Synchronized Dual-Arm Rearrangement

Rahul Shome[1], Kiril Solovey[2], Jingjin Yu[1], Kostas Bekris[1], and Dan Halperin[2]

[1]Rutgers University, NJ, USA and [2]Tel Aviv University, Israel

Abstract. The problem of object rearrangement arises in a variety of real world applications including warehouse automation and service robotics. The problem of packing a set of scattered objects on a table-top from their initial locations to a well-aligned target arrangement can be solved efficiently with the use of multiple arms. The current work inspects the combinatorial challenges that arise from the number of objects and arms. The proposed method provides a framework to use a set of multi-arm motion planning solutions for moving the objects to find the object assignments to arms that optimize desired metrics. The quality of the resulting solutions are evaluated against baseline approaches as well as single arm solutions to demonstrate the efficacy of efficient multi-arm rearrangement planning methods.

1 Introduction

Motivation: Automation setups where objects are picked from manipulators and need to be arranged in specific configurations, e.g., packing tasks.

Objective: Study how multiple manipulators can improve the efficiency of such automation setups. What is the improvement in solution cost in terms of desirable metrics when we use multiple arms instead of one? How difficult is the problem? What are methods that can be efficient in this context?

Preview of Related Work: i) There has been work on single-arm rearrangement that focuses either on efficiency or aims to provide some guarantees. ii) There has been work on multi-arm/robot planning but not much that reasons in terms of coordinating arms for rearrangement tasks. This work deals with the integration of the above two aspects, i.e., it aims to bring together rearrangement and multi-robot challenges.

Challenges: Even simpler challenges are hard. This is a very high-dimensional problem. Combinatorial challenges arise both from the presence of multiple arms and multiple objects. Integration of task planning aspects and multi-robot coordination.

Focus here/Assumptions: This work focuses on two manipulators that move objects where their start and goal locations do not overlap. Once the objects are picked, then there are no interactions between the arm/object system and

the other objects. We also focus on the case that the two arms operate in a synchronous manner, i.e., they pick and place objects simultaneously.

Contributions: We study this problem for different models of the underlying setup (discrete, continuous disks, general case). We propose a framework that is computationally efficient relative to an exhaustive search, while it achieves similar path quality. We show the significant benefits of using multiple arms versus an individual arm. For the proposed framework we can achieve desirable guarantees for a relaxation of the original problem.

Summary of evaluation and results: Path quality similar to that achieved by exhaustive search: almost double the quality of a single arm solution. Much faster computation compared to exhaustive.

2 Related Work

The current work dealing with dual-arm object rearrangement touches upon the challenging intersection of a variety of rich bodies of prior work. The problem of rearrangement is closely tied with multi-robot planning and coordination. The challenge with multi-body planning is the high dimensionality. Initial approaches [2] tried dimensionality reduction strategies. Optimal strategies were developed for simpler instances of the problem [35]. The unlabeled case was shown to be hard [34]. Decentralized approaches [41] also used velocity tuning [24] to deal with these difficult instances. Assembly planning [44, 17, 38] is another related problem dealing with multi-body planning.

The discrete version of the multi-body problem, also known as the "pebble motion problem" has seen a lot of work [20, 6, 3, 16]. For such discrete problems on a pebble graph, feasibility times are linear and solution times are polynomial. Optimality is still challenging even in these setups. Heuristics have been used to provide high quality solutions [42, 31].

It might be possible to cast these known hard instances of the problem as related algorithmic problems which have efficient solvers. The pebble motion problem was reduced to network flow [46, 47]. Single arm object rearrangement was also cast as a traveling salesman problem [18]. These provide the inspiration to closely inspect the nature of the problem to derive efficient solutions.

Our problem can be formulated in a variety of related classes of problems in combinatorial literature. Most of the approaches assume that all the edge weights are known, ie. the motion planning for all the edges has to be performed before any operation on the graph that uses the weights.

Formulate the Multi-robot rearrangement problem as a multi-TSP, ie. TSP for multiple agents. This can itself be tweaked with assumptions on same or different start and goal positions, symmetric or asymmetric metrics, directed or undirected graphs or posing the problem as an optimization A seminal paper [13] that has the formulation for k-TSP, with solutions that split the tour, as well as demonstration of a worst case for 2-TSP.

Prior work [28] has formulated the problem of multi-start to multi-goal k-TSP as an optimization. This assumes symmetric edge weights. Some work [14] has been done on dealing with graphs which can handle asymmetric edge weights, ie. directed graphs.

If the object transfer is seen as an errand [33] then a graph constructed with nodes being object-arm assignments and edges implying the order of coordinated execution. There has been prior work [15] that operates over the assignment graph(where pairwise assignments are nodes), an object's group consists of sets of nodes that include an object

The multi robot rearrangement can also be posed as an instance of multi vehicle pickup and delivery, where every manipulator is a vehicle and every transfer corresponds to a pickup and delivery. Most of the work has been motivated by ride-sharing applications and timed task scheduling, so there are a lot of variations that deal with time windows but this line of research provides some ILP formulations for the problem. Prior work [10] has been done that demonstrated applications on mobile robots. A prior work poses a more constraining version of our problem that takes into account time windows and robot-robot transfers. Some seminal work [23, 30] has also explored the complexity of the problem. By analyzing a survey [27] of the different problem variations, it seems we are in the "paired" category and multi-vehicle i.e., Multi vehicle pickup and delivery problem. ILP formulations [30] have been made to solve the problem. Prior work [39] has also drawn insights into the expected cost of such solutions.

Most of the works that delve into the combinatorial challenges of the problem do not account for the costs incurred due to coordination of the agents during task execution. A line of work [7] exists on Pickup and Delivery problems under track contention, which imposes constraints based on such coordination. Most of the previous work however concedes the hardness of the problem and it is difficult to find optimal, or near-optimal solutions fast.

There has been large amount of robotics research that has looked into the problem of motion planning for manipulators and movable objects. Navigation among movable objects deals with the combinatorial challenges of multiple objects [43, 8, 11, 25, 40] and has been shown to be a hard problem. This has been extended to manipulation applications [37]. Manipulators opened the applications of rearrangement planning [4, 26], including instances where objects can be grasped only once or monotone [37], as well as non-monotone instances [19, 36]. There has been a lot of recent work on dealing with the hard instances of rearrangement planning, including efficient algorithms to solve it [32, 5, 9, 18, 21, 22].

3 Problem Setup and Notation

The current work addresses the problem of rearrangement of objects resting on a plane. The following section defines the problem for n objects and 2 manipulators.

We start with several basic definitions. The workspace $\mathcal{W}$ consists of a bounded planar surface where objects can be placed. The set of n objects, $\mathcal{O} = \{o_1 \cdots o_n\}$ occupy $\mathcal{W}$ and can acquire stable poses on the surface. $\mathcal{P}^i \subset SE(3)$ corresponds to the pose-space of the object i. A pose of an object i is denoted by $p_i \in \mathcal{P}_i$. An arrangement space $\mathcal{A} = \mathcal{P}_1 \times \mathcal{P}_2 \ldots \mathcal{P}_n$ is the Cartesian product of all the pose spaces. An arrangement $A \in \mathcal{A}$ consists of a set of poses for n objects, $(p_1, \ldots p_n)$. A valid arrangement is a set of stable poses that rest on the planar surface and are pairwise collision free, i.e., the objects cannot overlap with each other. Let $\mathcal{A}^f \subset \mathcal{A}$ be the set of valid object arrangements.

The workspace is also shared by manipulators $m = \{m_1, m_2\}$. The current setup consists of 2 manipulators **Kiril:** should we state that each manipulator has a single arm?. The reachable workspace of the manipulator $m_k, k = 1, 2$ is denoted by $\mathcal{W}_R^k \subset \mathcal{W} \subset SE(3)$. A manipulator is capable of performing pick-and-place actions on the objects at the stable poses. The collision free configuration space of the manipulator m_k is $\mathbb{C}_{free}^{m_k}$ (**Kiril:** while ignoring the other manipulator and the obstacles?. A configuration $q_k \in \mathbb{C}_{free}^{m_k}$. The manipulator trajectory $\pi : [0, 1] \rightarrow \mathbb{C}_{free}^{m_k}$ is a collision free trajectory for the manipulator m_k. **Kiril:** Should we include more detailed definitions of the robot's motion capabilities here?

We are ready to define the rearrangement problem. Given the object set $\mathcal{O}$, the problem consists of computing manipulator trajectories π_1, π_2 that move the objects from an initial arrangement $(p_1^{init}, \ldots, p_n^{init}) \in \mathcal{A}^f$ to a goal arrangement $(p_1^{goal}, \ldots, p_n^{goal}) \in \mathcal{A}^f$. Of course, we require that the trajectories will not induce collisions between the two manipulators, the stationary objects, and the objects being in transit. **Kiril:** Should we make it more formal? We make a few simplifying assumptions to make the problem more manageable.

1. For any object $o_i \in \mathcal{O}$, manipulator $k \in m$, and two arrangements

$$(p_1, \ldots, p_i, \ldots, p_n), (p_1, \ldots, p_i', \ldots, p_n) \in \mathcal{A}^f$$

   there exists a trajectory for the manipulator that picks o_i and moves it from p_i to p_i', without inducing collisions with the other stationary objects. This also implies that all valid poses of the objects are reachable by both manipulators.
2. **Kiril:** Need to add here assumption about the input, i.e., that the starts and targets do not overlap.

Kiril: It seems to me that the current notation that is described below (soma, coma, etc.) is quite complex, and I think that we can simplify it. Let me suggest the alternative notation: the (asynchronous) problem consists of finding two single-arm trajectories π^1, π^2, one for each arm, which move the objects from initial to goal arrangements. The notation (π_1, π_2) represents a dual-arm trajectory, in which π_1, π_2, are executed simultaneously by the two arms. Given a single-arm trajectory π denote by $o(\pi) \rightarrow 2^{\mathcal{O}}$ the set objects that are manipulated during the execution of π. That is, given that $o(\pi) = \{i_1, \ldots, i_k\}$ before

the execution of π each object o_{i_j}, for $1 \leq j \leq k$, has resided in $p_{i_j}^{initial}$, whereas after the execution each such object is located in $p_{i_j}^{goal}$.

The synchronous problem consists of finding a sequences of dual-arm trajectories in which the two arms pick and place object while in sync, i.e., if one is picking an object, then the second arm should be picking another object (and vice versa). Let us formally define this. A synchronous solution is denoted by

$$\Pi = \left((\pi_1^1, \pi_1^2), (\pi_{1\to 2}^1, \pi_{1\to 2}^2), (\pi_2^1, \pi_2^2), \ldots, (\pi_{\ell-1\to\ell}^1, \pi_{\ell-1\to\ell}^2), (\pi_\ell^1, \pi_\ell^2)\right).$$

A dual arm trajectory of the form (π_i^1, π_i^2) represents a *transfer* trajectory in which every arm begins its motion with a contact with one specific object that is located in its initial position, and ends this motion (while still in contact with the same object), only that now its located in its goal location. Moreover, every arm manipulates a single distinct object at a time, i.e., $o(\pi_i^1) = \{j_i^1\}, o(\pi_i^2) = \{j_i^2\}$, where $i^1 \neq i^2$.

We now describe *transit* trajectories of the $(\pi_{i-1\to i}^1, \pi_{i-1\to i}^2)$. Arm k executing $\pi_{i-1\to i}^k$ brakes contact with object j_{i-1}^k (which is now located in its goal location) and moves until establishing contact with object j_i^k. In general, no object is manipulated throughout this motion and no other object is touched throughout the execution.

A single object-manipulator assignment (`soma`) is a pair

$$\texttt{soma}(k, i) = [m_k, o_i] \ s.t. \ k \in \{1, 2\}, o_i \in \mathcal{O} \cup \emptyset,$$

which represents the action of m_k picking o_i from p_i^{init} and placing it in p_i^{goal}. The path $\pi_{[m^k, o^i]}$ corresponds to a manipulator trajectory that executes the `soma`.

A complete object-manipulator assignment (`coma`) is a pair of `soma` , one for each manipulator

$$\texttt{coma}(i, j) = \{\texttt{soma}_1^i, \texttt{soma}_2^j\} = \{[m_1, o_i], [m_2, o_j]\} \ s.t. \ o_i, o_j \in \mathcal{O} \cup \emptyset, o_i \neq o_j$$

Two configuration sets, one for the initial picking configurations and the other for the final placing configurations

$$S(i, j) = S(\texttt{coma}(i, j)) = \{q_{init}^{[m_1, o_i]}, q_{init}^{[m_2, o_j]}\}$$

$$T(i, j) = T(\texttt{coma}(i, j)) = \{q_{goal}^{[m_1, o_i]}, q_{goal}^{[m_2, o_j]}\}$$

An ordered sequence of `coma` corresponds to a `scoma`

$$\texttt{scoma} = \{\texttt{coma}_1, \texttt{coma}_2 \ldots \texttt{coma}_l\}$$

where the index indicates the order of execution, ie. $\texttt{coma}_i$ is executed before $\texttt{coma}_{i+1}$ and each object o_i appears only once in any of the `coma` .
Synchronization assumption : The picks and places a synchronized for all the objects.

3.1 Cost Metrics

Define the cost metrics that we care about: a) sum of distances traveled by the end effector to solve the problem or b) the makespan of the process (under a constant velocity assumption, this is analogous to the sum of maximum distance traveled by an end effector during each synchronized operation).

The synchronized transfer by the two manipulators on a pair of objects o_i, o_i can be executed with a valid trajectory defined as follows

$$\pi_{\texttt{coma}(i,j)} : [0,1] \rightarrow \mathbb{C}^{m_1}_{free} \times \mathbb{C}^{m_2}_{free}$$

$$\pi_{\texttt{coma}(i,j)}[0] = S(\texttt{coma}(i,j)), \quad \pi_{\texttt{coma}(i,j)}[1] = T(\texttt{coma}(i,j))$$

The synchronized transit of the two manipulators represent a transition between $\texttt{coma}_l \rightarrow \texttt{coma}_{l+1}$, corresponding to say $\texttt{coma}(i_1, j_1) \rightarrow \texttt{coma}(i_2, j_2)$.

$$\pi_{\texttt{coma}_l \rightarrow \texttt{coma}_{l+1}} : [0,1] \rightarrow \mathbb{C}^{m_1}_{free} \times \mathbb{C}^{m_2}_{free}$$

$$\pi_{\texttt{coma}_l \rightarrow \texttt{coma}_{l+1}}[0] = T(\texttt{coma}_l), \quad \pi_{\texttt{coma}_l \rightarrow \texttt{coma}_{l+1}}[1] = S(\texttt{coma}_{l+1})$$

Such a trajectory represent the manipulators moving without the object from the target configuration of the first $\texttt{coma}_l$ to the initial configuration of the second $\texttt{coma}_{l+1}$.

Each trajectory π_{full} *st.* $\pi_{full}[i] \in \mathbb{C}^{m_1}_{free} \times \mathbb{C}^{m_2}_{free}$ can be decomposed into the constituent arm trajectories

$$\pi^1 \ \textit{st.}\ \pi^1[i] \in \mathbb{C}^{m_1}_{free} \ \ \textit{and} \ \ \pi^2 \ \textit{st.}\ \pi^2[i] \in \mathbb{C}^{m_2}_{free} \ \ \forall i \in [0,1]$$

Maximum of Distances Similar to the *MAKESPAN* of the solution, the maximum of distances metric tries to ensure that the distance traveled along the longest route for any of the manipulators is minimized.

$$\begin{aligned} \text{cost}(\texttt{coma}_l) &= \text{cost}(\pi_{\texttt{coma}_l}) \\ &= \max\{\|\pi^1_{\texttt{coma}_l}\|, \|\pi^2_{\texttt{coma}_l}\|\} \end{aligned}$$

$$\begin{aligned} \text{cost}(\texttt{coma}_l \rightarrow \texttt{coma}_{l+1}) &= \text{cost}(\pi_{\texttt{coma}_l \rightarrow \texttt{coma}_{l+1}}) \\ &= \max\{\|\pi^1_{\texttt{coma}_l \rightarrow \texttt{coma}_{l+1}}\|, \|\pi^2_{\texttt{coma}_l \rightarrow \texttt{coma}_{l+1}}\|\} \end{aligned}$$

Given a valid sequence of $\texttt{coma}$ or a $\texttt{scoma} = \{\texttt{coma}_1, \cdots \texttt{coma}_l \cdots \texttt{coma}_L\}$

$$\text{cost}(\texttt{scoma}) = \text{cost}(\texttt{coma}_l) + \sum_{l=1}^{L} \text{cost}(\texttt{coma}_l) + \text{cost}(\texttt{coma}_l \rightarrow \texttt{coma}_{l+1})$$

3.2 Problem Statement

Define a schedule for the two arms to simultaneously move the N objects from their start poses to their goal ones, as well as paths that the arms should follow, so as to minimize the corresponding cost metric. There exists a

$$\texttt{scoma}^* = \underset{\texttt{scoma}}{\texttt{argmin}}\ \text{cost}(\texttt{scoma})$$

and the corresponding cost is $c^* = cost(\texttt{scoma}^*)$. The synchronized dual-arm rearrangement problem consists of finding $\texttt{scoma}^*$.

4 Baseline Approaches

4.1 Exhaustive Search Solution

The baseline optimal search is a brute force expansion of all possible object assignments to manipulators. A node in the search tree corresponds to the task assignment set `coma`. Each branch of the tree corresponds to the sequence of `coma`. At every level, all combinations of remaining object assignments to manipulators give rise to the branching factor of this tree.

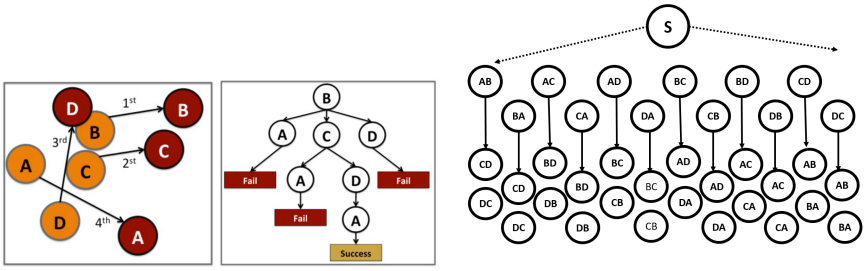

Fig. 1. Search trees in the single and dual arm rearrangement problems for 4 objects.

This exhaustive backtracking search is demonstrated for $k = 1, 2$ in Fig 1. For $k = 2$, such a tree will move 2 objects at every node, and thereby has depth $n/2$. The number of choices at any level l would be ${}^{n-2l}P_2$. The total number of nodes in this tree would look like

$$\begin{aligned} {}^{n}P_2 + {}^{n}P_2 \times {}^{n-2}P_2 + ...+ \\ {}^{n}P_2 \times {}^{n-2}P_2 \times {}^{n-4}P_2 ... \times {}^{2}P_2 \end{aligned} = \sum_{l=1}^{n/2} \prod_{k=0}^{2l-1} (n-k)$$

which is in the order of $(n!)$.

The search assumes that the arms are synchronized and every time that they transfer two objects they simultaneously complete their motions when they have placed the objects at their goals.

Traversing an edge of the search tree corresponds to solving a multi-robot motion planning problem (e.g., using dRRT* [12]) where the arms have to move from the target configurations of the objects of the parent node to the initial configurations of the objects at the child node and then transfer the child objects to their goal configurations. For instance, in Fig 1, traversing the edge $(AB) \rightarrow (DC)$ means that arms start at a configuration where the left arm has just placed object A at its goal and the right arm has just placed object B at its goal. Then, the arms move so as to grasp objects D and C, respectively, at their initial configuration and transfer them to their target configuration.

A benefit of this representation is that it is relatively general (e.g., no assumptions about object or scene geometries). It can discover the optimal "synchronized" solution to the defined rearrangement problem. Furthermore, it does not require that the arms return to an initial or safe configuration after placing every object. It also allows the discovery of optimal coordinated solution.

4.2 MILP Formulation

Mixed Integer Linear Programming (MILP) formulations provide useful, widely used optimization tools for a wide variety of problems that can be modeled for MILP. Prior work has used these techniques for solving m-TSP [13][28][14] and pickup-and-delivery problems [10][30]. The novelty of the problem at hand stems from the consideration of interaction costs of the robots. This makes the solution to the dual-arm rearrangement problem significantly more coupled, and the resulting models larger and more complex. The following section outlines an MILP formulation for the synchronized dual-arm rearrangement problem.

Graph Representation: The problem can be represented as a directed graph where each vertex v corresponds to a $\texttt{coma}_v$, ie. complete assignments of objects to both manipulators wherein the manipulators perform the transfers of the objects in the coma. Edges connect all valid transitions between pairs of coma. These correspond to the transits of both the manipulators. A valid transition is one where an object does not appear more than once. Let us denote the set of objects involved in $\texttt{coma}_1(i_1, j_1)$ and $\texttt{coma}_2(i_2, j_2)$ as $\lceil \texttt{coma}_1, \texttt{coma}_2 \rceil = \{o_{i_1}, o_{j_1}, o_{i_2}, o_{j_2}\}$.

In order to aid in the model formulation a home vertex h is added and a 0 cost connection is added to every node on the graph. The directed graph, $\hat{G}$ is thereby defined as follows:

$$\hat{G}(\hat{V}, \hat{E})$$
$$\hat{V} = \{v = (o_i, o_j) \;\; \forall o_i, o_j \in \mathcal{O}, i \neq j\} \cup \{h\}$$
$$\hat{E} = \{e(u, v) \; \forall u, v \in \hat{V} \; st. \; u \neq v, o_i \neq o_j \forall o_i, o_j \in \lceil \texttt{coma}_u, \texttt{coma}_v \rceil\}$$
$$\cup \{e(h, v) \; \forall v \in \hat{V}\} \cup \{e(v, h) \; \forall v \in \hat{V}\}$$

The cost of an edge encodes both the transfer and transit costs.

$$\begin{aligned} c_e &= \text{cost}(u) + \text{cost}(u, v) \\ &= \text{cost}(\texttt{coma}_u) + \text{cost}(\texttt{coma}_u \rightarrow \texttt{coma}_v) \end{aligned}$$

These are pre-computed to take into account the multi-robot interactions that arise from moving both manipulators at the same time.

In such a graph, $\texttt{scoma}^*$ will be contained in a tour starting from h, while traversing each vertex corresponding to the $\texttt{scoma}^*$ in order. This will correspond to the least cost such tour that exists on the graph.

The formulation needs the definition of the following sets, $\delta_{in}(v)$ as the in-edge set of a vertex, $\delta_{out}(v)$ as the out-edge set, and $\gamma(o)$ as the subset of edges, $e(u, v)$ in $\hat{E}$ such that $o \in u$.

$$\delta_{in}(v) = \{e(u, v)\ st.\ e \in \hat{E}, u \in \hat{V}\}$$
$$\delta_{in}(v) = \{e(v, u)\ st.\ e \in \hat{E}, u \in \hat{V}\}$$
$$\gamma(o) = \{e(u, v)\ st.\ e \in \hat{E}, o \in \texttt{coma}_u\}$$

$$\min \sum_{e_\in \hat{E}} c_e x_e \tag{1}$$
$$x_e \in \{0, 1\} \quad \forall e \in \hat{E} \tag{2}$$
$$\sum_{e \in \delta_{in}(v)} x_e \leq 1 \quad \forall v \in \hat{V} \tag{3}$$
$$\sum_{e \in \delta_{out}(v)} x_e \leq 1 \quad \forall v \in \hat{V} \tag{4}$$
$$\sum_{e \in \delta_{in}(v)} x_e = \sum_{e \in \delta_{out}(v)} x_e \quad \forall v \in \hat{V} \tag{5}$$
$$\sum_{e \in \delta_{in}(h)} x_e = 1 \tag{6}$$
$$\sum_{e \in \delta_{out}(h)} x_e = 1 \tag{7}$$
$$\sum_{e \in \gamma(o)} x_e = 1 \quad \forall o \in \mathcal{O} \tag{8}$$
$$\sum_{e(u,v) \forall u,v \in S, u \neq v} x_e \leq |S| \quad \forall S \subset \hat{V}, |S| \leq \frac{n}{2} \tag{9}$$

Model. 1

Model: The problem of discovering $\texttt{scoma}^*$ can now be formulated as an MILP. Indicator variables x_e are defined for every edge e. In Model 1, Line 1 represents the optimization objective wherein the set of edges corresponding to the optimal $\texttt{scoma}^*$ tour would be activated. The subsequent lines express the constraints that have to be imposed to satisfy the structure of the synchronized dual-arm arm rearrangement problem.

Line 2 specifies that the indicator variable can only attain values 0 or 1. Line 3 and Line 4 specifies that the in-degree and out-degree for any vertex can be either 0 or 1. Line 5 imposes the additional flow constraint such that a vertex has to be part of directed tour or not be part of it, ie. the in-degree and out-degree are conserved. Line 6 and Line 7 ensures that the home vertex h is part of the tour. It is important to note that the edges in and out of h do not affect the final solution. The constraint in Line 8 guarantees that an object o is transfered only once, ie. out of all the `coma` that can potentially transfer object o, only one can be part of the tour. Line 9 represents the subtour elimination constraint. This is applied lazily on solution tours to ensure that the solution is a tour of length $\frac{n}{2} + 1$.

Algorithm 1: `Lazy-MILP`$(\mathcal{O}, m, A_{init}, A_{goal})$

```
1  Ĝ(V̂, Ê) ← Construct(O, m, A_init, A_goal);
2  AssignHeuristicCosts(Ĝ);
3  converged ← False;
4  E_cand ← ∅;
5  while not converged do
6  |   AssignCoordinationCosts(E_cand, Ĝ);
7  |   E_ilp ← ILPSolver(Ĝ);
8  |   converged ← E_ilp == E_cand;
9  |   E_cand ← E_ilp;
10 Π* ← trace(E_cand);
11 return Π*;
12 b
```

It is possible that the MILP can be evaluated lazily, as in Algo 1. This updates the model with more accurate motion planning costs on top of heuristically obtained feasible candidate solutions. This presents an anytime algorithm that incrementally performs expensive motion planning only on candidate solution edges on $\hat{G}$. The method keeps updating the costs on the graph till the optimal solution containing `scoma`* is obtained. Line 1 constructs the graph $\hat{G}$. Line 2 assigns the heuristic costs to the edges of the graph. The algorithm loops till convergence to optimal. Line 6 performs coordinated motion planning over the candidate solution edge set $\mathcal{E}_{cand}$. The `ILPSolver` executes the optimization represented by Model 1. If the previous candidate solution is the same as the current solution, then the model has converged and Line 10 reports the traced solution trajectory corresponding to $\mathcal{E}_{cand}$.

5 Efficient Approximation via Edge Matching

The challenge of the dual-arm rearrangement problem stems from the interactions of the two manipulators and the costs incurred in coordinating them during

the synchronized transfers and transits. In the worst case, the expensive step of motion planning has to be performed for every edge in the search tree for exhaustive search, and every edge in $\hat{G}$ in the MILP formulation. The method proposed in this section attempts to provide high quality solutions to the problem while reducing the number of motion planning queries necessary to discover such a solution.

We can build a graph with nodes equal to the number of objects(Fig 2). Consider a weighted directed graph $G(V, E)$, where V corresponds to the set of objects O. Each directed edge, $e = (u, v, w) \in E$ is an ordered pair of nodes(objects), where the order determines the assignment of an object to an arm, and a weight or cost for the simultaneous dual-arm execution. The edge corresponds to the best available coordinated path of transferring the corresponding objects with the two arms in terms of the optimization objective (e.g., minimizing distance or delay, etc.) For example, $e = (A, B, w_{AB})$ corresponds to the left arm transferring A, the right arm moving B, and the cost of such a concurrent motion being w_{AB}. Note that all such motions start at the home position, and end at the home position, for both arms.

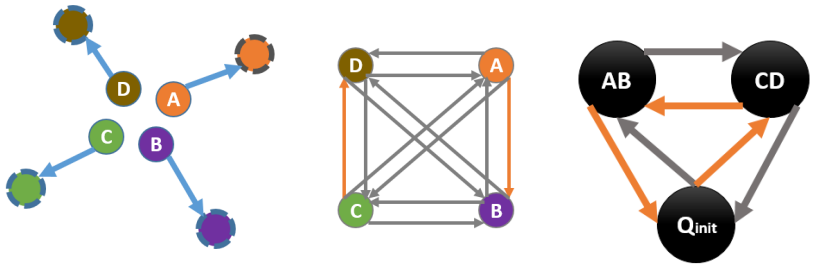

Fig. 2. The image shows an instance of a dual-arm monotone problem where none of the object placements impose constraints. The image in the middle represents a formulation of the problem as minimum weight edge matching on a fully connected directed graph. The image on the right represents the second stage of the method that solves the ordering of the object-arm assignments by solving an instance of TSP.

The method introduces a decomposition of the cost metric: a) first optimize object transfer by the two arms, i.e., solve the assignment problem first and then b) optimize the schedule for the object assignment.

Here we will describe the two-step approach: a) weighted edge matching to get the assignment that provides the optimal transfers and b) solve the TSP given the assignments so as to minize transit costs.

Each pair of objects is connected by two edges. For each pair of objects, only one lower cost edge, that optimizes the corresponding objective edge is sufficient to be considered in the graph. Then, on this (general) graph we solve a

Minimum Weight Perfect Matching (MWPM) problem using *Edmond's Blossom* algorithm.

In an unconstrained setup, Fig 2(left), where each object movement can be done without dependencies and constraints on other object motions, a minimum weight edge matching on G, corresponds to the optimal solution to the problem.

As a part of the proposed solution outlined in Algorithm 2, the approach imposes certain assumptions to the setup. The method commits to fixed solutions $\pi_{[m^x, o^x]}$ for every object transfer task assignment to a manipulator. Additionally, all such solutions π would begin at a valid home configuration $q^{m^x}_{home} \in \mathbb{C}^{m^x}_{free}$ for the manipulator m^x, pick up the object o^x, transfer the object and finally return to $q^{m^x}_{home}$.

The space of possible solutions is cast into a graphical representation, shown in Fig 2. For the sake of simplicity, the specification of such a graph is demonstrated for $k = 2$, though such the approach is general to hypergraphs for $k > 2$.

A rearrangement graph $G(V, E)$ is defined such that $V = \mathcal{O}$. For $k = 2$, $\mathcal{M} = m_1, m_2$. A directed edge $e = [o_i, o_j] \in E$ corresponds to the set of task assignments $\texttt{coma}_{e(i,j)} = \{[m_1, o_i], [m_2, o_j]\}$.

An equivalent undirected graph is obtained by keeping track of the minimum weighted edge for all $u, v \in V$ and keeping $\underset{e}{argmin}\ e = \{e(u, v), e(v, u)\}$.

The solution to a minimum weighted edge matching problem on $G(V, E)$ yields $E^* \subseteq E'$. On the set of object arm assignments a TSP provides the optimal schedule.

Algorithm 2: `Edge-Matching-TSP`$(\mathcal{O}, m, A_{init}, A_{goal})$

$G(V, E)$;
$V \leftarrow \mathcal{O}$;
for $o_i, o_j \in \mathcal{O}$ *s.t.* $i \neq j$ **do**
  $w_{(i,j)} \leftarrow \infty$;
for $o_i, o_j \in \mathcal{O}$ *s.t.* $i \neq j$ **do**
  $w_{(i,j)} \leftarrow \text{cost}(\texttt{coma}(i, j))$;
  if $w_{(i,j)} < w_{(j,i)}$ **then**
    $E \leftarrow E \cup e(i, j, w_{(i,j)})$;
$E^* \leftarrow \texttt{EdgeMatching}(G)$;
$G_{tsp}(V_{tsp}, E_{tsp})$;
$V_{tsp} \leftarrow V_{tsp} \cup \texttt{coma}_e \forall e \in E^*$;
$V_{tsp} \leftarrow V_{tsp} \cup \texttt{coma}_{home}$;
for $\texttt{coma}_u, \texttt{coma}_v \in V_{tsp}$ *s.t.* $u \neq v$ **do**
  $w_{(u,v)} \leftarrow \text{cost}(\texttt{coma}_u, \texttt{coma}_v)$;
  $E_{tsp} \leftarrow E_{tsp} \cup e(u, v, w_{(u,v)})$;
$\mathcal{T} \leftarrow \texttt{TSP}(G_{tsp})$;
$\Pi^* \leftarrow \texttt{trace}(\mathcal{T})$;
`return` Π^*;

6 Integration with Motion Planning

We will study this problem under three different models:

- a completely discrete setup where the objects and the robot occupy nodes of a graph;
- a continuous setup where robots are disks on a plane and the objects exist on a planar surface;
- and the more general case of standard manipulators.

Potentially analogous problems- *Errand scheduling* : The model might be cast as an errand scheduling problem. *Pickup and delivery* : The model might be cast as a pickup and delivery problem.

6.1 Discrete Case

Assume we have a graph, where each object can occupy a node and an arm can also occupy nodes. We do not care about collisions involving objects. We care only about collisions between arms.

Path planning component: an optimal multi-robot planner on a discrete representation (e.g., M*). Combined with the exhaustive search for the scheduling, this would give the optimum solution. An ILP can also be applied.

Potential questions to address:

1. Can we have an approach that is faster than exhaustive search in the discrete case that addresses the challenge (e.g., does the "errand scheduling" formulation help here)?
2. What is the worst-case performance of the two arms solution relative to the one arm case? i.e., can we have a bound that is better than 1 for the ratio $\frac{\text{2 arm solution cost}}{\text{single arm solution cost}}$ in the worst-case.
   - Can some solution quality arguments be made for a single robot case on such a discrete graph?
   - Then, can some solution quality arguments be made for a 2 robot case on a discrete graph? What is the cost of coordination?
   - What is the relationship to 2-TSP? Can a bound be argued for 2-TSP under this setup?
   - Can "bad" examples be designed to show the worst case performance and arguments made that this is not as common or performance is not that bad? For example, if the starts and goals are in small discrete, tight groups?
3. We can potentially consider a grid-based representation that allows to encode the size of the robot by occupying multiple vertices. Do these results arise more naturally for a grid-based discrete graph(possibly with $\sqrt{2}$ bounds over the general graph case)? What is the relationship of the size of the robot to the solution quality?

6.2 Continuous Disk Setup

Path planning component: Recent results from Kirkpatrick et al have described the optimal geometric motions for a 2-disk planning problem on a plane. Combined with the exhaustive search for the scheduling, this would give the optimum solution.

Do the same results and insights from the discrete graph case apply to the continuous case?

6.3 General Manipulators in a Continuous Setup

The motion planner used for coming up with multi-arm motions is the `dRRT*`[12].

In the general case geometric bounds will not apply to the individual motions because

- *General manipulators might not have identical solutions for the same 2-object transfer problem (for instance., if 2 Kuka arms are on opposite sides of the table);*
- *A direct estimate of a bound might not be possible for manipulator solutions given the object transfer locations.*

Introduce our decomposition of the cost metric: a) first optimize object transfer by the two arms, i.e., solve the assignment problem first and then b) optimize the schedule for the object assignment.

What is the (worst-case) degradation that arises due to this cost decomposition? Can it be bounded? Point to empirical results to show that this is a practical solution

7 Bounding Costs under Disc Manipulator Model in a Continuous Setting

We seek to bound the cost of k-arm manipulation. First, we make some observations (Lemma 1 and Lemma 2) for general manipulator setups, assuming that the workspace is a unit square tabletop and all manipulators have access to the entire workspace (when collision between manipulators is not considered). We assume that each pickup/drop-off of an object incurs cost c_{pd} and moving the end-effector incurs a cost of c_t per unit distance. Note that c_t may be used to capture distance (energy) cost or time cost.

Lemma 1. *For an infinite class of rearrangement problems, a k-arm ($k \geq 2$) solutions can be arbitrarily better than the optimal single arm solution.*

Proof. Consider the case where n objects to be rearranged are divided into two groups of $n/2$ each. Objects in each group are with ε distance from each other and are also ε-close to their goals. Let the distance between the two groups be about 1, i.e., the two groups of objects are at two opposite ends of the a table. Assume further that the initial positions of two end-effectors (in a 2-arm

setting) are just at the edge of the table with each end-effector ε close to a group of $n/2$ objects. The cost incurred by a 2-arm solution is then no more than $nc_{pd} + 2n\varepsilon c_t$. On the other hand, a single arm solution incurs a cost of at least $nc_{pd} + (2n-1)\varepsilon c_t + c_t$. If c_{pd} and ε are sufficiently small, then the 2-arm solution, and in general a k-arm solution constructed similarly, can be arbitrarily better than a single arm solution. □

Lemma 2. *For any rearrangement problem, the best k-arm ($k \geq 2$) solution cannot be worse than an optimal single arm solution.*

Proof. We simply let only one of the arms in a k-arm setup carry out the best single arm solution while letting all other arms remain in place and outside of the unit square workspace. □

Lemma 1 suggests that for certain setups, there may not be a meaningful bound on the performance ratio between a k-arm solution and a single arm solution. This motivates the examination of other setups and we look at an often used setting: randomly chosen start and goal locations for n objects (within a unit square). Here, we seek to derive a meaningful bound on what can be obtained using a 2-arm solution as compared to a single arm solution. First, we derive a conservative cost of a single arm solution.

For a single arm, the (optimal) cost has three main parts: 1. pickup and drop-off of the n objects with a cost of $C_{pd} = nc_{pd}$ (assuming the starts and goals do not have any overlap), 2. the fixed transfer cost from start to goal for each object, C_{sg}, and 3. the optimal transfer cost going from the goals to starts, C_{gs}. The optimal single arm solution cost (e.g., makespan) is then

$$C_{single} = nc_{pd} + C_{sg} + C_{gs}. \tag{10}$$

To estimate (10), we first note that the randomized setup will allow us to obtain the expected C_{sg} as ([29]):

$$\frac{2 + \sqrt{2} + 5\ln(1+\sqrt{2})}{15} nc_t \approx 0.52nc_t. \tag{11}$$

We may approximate C_{gs} by simply computing an optimal assignment of the goals to the starts of the objects excluding the matching of the same start and goal. Denoting the distance cost of this matching as C_{gs}^M, clearly $C_{gs} > C_{gs}^M$ because the paths produced by the matching may form multiple closed loops instead of the desired single loop that connects all starts and goals. However, the number of loops produced by the matching procedure is on the order of $\ln n$ and therefore, $C_{gs} < C_{gs}^M + \ln nc_t$, by [39]. By [1], $C_{gs}^M = \Theta(\sqrt{n \ln n})$. Putting these together, we have

$$\Theta(\sqrt{n \ln n} c_t) = C_{gs}^M < C_{gs} < C_{gs}^M + \ln nc_t = \Theta(\sqrt{n \ln n} c_t) + \ln nc_t, \tag{12}$$

which implies that $C_{gs} \approx C_{gs}^M$ because for large n, $\ln n \ll \sqrt{n \ln n}$. It is estimated in [45] that $C_{gs}^M \approx 0.44\sqrt{n \ln n} c_t$ for large n. We conclude that

$$C_{single} \approx nc_{pd} + (0.52n + 0.44\sqrt{n \ln n})c_t. \tag{13}$$

Noting that the $0.44\sqrt{n \ln n} c_t$ term may also be ignored for large n, we have

$$C_{single} \approx (c_{pd} + 0.52c_t)n. \tag{14}$$

That is, the transfer cost (end-effector moving with objects) dominates the transit cost (without objects). For the 2-arm setting, we make the simplification that each arm's volume is represented as a disc of some radius r. For obtaining a 2-arm solution, we partition the n objects randomly into two piles of $n/2$ objects each; we can then obtain two initial solutions similar to the single arm case. In expectation, these two halves should add up to approximately (14) as well.

From the initial 2-arm solution, we construct an *asynchronous* 2-arm solution that is collision-free. We make the further assumption that pickups and drop-offs can be achieved without collisions between the two arms, which can be achieved with properly designed end-effectors. The main overhead is then the potential collision between the two (disc) arms during transfer and transit operations. Because there are $n/2$ objects for each arm to work with, an end-effector may travel a path formed by $n+1$ straight line segments. Therefore, there are up to $(n+1)^2$ intersections between the two end-effector trajectories where potential collision may happen. However, because for the transits and transfers associated with one pair of objects (one for each arm) can have at most four intersections, there are at most $2n$ potential collisions to handle. For each intersection, we may let one end-effector wait while letting the other one circling around it until the collision is resolved, which incurs a cost that is bounded by $2\pi \cdot r \cdot c_t$. Adding up all the potential extra cost, we have that a 2-arm solution has a cumulative cost of

$$C_{dual} = C_{single} + 2n(2\pi r c_t) \approx (c_{pd} + 0.52c_t + 4\pi r c_t)n. \tag{15}$$

For small r, C_{dual} is almost the same as C_{single} if we consider c_t as a distance (e.g., energy) cost. However, if we consider makespan, a time-based objective, then we have the 2-arm cost as

$$C^t_{dual} \approx (c_{pd} + 0.52c_t)\frac{n}{2} + 4n\pi r c_t \tag{16}$$

and we have the cost ratio as

$$\frac{C^t_{dual}}{C_{single}} \approx \frac{(c_{pd} + 0.52c_t)\frac{n}{2} + 4n\pi r c_t}{(c_{pd} + 0.52c_t)n} = \frac{1}{2} + \frac{4\pi r c_t}{c_{pd} + 0.52c_t}. \tag{17}$$

When r is small or when c_t/c_{pd} is small, we observe that the 2-arm solution uses roughly half of the time needed as the single arm solution. On the other hand, a 2-arm solution cannot do better than saving over $1/2$ of the required task completion time (i.e., makespan) as compared to a single arm solution. The 2-arm argument can be directly extended to k-arm for arbitrary fixed k; we summarize the discussion in the following theorem.

Theorem 1. *For rearranging objects with non-overlapping starts and goals that are uniformly distributed in a unit square, a k-arm solution can have an asymptotic speedup of k over the single arm solution.*

We note that the same can be said for a *synchronous* 2-arm solution if when c_t/c_{pd} is small. However, for synchronous solutions, C^t_{dual} is no longer $C_{single}/2$ for large n due to synchronization.

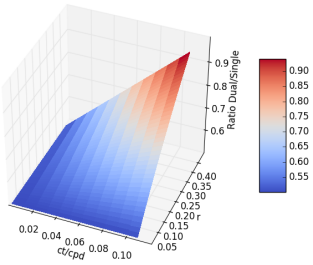

Fig. 3. Bounds

8 Evaluation

Describe experiments: we will design benchmarks per model Describe alternative algorithms a) *single-arm solution per model,* b) *exhaustive search integrated with the corresponding path planner for each model,* c) *approximate solution we propose [unless we come up with something better for the easier models].* d) *1-TSP broken into 2 and coordinated*

What do we measure: a) solution quality in terms of the cost metrics identified and b) computation time.

8.1 Comparison methods

The method is evaluated against competing approaches to demonstrate its efficacy.

- **Exhaustive approach**: This method performs a brute force search under synchronization assumptions.
- **ILP Solution**: Drawing from the advances in multi-vehicle pickup and delivery solutions [7], a comparison can be done against a similar method that solves an `m-TSP` problem first and proceeds to resolve collisions on the schedule.
- **Split-TSP**: This method attempts to perform a `1-TSP` and splits the solutions, with coordination applied to the solutions obtained therein.

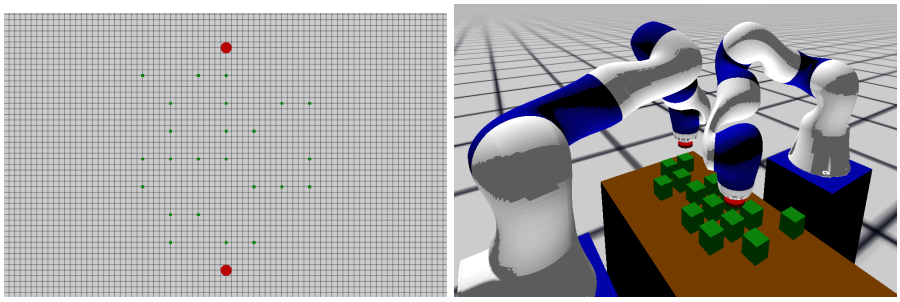

Fig. 4. The benchmark environments

8.2 Continuous Disk

The results are shown in Figure 5.

8.3 General Manipulator

The results are shown in Figure 6.

9 Discussion

Let us assume a ρ-Approx algorithm, A' exists for the problem.

Consider for instance the problem of rearrangement with 3 arms: Assuming a hypothetical manipulator that incurs a constant transit cost on the plane(ie. if the manipulator is not holding an object, the cost for any transit in a bounded plane is constant). The problem of an optimal set of object assignments should reduce to 3D-matching. ρ-Approx A' should return a solution to this problem as well. However, we know that 3DM is hard to approximate. Hence A' does not exist, ie. the rearrangement problem for $k > 2$ arms should be hard to approximate.

References

1. Ajtai, M., Komlós, J., Tusnády, G.: On optimal matchings. Combinatorica **4**(4), 259–264 (1984). DOI 10.1007/BF02579135
2. Aronov, B., de Berg, M., van der Stappen, A.F., Švestka, P., Vleugels, J.: Motion planning for multiple robots. Discrete & Computational Geometry **22**(4), 505–525 (1999)
3. Auletta, V., Monti, A., Parente, M., Persiano, P.: A linear-time algorithm for the feasibility of pebble motion on trees. Algorithmica **23**(3), 223–245 (1999)
4. Ben-Shahar, O., Rivlin, E.: Practical pushing planning for rearrangement tasks. IEEE Transactions on Robotics and Automation **14**(4), 549–565 (1998)
5. Berenson, D., Srinivasa, S., Kuffner, J.: Task space regions: A framework for pose-constrained manipulation planning. The International Journal of Robotics Research **30**(12), 1435–1460 (2011)

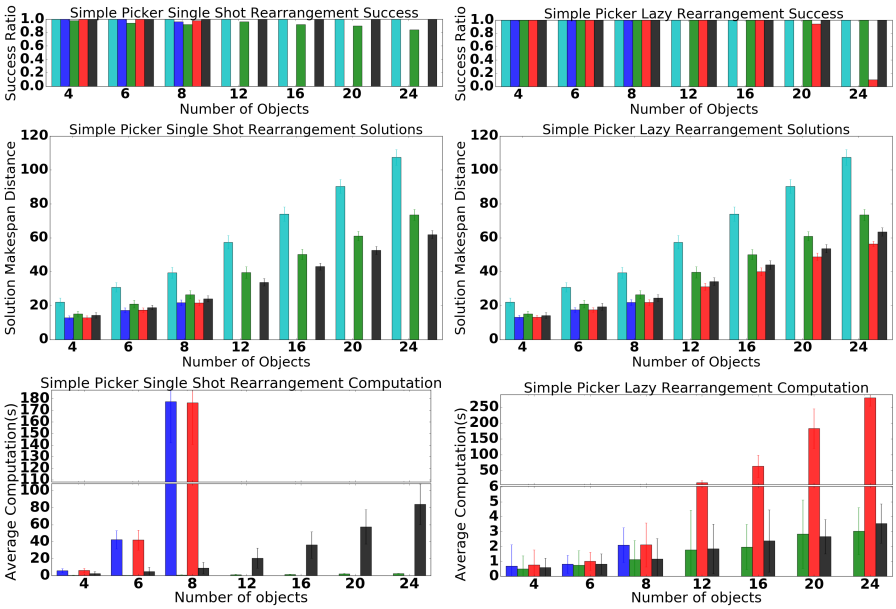

Fig. 5. Results from the experiments with a continuous disk picker averaged over 50 experiments. The *top* figure shows the total computation time and solution times. The computation time includes both motion planning time and combinatorial overheads. The lightly shaded bars represent the computation time. The *bottom* figure shows the same comparison only taking into account the combinatorial overhead (lightly shaded bars) and the solution time(solid colors).

6. Călinescu, G., Dumitrescu, A., Pach, J.: Reconfigurations in graphs and grids. SIAM Journal on Discrete Mathematics **22**(1), 124–138 (2008)
7. Caricato, P., Ghiani, G., Grieco, A., Guerriero, E.: Parallel tabu search for a pickup and delivery problem under track contention. Parallel Computing **29**(5), 631–639 (2003)
8. Chen, P.C., Hwang, Y.K.: Practical path planning among movable obstacles. In: Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on, pp. 444–449. IEEE (1991)
9. Cohen, B., Chitta, S., Likhachev, M.: Single-and dual-arm motion planning with heuristic search. The International Journal of Robotics Research **33**(2), 305–320 (2014)
10. Coltin, B.: Multi-agent pickup and delivery planning with transfers (2014)
11. Demaine, E.D., Demaine, M.L., O'Rourke, J.: Pushpush and push-1 are np-hard in 2d. arXiv preprint cs/0007021 (2000)
12. Dobson, A., Solovey, K., Shome, R., Halperin, D., Bekris, K.E.: Scalable Asymptotically-Optimal Multi-Robot Motion Planning. In: The 1st IEEE International Symposium on Multi-Robot and Multi-Agent Systems (MRS). Los Angeles, USA (2017)
13. Frederickson, G.N., Hecht, M.S., Kim, C.E.: Approximation algorithms for some routing problems. In: Foundations of Computer Science, 1976., 17th Annual Sym-

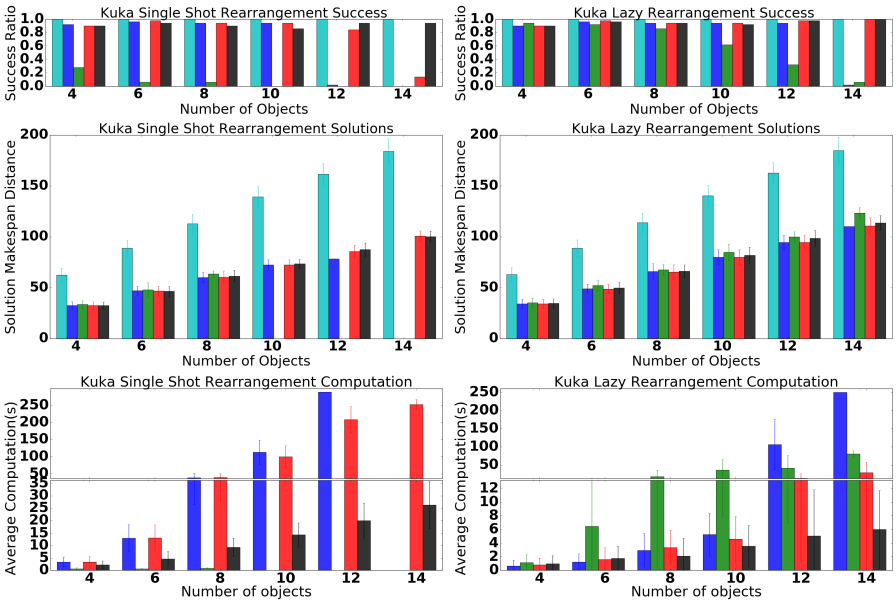

Fig. 6. Results from the experiments with a general Kuka manipulator.

posium on, pp. 216–227. IEEE (1976)
14. Friggstad, Z.: Multiple traveling salesmen in asymmetric metrics. In: Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, pp. 173–188. Springer (2013)
15. Garg, N., Konjevod, G., Ravi, R.: A polylogarithmic approximation algorithm for the group steiner tree problem. Journal of Algorithms **37**(1), 66–84 (2000)
16. Goraly, G., Hassin, R.: Multi-color pebble motion on graphs. Algorithmica **58**(3), 610–636 (2010)
17. Halperin, D., Latombe, J.C., Wilson, R.H.: A general framework for assembly planning: The motion space approach. Algorithmica **26**(3-4), 577–601 (2000)
18. Han, S.D., Stiffler, N.M., Krontiris, A., Bekris, K.E., Yu, J.: Complexity results and fast methods for optimal tabletop rearrangement with overhand grasps. arXiv preprint arXiv:1711.07369 (2017)
19. Havur, G., Ozbilgin, G., Erdem, E., Patoglu, V.: Geometric rearrangement of multiple movable objects on cluttered surfaces: A hybrid reasoning approach. In: Robotics and Automation (ICRA), 2014 IEEE International Conference on, pp. 445–452. IEEE (2014)
20. Kornhauser, D.M., Miller, G.L., Spirakis, P.G.: Coordinating pebble motion on graphs, the diameter of permutation groups, and applications. Master's thesis, M. I. T., Dept. of Electrical Engineering and Computer Science (1984)
21. Krontiris, A., Bekris, K.E.: Dealing with difficult instances of object rearrangement. In: Robotics: Science and Systems (2015)
22. Krontiris, A., Bekris, K.E.: Efficiently solving general rearrangement tasks: A fast extension primitive for an incremental sampling-based planner. In: Robotics and Automation (ICRA), 2016 IEEE International Conference on, pp. 3924–3931. IEEE (2016)

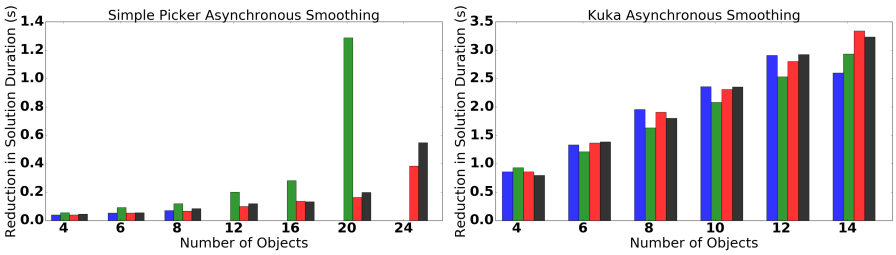

Fig. 7. Improvement in solution duration from asynchronous smoothing of the solutions.

23. Lenstra, J.K., Kan, A.: Complexity of vehicle routing and scheduling problems. Networks **11**(2), 221–227 (1981)
24. Leroy, S., Laumond, J.P., Siméon, T.: Multiple path coordination for mobile robots: A geometric algorithm. In: IJCAI, vol. 99, pp. 1118–1123 (1999)
25. Nieuwenhuisen, D., van der Stappen, A.F., Overmars, M.H.: An effective framework for path planning amidst movable obstacles. In: Algorithmic Foundation of Robotics VII, pp. 87–102. Springer (2008)
26. Ota, J.: Rearrangement of multiple movable objects-integration of global and local planning methodology. In: Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on, vol. 2, pp. 1962–1967. IEEE (2004)
27. Parragh, S.N., Doerner, K.F., Hartl, R.F.: A survey on pickup and delivery models part ii: Transportation between pickup and delivery locations. Journal für Betriebswirtschaft **58**(2), 81–117 (2008)
28. Rathinam, S., Sengupta, R.: Matroid intersection and its application to a multiple depot, multiple tsp (2006)
29. Santaló, L.A.: Integral geometry and geometric probability. Cambridge university press (2004)
30. Savelsbergh, M.W., Sol, M.: The general pickup and delivery problem. Transportation science **29**(1), 17–29 (1995)
31. Sharon, G., Stern, R., Felner, A., Sturtevant, N.R.: Conflict-based search for optimal multi-agent pathfinding. Artificial Intelligence **219**, 40–66 (2015)
32. Siméon, T., Laumond, J.P., Cortés, J., Sahbani, A.: Manipulation planning with probabilistic roadmaps. The International Journal of Robotics Research **23**(7-8), 729–746 (2004)
33. Slavik, P.: Errand scheduling problem (1997)
34. Solovey, K., Halperin, D.: On the hardness of unlabeled multi-robot motion planning. The International Journal of Robotics Research **35**(14), 1750–1759 (2016)
35. Solovey, K., Yu, J., Zamir, O., Halperin, D.: Motion planning for unlabeled discs with optimality guarantees. arXiv preprint arXiv:1504.05218 (2015)
36. Srivastava, S., Fang, E., Riano, L., Chitnis, R., Russell, S., Abbeel, P.: Combined task and motion planning through an extensible planner-independent interface layer. In: Robotics and Automation (ICRA), 2014 IEEE International Conference on, pp. 639–646. IEEE (2014)
37. Stilman, M., Schamburek, J.U., Kuffner, J., Asfour, T.: Manipulation planning among movable obstacles. In: Robotics and Automation, 2007 IEEE International Conference on, pp. 3327–3332. IEEE (2007)

38. Sundaram, S., Remmler, I., Amato, N.M.: Disassembly sequencing using a motion planning approach. In: Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on, vol. 2, pp. 1475–1480. IEEE (2001)
39. Treleaven, K., Pavone, M., Frazzoli, E.: Asymptotically optimal algorithms for one-to-one pickup and delivery problems with applications to transportation systems. IEEE Transactions on Automatic Control **59**(9), 2261–2276 (2013)
40. Van Den Berg, J., Stilman, M., Kuffner, J., Lin, M., Manocha, D.: Path planning among movable obstacles: a probabilistically complete approach. In: Algorithmic Foundation of Robotics VIII, pp. 599–614. Springer (2009)
41. Van Den Berg, J.P., Overmars, M.H.: Prioritized motion planning for multiple robots. In: Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on, pp. 430–435. IEEE (2005)
42. Wagner, G., Kang, M., Choset, H.: Probabilistic path planning for multiple robots with subdimensional expansion. In: Robotics and Automation (ICRA), 2012 IEEE International Conference on, pp. 2886–2892. IEEE (2012)
43. Wilfong, G.: Motion planning in the presence of movable obstacles. Annals of Mathematics and Artificial Intelligence **3**(1), 131–150 (1991)
44. Wilson, R.H., Latombe, J.C.: Geometric reasoning about mechanical assembly. Artificial Intelligence **71**(2), 371–396 (1994)
45. Yu, J., Chung, S.J., Voulgaris, P.G.: Target assignment in robotic networks: Distance optimality guarantees and hierarchical strategies. IEEE Transactions on Automatic Control **60**(2), 327–341 (2015)
46. Yu, J., LaValle, S.M.: Multi-agent path planning and network flow. In: Algorithmic Foundations of Robotics X, pp. 157–173. Springer (2013)
47. Yu, J., LaValle, S.M.: Optimal multirobot path planning on graphs: Complete algorithms and effective heuristics. IEEE Transactions on Robotics **32**(5), 1163–1177 (2016)