Dual-arm Rearrangement

Rahul Shome, Kiril Solovey, Jingjin Yu, Kostas E. Bekris, and Dan Halperin

Rutgers University, NJ, USA and 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 and Contribution

Related efforts:

- multi-robot planning and coordination
- manipulation task planning in general
- rearrangement
- combinatorial literature

2.1 Combinatorial Literature

Our problem can be formulated in a variety of related classes of problem, some of which and some select references are mentioned below. Please 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.

The Traveling Salesman Problem: 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 [5] 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 [10] has formulated the problem of multi-start to multi-goal k-TSP as an optimization. This assumes symmetric edge weights. Some work [6] has been done on dealing with graphs which can handle asymmetric edge weights, ie. directed graphs.

Errand Scheduling: If the object transfer is seen as an errand [12] then a graph constructed with nodes being object-arm assignments and edges implying the order of coordinated execution. There has been prior work [7] that operates over the assignment graph(where pairwise assignments are nodes), an object's group consists of sets of nodes that include an object

Pickup and Delivery: 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 [3] 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 [8, 11] has also explored the complexity of the problem. By analyzing a survey [9] 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 [11] have been made to solve the problem. Prior work [13] 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 [2] 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.

3 Problem Setup and Notation

These are the ingredients of our problem:

- *2 robotic arms that can reach every object position*
- *N objects*
- *Start and goal poses of objects are given and do not overlap*
- *Transfers of objects by the arms do not introduce interaction with the stably resting objects (e.g., overhead grasps).*
- *The arms must synchronolously transfer an object or transit between grasps.*

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.

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 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 rearrangement problem consists of moving between an initial object arrangement, $A_{init} \in \mathcal{A}^f$ to a target object arrangement $A_{goal} \in \mathcal{A}^f$.

The workspace is also shared by manipulators $m = \{m_1, m_2\}$. The current setup consists of 2 manipulators. The reachable workspace of the manipulator $m_k, k = 1, 2$ is denoted by $\mathcal{W}_R^k \subset \mathcal{W} \subset SE(3)$. It is assumed that all valid poses of the objects are reachable by both manipulators. This means,

$$p_i \subset \mathcal{W}_R^k, \forall p_i \in A, \forall A \in \mathcal{A}^f, \forall k$$

. A manipulator is capable of performing Pick-and-Place actions on the objects at the stable poses. We assume that there is no interaction between the manipulator and picked object with any other object in their stable poses.

A collision free configuration space of the manipulator m_k is $\mathbb{C}_{free}^{m_k}$. A configuration $q_k \in \mathbb{C}_{free}^{m_k}$. The manipulator trajectory $\pi : [0, 1] \to \mathbb{C}_{free}^{m_k}$ is a collision free trajectory for the manipulator m_k.

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

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

$\pi_{[m^k, o^i]}$ corresponds to a manipulator trajectory that executes the `soma`. The manipulator m_k attains states $q_{init}^{[m_k, o_i]}$ and $q_{goal}^{[m_k, o_i]}$ to pick and place the object o_i at the initial pose $p_i^{init} \in A_{init}$ and target pose $p_i^{goal} \in A_{goal}$.

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

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

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

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 travelled 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 travelled by an end effector during each synchronized operation).

The manipulator trajectory $\pi_{\texttt{soma}(k,i)} : [0,1] \rightarrow \mathbb{C}^{m_k}_{free}$ performs the object transfer while keeping the manipulator collision free with static obstacles. The trajectory $\pi_{[\texttt{soma}(k,i),\texttt{soma}(k,j)]} : [0,1] \rightarrow \mathbb{C}^{m_k}_{free}$ represents the transit motion of the manipulator between $q^{\texttt{soma}(k,i)}_{goal} \rightarrow q^{\texttt{soma}(k,j)}_{init}$. A cost function $\text{cost}(\pi)$ is defined for a trajectory.

$$\text{cost}(\texttt{soma}(k,i)) = \text{cost}(\pi_{\texttt{soma}(k,i)}) = \|\pi_{\texttt{soma}(k,i)}\|$$

$$\text{cost}([\texttt{soma}(k,i),\texttt{soma}(k,j)])\text{cost}(\pi_{[\texttt{soma}(k,i),\texttt{soma}(k,j)]}) = \|\pi_{[\texttt{soma}(k,i),\texttt{soma}(k,j)]}\|$$

The cost of a $\texttt{scoma} = \{\texttt{coma}_1, \texttt{coma}_2 \ldots \texttt{coma}_l\}$,

$$\text{cost}(\texttt{scoma}) = \text{cost}(\texttt{coma}_l) + \sum_{i=1}^{l-1} (\text{cost}(\texttt{coma}_i) + \text{cost}(\texttt{coma}_i, \texttt{coma}_{i+1}))$$

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.

$$\text{cost}(\texttt{coma}(i,j)) = \max\{\text{cost}(\texttt{soma}(1,i)), \text{cost}(\texttt{soma}(2,j))\}$$

$$\begin{aligned}&\text{cost}(\texttt{coma}(i_1,j_1), \texttt{coma}(i_2,j_2))\\&= \max\{\text{cost}([\texttt{soma}(1,i_1), \texttt{soma}(1,i_2)]), \text{cost}([\texttt{soma}(2,i_1), \texttt{soma}(2,i_2)])\}\end{aligned}$$

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 $c^* = \mathit{cost}(\texttt{scoma}^*)$. The 2-arm rearrangement problem consists of finding $\texttt{scoma}^*$.

3.3 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 $\texttt{coma}$. Each branch of the tree corresponds to the sequence of $\texttt{coma}$. At every level, all combinations of remaining object assignments to manipulators give rise to the branching factor of this tree.

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

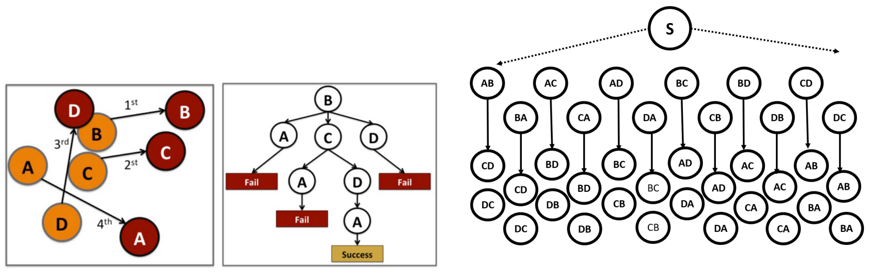

Fig. 1. Search trees in the single and dual arm rearrangement problems for 4 objects.

$n/2$. 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*` [4]) 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.

3.4 ILP Formulation

An ILP formulation of the problem is discussed in this section. The formulation attempts to pose an multi-robot pickup and delivery problem with interaction constraints as an ILP. Choices can be made in terms of assigning costs during the solutions:

– The costs can arise from heuristic measures based solely on object positions

– The costs can be heuristic measures based on single robot optimal paths without interactions
– The costs can correspond to coordinated motion planning solutions for multiple robots.

It is possible that the ILP can be evaluated lazily with more accurate motion planning costs on top of heuristically obtained feasible candidate solutions.

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

```
1  $G_{rearrangement}(V, E)$;
2  AssignHeuristicCosts($E$);
3  setupILPSolver;
4  $converged \leftarrow False$;
5  $\mathcal{E}_{cand} \leftarrow \emptyset$;
6  while not $converged$ do
7      AssignCoordinationCosts($\mathcal{E}_{cand}, G_{rearrangement}$);
8      $\mathcal{E}_{ilp} \leftarrow$ ILPSolver($G_{rearrangement}$);
9      $converged \leftarrow \mathcal{E}_{ilp} == \mathcal{E}_{cand}$;
10     $\mathcal{E}_{cand} \leftarrow \mathcal{E}_{ilp}$;
11 $\Pi^* \leftarrow$ trace($\mathcal{E}_{cand}$);
12 return $\Pi^*$;
```

4 Contribution

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.

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

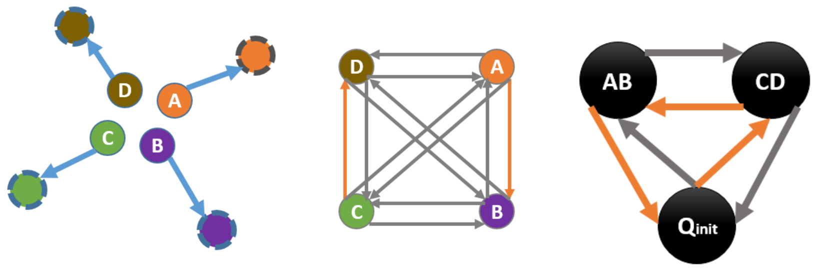

Fig. 2. The image shows an instance of a dual-arm monotone problem where none of the object placements impose constraints. The image in the middle represents a formulation of the problem as minimum weight edge matching on a fully connected directed graph. The image on the right represents the second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

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.

We will study this problem under three different models:

– a completely discrete setup where the objects and the robot occupy nodes of a graph;

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

```
1  $G(V, E)$;
2  $V \leftarrow \mathcal{O}$;
3  for $o_i, o_j \in \mathcal{O}$ s.t. $i \neq j$ do
4      $w_{(i,j)} \leftarrow \infty$;
5  for $o_i, o_j \in \mathcal{O}$ s.t. $i \neq j$ do
6      $w_{(i,j)} \leftarrow \mathrm{cost}(\mathtt{coma}(i, j))$;
7      if $w_{(i,j)} < w_{(j,i)}$ then
8          $E \leftarrow E \cup e(i, j, w_{(i,j)})$;
9  $E^* \leftarrow \mathtt{EdgeMatching}(G)$;
10 $G_{tsp}(V_{tsp}, E_{tsp})$;
11 $V_{tsp} \leftarrow V_{tsp} \cup \mathtt{coma}_e \forall e \in E^*$;
12 $V_{tsp} \leftarrow V_{tsp} \cup \mathtt{coma}_{home}$;
13 for $\mathtt{coma}_u, \mathtt{coma}_v \in V_{tsp}$ s.t. $u \neq v$ do
14     $w_{(u,v)} \leftarrow \mathrm{cost}(\mathtt{coma}_u, \mathtt{coma}_v)$;
15     $E_{tsp} \leftarrow E_{tsp} \cup e(u, v, w_{(u,v)})$;
16 $\mathcal{T} \leftarrow \mathtt{TSP}(G_{tsp})$;
17 $\Pi^* \leftarrow \mathtt{trace}(\mathcal{T})$;
18 return $\Pi^*$;
```

- 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.

4.1 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.

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?

4.2 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?

4.3 General Manipulators in a Continuous Setup

The motion planner used for coming up with multi-arm motions is the `dRRT`*[4]. 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.

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

5 Bounding Costs under Disc Manipulator Model in the Continuous Setting

We seek to bound the cost of k-arm manipulation. First, we make some observations for general manipulator setups, assuming that the workspace is a unit square tabletop and all manipulators have full access to the workspace. 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.

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 (Proof sketch). Consider the case where n objects to be rearranged are divided into two groups of $n/2$ each. Objects within each group are within ε distance from each other and are also ε-close to their goals. The distance between the two groups is $1/2$. Assume further that the initial positions of two end-effectors (in a k-arm setting) are just above the two groups of objects. The cost incurred by a k-arm ($k \geq 2$) solution is then no more than $2n\varepsilon c_t + nc_{pd}$. On the other hand, a single arm solution incurs a cost of at least $nc_{pd} + 1/2$. If c_{pd} and ε are sufficiently small, then the k-arm solution can be arbitrarily better than a single arm solution.

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

Proof (Proof sketch). We simply let only one of the arms in a k-arm setup carry out the best single arm solution.

Lemma 1 suggests that for certain setups, there isn't 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 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 gained using a synchronous 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/drop-off of the n objects with a cost of $C_{pd} = nc_{pd}$ (assuming the no-overlap setup), 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 bounded by

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

The randomized setup will allow us to compute the expected C_{sg}. We can further obtain C_{gs} using [13, 1]. We note that the optimal single arm strategy is a TSP tour connecting the goals to the starts.

For the 2-arm setting, we make the simplification that each arm's volume is represented as a disc of some radius r. For the 2-arm solution, we simply partition the n objects randomly into two equal parts of $n/2$ objects; we can then obtain two initial solutions as we have done for the single arm case. In expectation, these two halves should have the same value, say, C_{2-init}.

From these two initial solutions, we construct a synchronous 2-arm solution. We make the further assumption that the pickup/drop-off 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 transfers (with and without objects). We note that for each pair of objects, there are at most two occasions where the two arms may collide, each incurring some extra cost that is bounded and proportional to r. Let this cost be $c_c(r)$. The two arm solution then has a (makespan) cost of

$$C_{2-init} + nc_c(r).$$

We should then be able to argue that the ratio

$$\frac{C_{2-init} + nc_c(r)}{C_{single}}$$

is not too bad when r is not very large. We can also generate plots illustrating how the ratio change as n and r changes.

6 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.

6.1 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 [2], 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.

6.2 Continuous Disk

The results are shown in Figure 3.

6.3 General Manipulator

The results are shown in Figure 4.

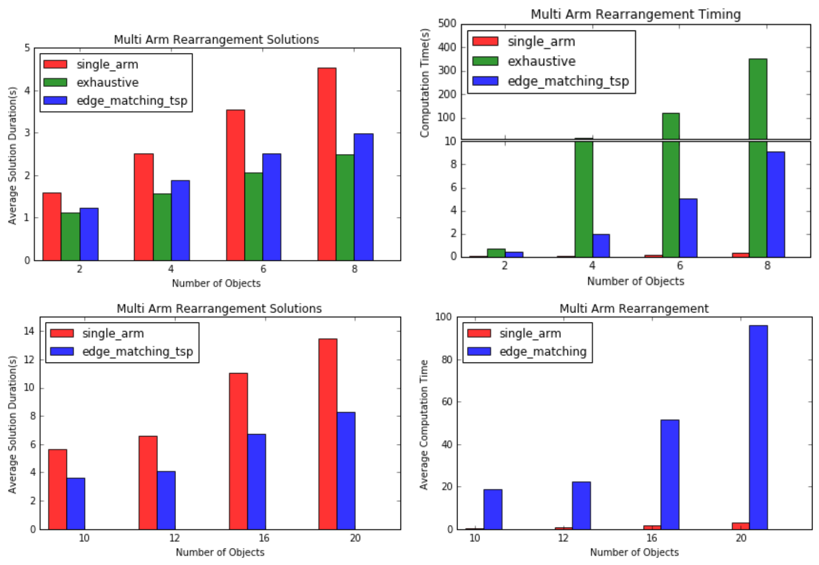

Fig. 3. Results from the experiments with a continuous disk picker.

7 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. 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)
3. Coltin, B.: Multi-agent pickup and delivery planning with transfers (2014)
4. 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)

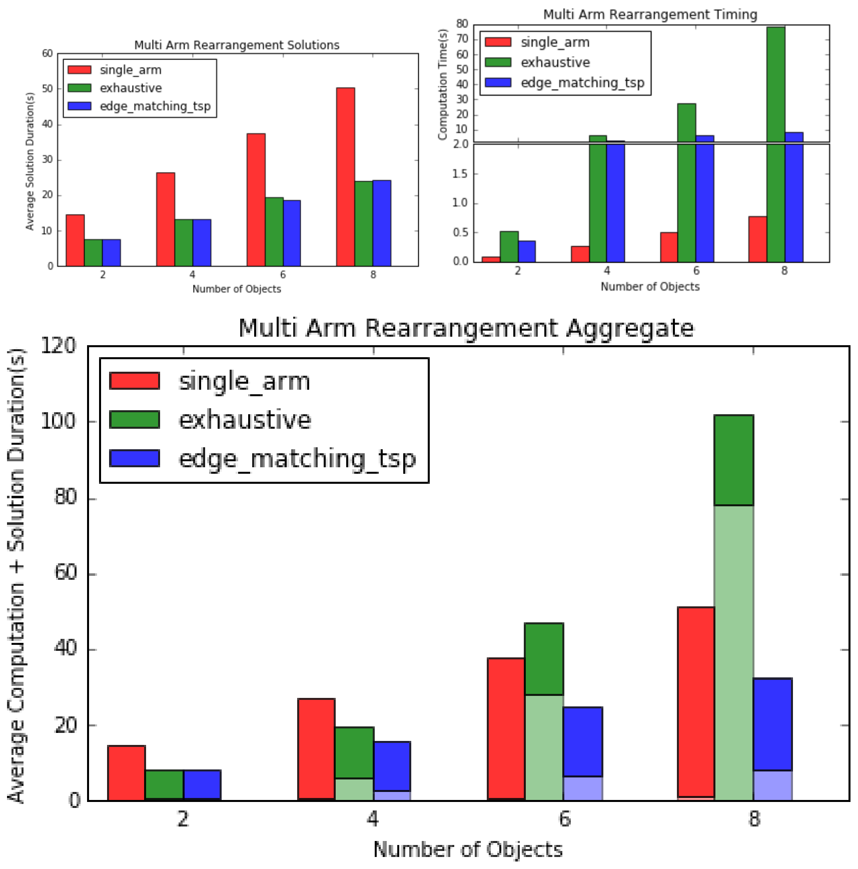

Fig. 4. Results from the experiments with a general Kuka manipulator.

5. Frederickson, G.N., Hecht, M.S., Kim, C.E.: Approximation algorithms for some routing problems. In: Foundations of Computer Science, 1976., 17th Annual Symposium on, pp. 216–227. IEEE (1976)
6. Friggstad, Z.: Multiple traveling salesmen in asymmetric metrics. In: Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, pp. 173–188. Springer (2013)
7. Garg, N., Konjevod, G., Ravi, R.: A polylogarithmic approximation algorithm for the group steiner tree problem. Journal of Algorithms **37**(1), 66–84 (2000)
8. Lenstra, J.K., Kan, A.: Complexity of vehicle routing and scheduling problems. Networks **11**(2), 221–227 (1981)
9. 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)
10. Rathinam, S., Sengupta, R.: Matroid intersection and its application to a multiple depot, multiple tsp (2006)

11. Savelsbergh, M.W., Sol, M.: The general pickup and delivery problem. Transportation science **29**(1), 17–29 (1995)
12. Slavik, P.: Errand scheduling problem (1997)
13. 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)