete set of object poses and arm grasps as well as providing a heuristic to effectively bias the search. Limiting the search over the precomputed poses and grasps affects completeness. When the method fails, it is possible to expand the set of poses/grasps generated by the preprocessing step.

The **online search** will discover a sequence of grasps and handoffs for transferring the object given the constructed automaton and precomputation. Pseudocode is provided in Algorithm 2. It follows an A*-like strategy by maintaining a priority queue of problem configurations to expand in a tree-search fashion (Line 2). Expansions are only successful when low-level motion plans can be computed.

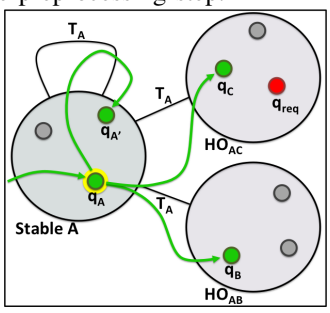

Fig. 9. Node expansion: Large disks correspond to nodes in the topological graph, while small disks to configurations. Selecting a configuration in a topological node can generate children to the reachable nodes that are reachable via edge on the topological graph.

Unlike traditional A*, the low-level primitive of expanding anode can not be trusted to produce the complete set of descendants, as this would require exhaustively computing motion plans to all adjacent configurations in $\mathcal{G}_{\texttt{MAM}}$. To this end, the proposed search method instead retains all visited nodes in the open set. A node that is selected from the open set, it is expanded, then its heuristic cost-to-go is inflated and then reinserted in the open set. In this way, the search is guaranteed to eventually re-visit nodes that have previously been expanded in an effort to discover motion plans to previously undiscovered descendants.

Algorithm 2: SEARCH(v_s, v_g)

```
1  v_s.expanded ← 0; v_s.cost ← 0;
2  P_Heap ← {v_s};
3  while goal_found = false do
4      u ← peek_min(P_Heap);
5      Children ← expand(u);
6      for o ∈ Children do
7          path ← Find_Path(u, o);
8          if path ≠ NULL then
9              if u.cost + path.cost < o.cost then
10                 o.pred ← u;
11                 o.heuristic ← (1 + o.expanded) · h(o);
12                 if o = v_g then
13                     goal_found ← true;
14                 update_cost(o);
15     u.expanded++;
16     u.heuristic ← (1 + u.expanded) · h(u);
17     reheap(P_Heap);
18 return trace_path(v_g);
```

In traditional A* fashion, the method begins by initializing the heap with the start state (Line 2); however, it will search indefinitely until a goal is found. Then each iteration the method examines the minimum heap element u and expands that node (Lines 4,5). For each child, the preprocessed roadmaps are queried to find motion plans to each child configuration (Line 7). If such a plan is found and has lower cost than the previous best cost to that child, then the child takes u as its parent (Lines 8-10), and the cost of the child's entire subtree is updated (Line 14). Unlike A*, nodes are not removed from the heap. Instead, the heuristic of the node is inflated based on the number of expansions from that configuration and the heap is re-sorted (Lines 16-17).

V. Evaluation

The proposed method greatly reduces the number of modes considered in search, and the objective of this section is to highlight the computational and solution length benefits of the proposed framework. The approach was tested in simulation using models of the Baxter robot platform in two environments, and it was compared to a general framework proposed in recent literature called Random-MMP [15]. Since Random-MMP also generates a search tree through the multi-modal search space, the method seemed appropriate to compare against. Another method in the literature performing multi-modal planning could also be considered; however, this method focuses on single-arm problems and does not address the combinatorial challenge that multi-arm manipulation poses [2].

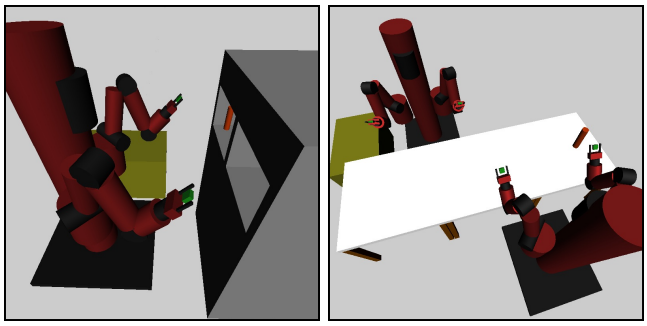

Fig. 10. The two setups used to evaluate the approach: Shelves environment with 2 arms (left) and Table environment with 4 arms (right). In both setups, the objective is to place the object on top of the yellow bin.

This paper examines two problem setups, seen in Figure 10, which are the Shelves environment and the Table environment. For Shelves, $n = 2$ and for Table, $n = 4$, where both setups use $k = 1$; furthermore, the input SIGs used in each setup are illustrated in Figure 11.

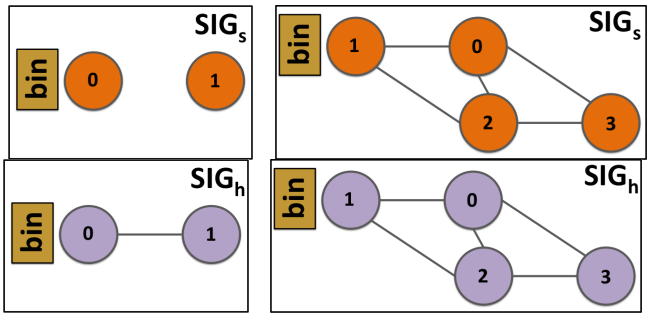

Fig. 11. The input SIGs used for the two problem setups. In the Shelves (left), the robot cannot pass the object to the other arm through a stable pose ($\texttt{SIG}_s$) but it can via a handoff ($\texttt{SIG}_h$). The Table (right) uses SIGs of the same topology; however, the two right arms of the robots cannot interact.

In the Shelves environment, the arms can perform handoff, but cannot pass the object through a stable pose. In the Table environment, all arms can interact except for arms 1 and 3. The Shelves environment is inspired by applications such as the Amazon Picking Challenge, where using both arms can potentially increase efficiency of the method. The Table environment is used as a prototypical example requiring multiple arm interactions.

The proposed search over $\mathcal{G}_{\texttt{MAM}}$ was compared with Random-MMP, and it shows competitive performance, as illustrated in Figure 12. Using comparable computation time, the proposed method generates paths up to five times shorter than Random-MMP, and significantly fewer state transitions. The A*-style search is able to appropriately leverage the precomputed heuristics to bias search in such a way to avoid redundant transitions.

Notably, the proposed method has a higher rate of timing out (after 15 minutes) for the harder problem, even though

on average it is faster than Random-MMP. This is likely due to heuristics biasing the search into a local minimum, causing the method to occasionally spend too much time exploring regions of the space which do not lend themselves to a solution. This motivates potential future work to determine what heuristics are appropriate in the context of multi-modal n-arm manipulation.

Method	$\mathcal{G}_{\texttt{MAM}}$ SEARCH		Random-MMP	
Environment	Shelves	Table	Shelves	Table
Search Time(s)	7.41s	321.45s	13.31s	379.44s
Expansions	31.12	142.43	38.78	271.45
Path Quality(s)	11.95s	17.08s	50.75s	75.09s
Transitions	4.39	5.70	10.00	14.28
Timeouts(%)	0%	9.82%	0%	1.72%

Fig. 12. Average statistics for the methods for 350+ experiments, which were solved by both methods. Search time is how long it took to find a path during online query resolution, while path quality is the duration of the solution path. Transitions is the number of high-level state transitions in the automaton. Timeout percentages are given over all attempted problem instances in each environment.

VI. Discussion

This work describes the topology of n-arm prehensile manipulation, generalizing state-of-the-art results [13], [25]. The given representation prunes redundant modes and yields a general search framework to solve multi-arm problem. A formal algorithm for constructing a manipulation graph ($\mathcal{G}_{\texttt{MAM}}$) and useful preprocessing tools are provided, and an online search method following a prior multi-modal motion planning framework is provided [15]

An important future step is evaluating and improving practical performance when $k > 1$. It is possible the greedy nature of the method causes issues with local minima, which should also be addressed. A related issue is to rigorously study the best approach to inflating heuristics in search to achieve good coverage of the search space. A possible integration with multi-robot planning methods could prove beneficial for the approach [26], [27]. The method will benefit from employing M*-style results to ensure planning for multiple arms uses the optimal decoupling.

The framework would be improved by adapting it toward mobile manipulation and non-prehensile manipulation primitives [3], [8]. Furthermore, the method could leverage caging results to practically handle the $k > 1$ case [5], [9]. The approach also needs to be made amenable to other practical issues, such as force control [24], grasp planning [1], sensing [18], [19] and reasoning over uncertainty [22], [23].

References

[1] B. Balaguer and S. Carpin. A learning method to determine how to approach an unknown object to be grasped. *International Journal of Humanoid Robotics*, 8(3):579–606, 2011.

[2] J. Barry, L. Kaelbling, and T. Lozano-Pérez. A Hierarchical Approach to Manipulation with Diverse Actions. In *IEEE Inter. Conf. on Robotics and Automation (ICRA)*, 2013.

[3] J. Barry, L. Kaelbling, and T. Lozano-Perez. A Hierarchical Approach to Manipulation with Diverse Actions. IEEE Internation Conference on Robotics and Automation, 2013.

[4] S. Cambon, R. Alami, and F. Gravot. A Hybrid Approach to Intricate Motion, Manipulation, and Task Planning. *International Journal of Robotics Research*, 28(1):104–126, 2009.

[5] P. Cheng, J. Fink, and V. Kumar. "abstractions and algorithms for cooperative multiple robot planar manipulation". In *Robotics: Science and Systems*, 2008.

[6] J. B. Cohen, M. Phillips, and M. Likhachev. Planning Single-arm Manipulations with n-Arm Robots. In *Proceedings of Robotics: Science and Systems (RSS)*, Berkeley, USA, July 2014.

[7] B. Dacre-Wright, J.-P. Laumond, and R. Alami. Motion Planning for a Robot and a Movable Object Amidst Polygonal Obstacles. In *IEEE International Conference on Robotics and Automation*, pages 2474–2480, 1992.

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

[9] J. Fink, M. Ani Hsieh, and V. Kumar. Multi-robot Manipulation via Caging in Environments with Obstacles. In *IEEE International Conference on Robotics and Automation*, 2008.

[10] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling. FFRob: An Efficient Heuristic for Task and Motion Planning. In *Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2014.

[11] M. Gharbi, J. Cortés, and T. Siméon. Roadmap Composition for Multi-Arm Systems Path Planning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2009.

[12] F. Gravot and R. Alami. A Method for Handling Multiple Roadmaps and Its Use for Complex Manipulation Planning. In *IEEE Internation Conference on Robotics and Automation (ICRA)*, 2003.

[13] K. Harada, T. Tsuji, and J.-P. Laumond. A Manipulation Motion Planner for Dual-Arm Industrial Manipulators. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 928–934, 2014.

[14] K. Hauser and J.-C. Latombe. Multi-modal Motion Planning in Non-Expansive Spaces. In *International Journal of Robotics Research (IJRR)*, volume 29, page 7, 2010.

[15] K. Hauser and V. Ng-Thow-Hing. Randomized Multi-Modal Motion Planning for a Humanoid Robot Manipulation Task. *International Journal of Robotics Research*, 30(6):678–698, 2011.

[16] L. P. Kaelbling and T. Lozano-Pérez. Hierarchical Task and Motion Planning in the Now. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[17] S. Karaman and E. Frazzoli. Sampling-based Algorithms for Optimal Motion Planning. *International Journal of Robotics Research (IJRR)*, 30(7):846–894, June 2011.

[18] D. Katz, A. Venkatraman, M. Kazemi, D. Bagnell, and A. Stentz. Perceiving, Learning and Exploiting Object Affordances for Autonomous Pile Manipulation. In *Robotics: Science and Systems (RSS)*, 2013.

[19] J. Kenney, T. Buckley, and O. Brock. Interactive Segmentation for Manipulation in Unstructured Environments. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1343–1348, 2009.

[20] Y. Koga, K. Kondo, J. J. Kuffner, and J.-C. Latombe. Planning Motions with Intentions. In *Proc. of SIGGRAPH*, pages 395–408, 1994.

[21] Y. Koga and J.-C. Latombe. On Multi-arm Manipulation Planning. In *Prof. of the IEEE Intern. Conference on Robotics and Automation (ICRA)*, 1994.

[22] M. Koval, N. Pollard, and S. S. Srinivasa. Pose Estimation for Cotact Manipulation with Manifold Particle Filters. IEEE/RSJ International Conference on Intelligent Robots and Systems, 2013.

[23] M. Koval, N. Pollard, and S. S. Srinivasa. Pre- and Post-Contact Policy Decomposition for Planar Contact Manipulation Under Uncertainty. In *Robotics: Science and Systems (RSS)*, 2014.

[24] E. Paljug, T. Sugar, V. Kumar, and X. Yun. Important Considerations in Force Control with Applications to Multi-Arm Manipulation. In *IEEE International Conference on Robotics and Automation*, pages 1270–1275, 1992.

[25] T. Siméon, J.-P. Laumond, J. Cortés, and A. Sahbani. Manipulation Planning with Probabilistic Roadmaps. *International Journal of Robotics Research (IJRR)*, 23(8):729–746, 2004.

[26] K. Solovey, O. Salzman, and D. Halperin. Finding a Needle in an Exponential Haystack: Discrete RRT for Exploration of Implicit Roadmaps in Multi-Robot Motion Planning. In *Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2014.

[27] G. Wagner, M. Kang, and H. Choset. Probabilistic Path Planning for Multiple Robots with Subdimensional Expansion. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2012.