Fast, High-Quality Dual-Arm Rearrangement in Synchronous, Monotone Tabletop Setups

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. Rearranging objects on a planar surface arises in a variety of robotic applications, such as product packaging. Using two arms can improve efficiency but introduces new computational challenges. This paper studies the structure of dual-arm rearrangement for synchronous, monotone tabletop setups and develops an optimal mixed integer model. It then describes an efficient and scalable algorithm, which first minimizes the cost of object transfers and then of moves between objects. This is motivated by the fact that, asymptotically, object transfers dominate the cost of solutions. Moreover, a lazy strategy minimizes the number of motion planning calls and results in significant speedups. Theoretical arguments support the benefits of using two arms and indicate that synchronous execution, in which the two arms perform together either transfers or moves, introduces only a small overhead. Experiments support these points and show that the scalable method can quickly compute solutions close to the optimal for the considered setup.

1 Introduction

Automation tasks in industrial and service robotics, such as product packing or sorting, often require sets of objects to be arranged in specific poses on a planar surface. Efficient and high-quality single-arm solutions have been proposed for such setups [18]. The proliferation of robot arms, however, including dual-arm setups, implies that industrial settings can utilize multiple robots in the same workspace (Fig 1). This work explores a) the benefits of coordinated dual-arm rearrangement versus single-arm, b) the combinatorial challenges involved and c) computationally efficient, high-quality and scalable methods.

A motivating point is that the coordinated use of multiple arms can result in significant improvements in efficiency. This arises from the following argument.

Lemma 1. *There are classes of tabletop rearrangement problems, where a k-arm ($k \geq 2$) solution can be arbitrarily better than the optimal single-arm one.*

For instance, assume two arms that have full (overhand) access to a unit square planar tabletop. There are n objects on the table, divided into two groups of $\frac{n}{2}$ each. Objects in each group are ε-close to each other and to their goals. Let the distance between the two groups be on the order of 1, i.e., the two groups are

Work by *RS*, *JU* and *KB* was supported by NSF CNS-1330789, IIS-1617744 and IIS-1734419, and *DH* by the ISF grant 825/15, Blavatnik Computer Science Research Fund, Blavatnik Interdisciplinary Cyber Research Center, Yandex and Facebook.

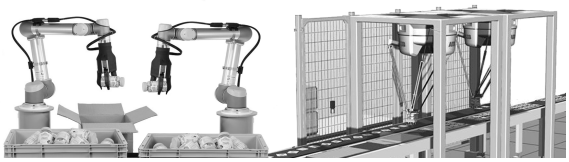

Fig. 1. Example of dual-arm setups that can utilize algorithms proposed in this work.

at opposite ends of the table. The initial position of each end-effector is ε-close to a group of objects. Let the cost of each pick and drop action be c_{pd}, while moving the end-effector costs c_t per unit distance. Then, the 2-arm cost is no more than $2nc_{pd} + 2n\varepsilon c_t$. A single arm solution costs at least $2nc_{pd} + (2n-1)\varepsilon c_t + c_t$. If c_{pd} and ε are sufficiently small, the 2-arm cost can be arbitrarily better than the single-arm one. The argument also extends to k-arms relative to $(k-1)$-arms.

In most practical setups, the expectation is that a 2-arm solution will be close to half the cost (e.g., time-wise) of the single-arm case, which is a desirable improvement. While there is coordination overhead, the best 2-arm solution cannot do worse; simply let one of the arms carry out the best single arm solution while the other remains outside the workspace. More generally:

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

The above points motivate the development of scalable algorithmic tools for such dual-arm rearrangement instances. This work considers certain relaxations to achieve this objective. In particular, monotone tabletop instances are considered, where the start and goal object poses do not overlap. Furthermore, the focus is on synchronized execution of pick-and-place actions by the two arms. i.e., where the two arms simultaneously transfer (different) objects, or simultaneously move towards picking the next objects. Theoretical arguments and experimental evaluation indicate that this does not significantly degrade solution quality.

The first contribution is the study of the combinatorial structure of synchronous, monotone dual-arm rearrangement. Then, a mixed integer linear programming (`MILP`) model is proposed that achieves optimal coordinated solutions in this domain. The proposed efficient algorithm `Tour_Over_Matching` (`TOM`) significantly improves in scalability. `TOM` first optimizes the cost of object transfers and assigns objects to the two arms by solving an optimal matching problem. It minimizes the cost of moves from an object's goal to another object's start pose per arm as a secondary objective by employing a `TSP` solution. Most of the computation time is spent on the many calls to a lower-level motion planner that coordinates the two arms. A lazy evaluation strategy is proposed, where motion plans are evaluated for candidate solution sequences. This results in significant computational improvement and reduces the calls to the motion planner. An analysis studies the expected improvement in solution quality versus the single arm case, as well as the expected cost overhead from a synchronous solution.

Finally, experiments for i) a simple planar picker setting, and ii) two 7-DOF Kuka arms, demonstrate a nearly two-fold improvement against the single arm case for the proposed approach in practice. The algorithm exhibits close to optimal solutions and good scalability. The lazy evaluation strategy significantly improves computation costs for both the optimal and proposed methods.

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. It is closely related to multi-robot planning and coordination where a challenge is the high dimensionality of the configuration space. Optimal strategies were developed for simpler instances of the problem [35], although in general the problem is known to be computationally hard [33]. Decentralized approaches [40] also used velocity tuning [24]. General multi-robot planning tries to plan for multiple high-dimensional platforms [42, 14] using sampling-based techniques. Recent advances provide scalable [34] and asymptotically optimal [9] sampling based frameworks.

In some cases, by restricting the input of the problem to a certain type, it is possible to cast known hard instances of a problem as related algorithmic problems which have efficient solvers. For instance, unlabeled multi-robot motion planning can be reduced to pebble motion on graphs [1]; pebble motion can be reduced to network flow [46]; and single-arm object rearrangement can be cast as a traveling salesman problem [18]. These provide the inspiration to closely inspect the structure of the problem to derive efficient solutions.

In this work we leverage a connection between dual-arm rearrangement and two combinatorial problems: (1) optimal matching [10] and (2) TSP. On the surface the problem seems closely related to multi-agent TSP. Prior work has formulated the k-TSP solution in terms of splitting a single tour [11] or as an optimization task [27]. Some work [12] deals with asymmetric edge weights which are more relevant to the problems of our interest. The problem can be posed as an instance of multi vehicle pickup and delivery (PDP) [26]. Prior work [7] has applied the PDP problem to robots, taking into account time windows and robot-robot transfers. Some seminal work [23, 29] has explored its complexity, and concedes to the hardness of the problem, while others studied cost bounds [38], and proposed ILP formulations [29]. Typically this line of work ignores coordination costs, though some methods [5] reason about it on candidate solutions.

Navigation among movable objects deals with the combinatorial challenges of multiple objects [43, 39] and has been shown to be a hard problem, and extended to manipulation applications [37]. Despite a lot of interesting work on challenges of manipulation and grasp planning, the current work shall make assumptions that avoid complexities arising from them. Manipulators opened the applications of rearrangement planning [3, 25], including instances where objects can be grasped only once or monotone [37], as well as non-monotone instances [19, 36]. Efficient solutions to assembly planning problems [44, 17] typically assumes monotonicity, as without it the problem becomes much more difficult. Recent work has dealt with the hard instances of task planning [4, 6] and rearrangement planning [21, 22, 18]. Sampling-based task planning has made a recent push towards guarantees of optimality [41, 30]. These are broader approaches that are invariant to the combinatorial structure of the application domain. The current work draws inspiration from these varied lines of research.

General task planning methods are unaware of the underlying structure studied in this work. Single-arm rearrangement solutions will also not be effective

in this setting. The current work tries to bridge this gap and provide insights regarding the structure of dual-arm rearrangement. Under assumptions that enable this study, an efficient solution emerges for this problem.

3 Problem Setup and Notation

Consider a planar surface and a set of n rigid objects $\mathcal{O} = \{o_1, o_2 \cdots o_n\}$ that can rest on the surface in stable poses $p_i \in \mathcal{P}_i \subset SE(3)$. The arrangement space $\mathcal{A} = \mathcal{P}_1 \times \mathcal{P}_2 \ldots \times \mathcal{P}_n$ is the Cartesian product of all $\mathcal{P}_i$, where $A = (p_1, \ldots, p_n) \in \mathcal{A}$. In valid arrangements $\mathcal{A}_{\text{val}} \subset \mathcal{A}$, the objects are pairwise disjoint.

Two robot arms m^1 and m^2 can pick and place the objects from and to the surface. $\mathbb{C}^k_{\text{free}}$ is the set of collision-free configurations for arm m^k (while ignoring possible collisions with objects or the other arm), and is assumed here to be a subset of $\mathbb{R}^d$. A path for m^k is denoted as $\pi^k : [0,1] \to \mathbb{C}^k_{\text{free}}$ and includes picking and placing actions. Let $\mathbb{C}^i_{\text{free}}(\pi^j)$ represent the collision free $\mathbb{C}$-space for m^i, given that m^j moves along π^j. This is a function of the paths' parametrization. The space of dual-arm paths $\mathcal{D}$ denotes pairs of paths for the two arms: $D = (\pi^1, \pi^2) \in \mathcal{D}$. Then, $A(A_{\text{init}}, D)$ is the resulting arrangement when the objects are at A_{init} and moved according to D. Let $\text{cost}(D) : \mathcal{D} \to \mathbb{R}$ be a cost function over dual-arm paths.

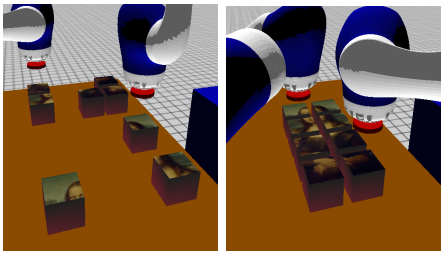

Fig. 2. An example of object rearrangement involving two robotic arms. Initial (left) and final (right) object configuration.

Definition 1 (Optimal Dual-arm Rearrangement). *Given arms m^1, m^2 and objects $\mathcal{O}$ to be moved from initial arrangement $A_{\text{init}} \in \mathcal{A}_{\text{val}}$ to target arrangement $A_{\text{goal}} \in \mathcal{A}_{\text{val}}$, the optimal dual-arm rearrangement problem asks for a dual-arm path $D^* \in \mathcal{D}$, which satisfies the following constraint:*

$$D^* = (\pi^{*1}, \pi^{*2}) \mid A(A_{\text{init}}, D^*) = A_{\text{goal}} \text{ and } \pi^{*i} \in \mathbb{C}^i_{\text{free}}(\pi^{*j}) \tag{1}$$

and optimizes a cost function: $D^* = \underset{\forall D \in \mathcal{D}}{\texttt{argmin}}\ \text{cost}(D)$.

A set of assumptions are introduced to deal with the problem's combinatorics. The reachable task-space $\mathcal{T}^k \subset SE(3)$ of arm m^k is the set of $SE(3)$ poses that objects attached to the arm's end effector can acquire.

Let the ordered set of objects moved during the arm path π^i be denoted as $\bar{\mathcal{O}}(\pi^k)$. In general, an object can appear many times in $\bar{\mathcal{O}}(\pi^k)$. The current work, however, focuses on monotone instances, where each object is moved once.

Assumption 1 (Monotonicity) *There are dual-arm paths $D = (\pi^1, \pi^2)$ that satisfy Eq. 1, where each object $o_i \in \mathcal{O}$ appears once in $\bar{\mathcal{O}}(\pi^1)$ or $\bar{\mathcal{O}}(\pi^2)$.*

For the problem to be solvable, all objects are reachable by at least one arm at both A_{init} and A_{goal}: $\forall\ p_i \in A_{\text{init}}$ and $\forall\ p_j \in A_{\text{goal}}$ and $\exists\ k \in [1,2] : p_i, p_j \subset \mathcal{T}^k$. The focus will be on simultaneous execution of transfer and move paths.

Transfers: Dual-arm paths $T(\pi^1_i, \pi^2_i) \in \mathcal{D}$, where $\bar{\mathcal{O}}(\pi^k_i) = o^k_i$ and each m^k:

- starts the path in contact with an object o^k_i at its initial pose in A_{init},
- and completes it in contact with object o^k_i at its final pose in A_{goal}.

Moves or Transits: Paths $M(\pi^1_{i\to i'}, \pi^2_{i\to i'}) \in \mathcal{D}$, $\bar{\mathcal{O}}(\pi^1_{i\to i'}) = \emptyset$, and each m^k:
- starts in contact with object o^k_i at its final pose in A_{goal},
- and completes it in contact with object $o^k_{i'}$ at its initial pose in A_{init}.

Assumption 2 (Synchronicity) *Consider dual-arm paths, which can be decomposed into a sequence of simultaneous transfers and moves for both arms:*

$$D = \left(T(\pi^1_1, \pi^2_1), M(\pi^1_{1\to 2}, \pi^2_{1\to 2}), \ldots, M(\pi^1_{\ell-1\to\ell}, \pi^2_{\ell-1\to\ell}), T(\pi^1_\ell, \pi^2_\ell)\right). \quad (2)$$

For simplicity, Eq. 2 does not include an initial move from $q^k_{\text{safe}} \in \mathbb{C}^k_{\text{free}}$ and a final move back to q^k_{safe}. An odd number of objects can also be easily handled. Then, the sequence of object pairs moved during a dual-arm path as in Eq. 2 is:

$$\Omega(D) = \left(\omega_i = (o^1_i, o^2_i) \mid i, j \in [1 \cdots \ell], \bigcup_i (o^1_i \cup o^2_i) = \mathcal{O}, \forall k, k' \in [1, 2], o^k_i \neq o^{k'}_j\right).$$

Given the pairs of objects ω_i, it is possible to express a transfer as $T(\omega_i)$ and a move as $M(\omega_{i\to j})$. Then, $D(\Omega)$ is the synchronous, monotone dual-arm path due to $\Omega = (\omega_1, \ldots, \omega_\ell)$, i.e., $D(\Omega) = (T(\omega_1), M(\omega_{1\to 2}), \ldots, M(\omega_{\ell-1\to\ell}), T(\omega_\ell))$.

Assumption 3 (Object Non-Interactivity) *There are collision-free transfers $T(\omega_i)$ and moves $M(\omega_{i\to i'})$ regardless of the object poses in A_{init} and A_{goal}.* This entails that there is no interaction between the n resting objects and the arms during transits. Similarly, there are no interactions between the arm-object system and the $n-2$ resting objects during the transfers. Collisions involving the arms, static obstacles and picked objects are always considered.

The metric this work focuses on relates to makespan and minimizes the sum of the longest distances traveled by the arms in each synchronized operation. Let $\|\pi^k\|$ denote the Euclidean arc length in $\mathbb{C}^k_{\text{free}} \subset \mathbb{R}^d$ of path π^k. Then, for transfers $\text{cost}(T(\pi^1_i, \pi^2_i)) = \max\{\|\pi^1_i\|, \|\pi^2_i\|\}$. Similarly, $\text{cost}(M(\pi^1_{i\to i'}, \pi^2_{i\to i'})) = \max\{\|\pi^1_{i\to i'}\|, \|\pi^2_{i\to i'}\|\}$. Then, over the entire dual-arm path D define:

$$\text{cost}(D) = \sum_{i=1}^{\ell} \text{cost}(T(\omega_i)) + \sum_{i=1}^{\ell-1} \text{cost}(M(\omega_{i\to i+1})). \quad (3)$$

Note that the *transfer* costs do not depend on the order with which the objects are moved but only on the assignment of objects to arms. The *transit* costs arise out of the specific ordering in $\Omega(D)$. Then, for the setup of Definition 1 and under Assumptions 1-3, the problem is to compute the optimal sequence of object pairs Ω^* so that $D(\Omega^*)$ satisfies Eq. 1 and minimizes the cost of Eq. 3.

Note on Assumptions: This work restricts the study to a class of monotone problems that relate to a range of industrial packing and stowing applications. The monotonicity assumption is often used in manageable variants of well-studied problems [44, 17]. A monotone solution also implies that every object's start and target is reachable by at least one arm. To deal with cases where the number of reachable objects is unbalanced between the two arms, a `NO_ACT` task assignment is introduced and considered in Section 6.

The synchronicity assumption allows to study the combinatorial challenges of the problem, which do not relate to the choice of time synchronization of different picks and placements. Section 6 describes the use of `dRRT*`[9] as the underlying motion planner that can discover solutions that can synchronize arm

motions for simultaneous picks, and simultaneous placements. The synchronicity assumption is relaxed through smoothing (Section 6).

The non-interactivity assumption comes up naturally in planar tabletop scenarios with top-down picks or delta robots. Such scenarios are popular in industrial settings. Once the object is raised from the resting surface, transporting it to its target does not introduce interactions with the other resting objects. This assumption is also relaxed in Section 6, with a lazy variant of the proposed method. Once a complete candidate solution is obtained, collision checking can be done with everything in the scene.

Overall, under the assumptions, the current work identifies a problem structure, which allows arguments pertaining to the search space, completeness, and optimality. Nevertheless, the smoothed, lazy variant of the proposed method will still return effective solutions in practice, even if these assumptions do not hold.

4 Baseline Approaches and Size of Search Space

This section highlights two optimal strategies to discover $D(\Omega^*)$: a) exhaustive search, which reveals the search space of all possible sequences of object pairs Ω and b) an MILP model. Both alternatives, however, suffer from scalability issues as for each possible assignment, it is necessary to solve a coordinated motion planning problem for the arms. This motivates minimizing the number of assignments ω considered, and the number of motion planning queries it requires for discovering $D(\Omega^*)$, while still aiming for high quality solutions.

Exhaustive Search: The exhaustive search approach shown in Fig. 3 is a brute force expansion of all possible sequences of object pairs Ω. Nodes correspond to transfers $T(\omega_i)$ and edges are moves $M(\omega_{i\to i'})$. The approach evaluates the cost for all possible branches to return the best sequence Ω. The total number of nodes is in the order of $O(n!)$, expressed over the levels L of the search tree as:

$$ {}^nP_2 + {}^nP_2 \times {}^{n-2}P_2 + ... + {}^nP_2 \times {}^{n-2}P_2 \times {}^{n-4}P_2 ... \times {}^2P_2 = \sum_{L=1}^{n/2} \prod_{k=0}^{2L-1} (n-k), $$

where ${}^nP_k$ is the k-permutations of n.

Motion plans can be reused, however, and repeated occurrences of $T(\omega_i)$ and $M(\omega_{i\to i'})$ should be counted only once for a total of:

- ${}^nP_2$ *transfers* of objects, and
- ${}^nP_2 \times {}^{n-2}P_2$ *transits* between all possible valid ordered pairs of ω.

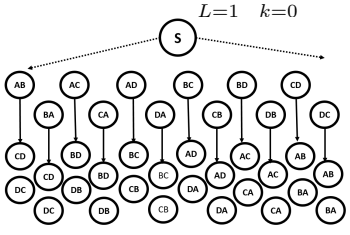

Fig. 3. Search tree for 4 objects.

Additional motion plans are needed for the initial move from q^k_{safe} and the return to it at the end of the process, introducing $2 \times {}^nP_2$ transits:

$$ \#\text{ of Transfers} + \#\text{ of Moves} = {}^nP_2 + \Big(({}^nP_2 \times {}^{n-2}P_2) + (2 \times {}^nP_2) \Big) \qquad (4) $$

This returns optimal synchronized solution but performs an exhaustive search and requires exponentially many calls to a motion planner.

MILP Formulation: Mixed Integer Linear Programming (`MILP`) formulations can utilize highly optimized solvers [16]. Prior work has applied these techniques for solving m-TSP [27, 12] and pickup-and-delivery problems [7, 29], but viewed

these problems in a decoupled manner. This work outlines an MILP formulation for the synchronized dual-arm rearrangement problem that reasons about coordination costs arising from arm interactions in a shared workspace.

Graph Representation: The problem can be represented as a directed graph where each vertex $v = \omega_v = (o_v^1, o_v^2)$ corresponds to a transfer $T(\omega_u)$ and edges $e(u, v)$ are valid moves $M(\omega_{u\to v})$. A valid edge $e(u, v)$ is one where an object does not appear more than once in the transfers of nodes u and v. The cost of a directed edge $e(u, v)$ encodes both the cost of the transfer $T(\omega_v)$ and the cost of the move $M(\omega_{u\to v})$. There is also a vertex S, which connects moves from and to the safe arm configurations q_{safe}^k. The directed graph $\hat{G}(\hat{V}, \hat{E})$ is defined:

$$\hat{V} = \{v = \omega_v = (o_v^1, o_v^2) \mid \forall\ o_v^1, o_v^2 \in \mathcal{O}, o_v^1 \neq o_v^2\} \cup \{S\}$$
$$\hat{E} = \{e(u, v) \mid \forall\ u, v \in \hat{V} \text{ so that } u \neq v,\ o_v^k \neq o_u^\ell\ \forall\ k, \ell \in [1, 2]\}$$
$$\cup\ \{e(S, v)\ \forall\ v \in \hat{V} \setminus S\} \cup \{e(v, S)\ \forall\ v \in \hat{V} \setminus S\}$$
$$cost_{e(u,v)} = \text{cost}(u) + \text{cost}(u, v) = \text{cost}(T(\omega_u)) + \text{cost}(M(\omega_{u\to v}))$$

Let $\text{cost}(S) = 0$. The total number of motion planning queries needed to be answered to define the edge costs is expressed in Eq. 4. The formulation proposed in this section tries to ensure the discovery of Ω^* on $\hat{G}$ as a tour that starts and ends at S, while traversing each vertex corresponding to Ω^*. To provide the MILP formulation, define $\delta_{\text{in}}(v)$ as the in-edge set v, and $\delta_{\text{out}}(v)$ as the out-edge set. Then, $\gamma(o)$ is the object coverage set $\gamma(o) = \{e(u, v) \mid e \in \hat{E}, o \in \omega_u\}$, i.e., all the edges that transfer o.

Model: Set the optimization objective as: $\min \sum_{e\in\hat{E}} cost_e x_e$ [A]
Eq. [B] below defines indicator variables. Eqs. [C-E] ensure edge-flow conserved tours. Eqs. [F-G] force S to be part of the tour. Eq. [H] transfers every object only once. Eq. [I] lazily enforces the tour to be of length $\frac{n}{2} + 1$. While the number of motion-planning queries to be solved is the same as in exhaustive search, efficient MILP solvers [16] provide a more scalable search process.

$$x_e \in \{0, 1\}\ \ \forall e \in \hat{E} \quad \text{[B]} \qquad \sum_{e\in\delta_{\text{in}}(S)} x_e = 1 \quad \text{[F]}$$
$$\sum_{e\in\delta_{\text{in}}(v)} x_e \leq 1\ \ \forall v \in \hat{V} \quad \text{[C]} \qquad \sum_{e\in\delta_{\text{out}}(S)} x_e = 1 \quad \text{[G]}$$
$$\sum_{e\in\delta_{\text{out}}(v)} x_e \leq 1\ \ \forall v \in \hat{V} \quad \text{[D]} \qquad \sum_{e\in\gamma(o)} x_e = 1\ \ \forall o \in \mathcal{O} \quad \text{[H]}$$
$$\sum_{e\in\delta_{\text{in}}(v)} x_e = \sum_{e\in\delta_{\text{out}}(v)} x_e\ \ \forall v \in \hat{V} \quad \text{[E]} \qquad \sum_{e(u,v)\in\mathfrak{T}} x_e < |\mathfrak{T}|\ \ \forall \mathfrak{T} \subset \hat{V}, |\mathfrak{T}| \leq \frac{n}{2} \quad \text{[I]}$$

5 Efficient Solution via Tour over Matching

The optimal baseline methods described above highlight the problem's complexity. Both methods suffer from the large number of motion-planning queries they have to perform to compute the cost measures on the corresponding search structures. For this purpose it needs to be seen whether it is possible to decompose the problem into solvable sub-problems.

Importance of Transfers: In order to draw some insight, consider again the tabletop setup with a general cost measure of c_t per unit distance. Lemma 1 suggests that under certain conditions, 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 another often used setting—randomly chosen non-overlapping start and goal locations for n objects (within a unit square). In order to derive a meaningful bound on the benefit of using a 2-arm solution to a single-arm solution, we first derive a conservative cost of a single-arm solution. A single-arm optimal cost has three main parts: 1) the portion of the transfer cost involving the pickup and drop-off of the n objects with a cost of $C_{pd} = nc_{pd}$, 2) the remaining transfer cost from start to goal for all objects C_{sg}, and 3) the transit cost going from the goals to starts C_{gs}. The single arm cost is

$$C_{\text{single}} = nc_{pd} + C_{sg} + C_{gs}. \tag{5}$$

To estimate Eq. 5, first note that the randomized setup will allow us to obtain the expected C_{sg}[28] as $\frac{2+\sqrt{2}+5\ln(1+\sqrt{2})}{15} nc_t \approx 0.52nc_t$. 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, $C_{gs} < C_{gs}^M + c_t \ln n$, by [38]. By [2], $C_{gs}^M = \Theta(\sqrt{n \ln n})$. Putting these together, we have,

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

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. Therefore,

$$C_{\text{single}} \approx nc_{pd} + (0.52n + 0.44\sqrt{n \ln n})c_t \approx (c_{pd} + 0.52c_t)n \tag{6}$$

noting that the $0.44\sqrt{n \ln n} c_t$ term may also be ignored for large n. The cost of dual arm solutions will be analyzed in Section 7.

Lemma 3. *For large n, the transfers dominate the cost of the solution.*

We formulate a strategy to reflect this importance of transfers. The proposed approach gives up on the optimality of the complete problem, instead focusing on a high-quality solution, which:

- first optimizes transfers and selects an assignment of object pairs to arms,
- and then considers move costs and optimizes the schedule of assignments.

This ends up scaling better by effectively reducing the size of the search space and performing fewer motion planning queries. It does so by optimizing a related cost objective and taking advantage of efficient polynomial-time algorithms.

$$\begin{aligned} \mathcal{G}(\mathcal{V}, \mathcal{E}) \\ \mathcal{V} = \{v = o, \forall o \in \mathcal{O}\} \\ \mathcal{E} = \{e(u,v) : \omega = (u,v), \forall u, v \in \mathcal{V}\} \\ \text{cost}(e(u,v)) = \text{cost}(T(\omega = (u,v))) \end{aligned} \tag{7}$$

$$\begin{aligned} \mathcal{G}_\Gamma(\mathcal{V}_\Gamma, \mathcal{E}_\Gamma) \\ \mathcal{V}_\Gamma = \{v = \omega, \forall \omega \in \{\Omega\}^*\} \cup \{\omega_S\} \\ \mathcal{E}_\Gamma = \{e(u,v) : \omega_{u\to v}, \forall u, v \in \mathcal{V}_\Gamma\} \\ \text{cost}(e(u,v)) = \text{cost}(M(\omega_{u\to v})) \end{aligned} \tag{8}$$

Foundations: Consider a complete weighted directed graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ (Eq. 7), where $v \in \mathcal{V}$ corresponds to a single object o. Each directed edge, $e = (o_i, o_j) \in \mathcal{E}$ is an ordered pair of objects o_i and o_j, where the order determines the assignment of an object to an arm m^1 or m^2. The cost of an edge cost(e) is the coordinated motion planning cost of performing the transfer corresponding to $\omega = (o_i, o_j)$. For instance, as shown in Fig 4(middle), $e(A, B)$ corresponds to m^1 transferring 'A', while m^2 transfers 'B', and the cost($e(A, B)$) = cost($T(\omega = (A, B))$ is the cost of such a concurrent motion. It should also be noted that since the arms are different, in general, cost($e(A, B)$) $\neq$ cost($e(B, A)$).

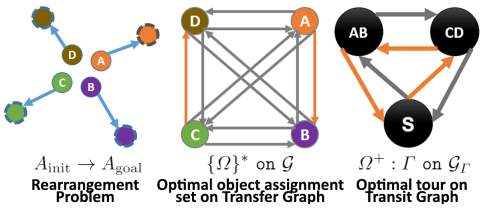

Fig. 4. (left) A dual-arm rearrangement problem. (middle) The same problem as minimum weight edge matching on a fully connected directed graph of *transfers*. (right) The ordering of the object-arm assignments from an optimal tour over a *transit* graph.

Following the observation made in Eq. 3, the cost of the transfers can be reasoned about independent of their order. Define $\{\Omega\}$ as the unordered set of $\omega \in \Omega$, then unordered transfer cost component corresponds to $\sum_{\omega \in \{\Omega\}} \text{cost}(T(\omega))$. Since $\mathcal{G}$ is a complete graph where every edge corresponds to every possible valid ω, by construction $\{\Omega\} \subset \mathcal{E}$. A candidate solution of a monotone dual-arm rearrangement problem must transfer every object exactly once. In terms of the graph, this means that in the subset of edges $\{\Omega\}$ every vertex appears in only one edge. We arrive at the following crucial observation.

Lemma 4 (Perfect Matching). *Every candidate solution to a monotone dual-arm rearrangement problem comprises of a set of unordered object-to-arm assignments $\{\Omega\}$ that is also a perfect matching solution on $\mathcal{G}$.*

As per the decomposition of the costs in Eq. 3, it follows that the cost($\{\Omega\}$) is the cost of the transfers in the solution. The solution to the minimum-weight perfect edge matching problem on such a graph would correspond to a $\{\Omega\}$ that optimizes the cost of *transfers* of all the objects.

Lemma 5 (Optimal Matching). *The set of object-to-arm assignments $\{\Omega\}^*$ that minimizes the cost of object transfers is a solution to the minimum-weight perfect matching problem on $\mathcal{G}$.*

Once such an optimal assignment $\{\Omega\}^*$ is obtained, the missing part of the complete solution is the set of transits between the object transfers and their ordering. Construct another directed graph $\mathcal{G}_\Gamma$ (Eqs. 8), where the vertices comprise of the $\omega \in \{\Omega\}^*$. An edge $e(\omega_{u \rightarrow v})$ corresponds to the coordinated *transit* motion between them. For instance, as in Fig 4(right) for an edge between $\omega(A, B)$ and $\omega(C, D)$, m^1 moves from the target pose of object 'A' to the starting pose of object 'C', and similarly m^2 moves from the target of 'B' to the start of 'D'. An additional vertex corresponding to the starting(and ending) configuration of the two arms (S) is added to the graph. The graph is fully connected again to represent all possible transits or moves. A complete candidate solution to the problem now requires the sequence of ω, which is a complete tour over $\mathcal{G}_\Gamma$, that visits all the vertices i.e., an ordered sequence of vertices $\Gamma = (S, \Omega_\Gamma, S)$.

Lemma 6 (Tour). *Any complete tour Γ over the graph $\mathcal{G}_\Gamma$, corresponds to a sequence of object-to-arm assignments Ω_Γ and is a candidate solution to the synchronous dual-arm rearrangement problem.*

Let $\mathbb{P}^{\{\Omega\}}$ represent the set of all possible ordering of the elements in $\{\Omega\}$. This means, any candidate tour corresponds to a $\Omega_\Gamma \in \mathbb{P}^{\{\Omega\}}$. An optimal tour on $\mathcal{G}_\Gamma$ minimizes the transit costs over the all possible candidate solutions in $\mathbb{P}^{\{\Omega\}^*}$.

$$\Omega^+ = \underset{\Omega \in \mathbb{P}^{\{\Omega\}^*}}{\texttt{argmin}} \sum_{e(u,v)\in\Gamma} \text{cost}(M(\omega_{u\to v}))) \tag{9}$$

This differs from the true optimal Ω^*, since the second step of finding the optimal transit tour only operates over all possible solutions that include the optimally matched transfers obtained in the first step. The insight here is that, even though Ω^+ reports a solution to a hierarchical optimization objective, the search space is much smaller, and the sub-problems more efficient to solve.

Algorithm: This section describes the algorithm `Tour_Over_Matching` (`TOM`) outlined in the previous section. The steps correspond to Algo 1.
`transfer_graph`: This function constructs a directed graph $\mathcal{G}$ defined by Eqs. 7. This step creates a graph with n vertices and ${}^nP_2$ edges.
`optimal_matching`: This function takes the graph $\mathcal{G}$ constructed as an argument and returns the unordered set of edges, corresponding to the set of optimal transfers over $\mathcal{G}$. Optimal matching over an undirected graph can be solved using Edmond's Blossom Algorithm [10, 13, 8]. The directed graph $\mathcal{G}$ is converted into an equivalent undirected graph $\mathcal{G}_U$. Since $\mathcal{G}$ is complete, every pair of vertices shares two directed edges. $\mathcal{G}_U$ only preserves the minimally weighted connection for every vertex pair. The result of matching is a subset of edges on $\mathcal{G}_U$ which correspond to a set of directed edges on $\mathcal{G}$ i.e., $\{\Omega\}^*$. The runtime complexity of the step is $O(|\mathcal{E}||\mathcal{V}|\log|\mathcal{V}|) = O({}^nP_2 n\log n) = O(n^3\log n)$.

Algorithm 1: `TOM`$(\mathcal{O}, S, A_{\text{init}}, A_{\text{goal}})$

```
G ← transfer_graph(O, A_init, A_goal);
{Ω}* ← optimal_matching(G);
G_Γ ← transit_graph({Ω}*, S);
Ω+ ← optimal_tour(G_Γ);
return D(Ω+);
```

`transit_graph`: This function constructs the directed transit graph $\mathcal{G}_\Gamma$ as per the set of Eqs. 8. This constructs $\frac{n}{2}+1$ vertices and ${}^{(\frac{n}{2}+1)}P_2$ edges.
`optimal_tour`: Standard TSP solvers like Gurobi [16] can be used to find the optimal tour over $\mathcal{G}_\Gamma$ corresponding to Ω^+.

The costs are deduced from coordinated motion plans over edges. The total number of such calls compared to the count from the baseline in Equ. 4, shows a saving in the order of $O(n^2)$ queries (# of Transfers + # of Moves).

$$\frac{\text{Baseline \#}}{\texttt{TOM \#}} = \frac{{}^nP_2 + \left({}^nP_2 \times {}^{n-2}P_2\right) + \left(2 \times {}^nP_2\right)}{{}^nP_2 + {}^{(\frac{n}{2}+1)}P_2} = \frac{4(n-1)((n-5)n+9)}{5n-2}$$

The evaluation performed here focuses on the maximum of distances (Eq. 3) for a fair comparison with the other methods. The prioritization of optimization objective, however, is also amenable to other cost functions, where carrying objects is often more expensive than object-free motions.

6 Integration with Motion Planning

The algorithms described so far are agnostic to the underlying motion planner. Depending upon the model of the application domain, different motion planning primitives might be appropriate. For planar environments with disk robot pickers (similar to delta robots), recent work [20] characterizes the optimal two-disk coordinated motions. The current implementation uses a general multi-robot motion planning framework dRRT* [9] for dual-arm coordinated planning.

In practice the cost of generating and evaluating two-arm motions can dominate the overall running time of the algorithm, when compared to the combinatorial ingredients that discover the high-level plan, i.e., execution order and and arm assignment. Even though TOM reduces this, further improvements can be made with lazy evaluation.

Lazy Evaluation: Recent work [31] introduces heuristics for dRRT*, which preprocess estimates of the shortest path costs for the arms. Dual-arm rearrangement can be significantly sped up if the motion planning queries are replaced with look-ups of such heuristics. Once a candidate Ω is obtained, motion planning can evaluate the solution $D(\Omega)$. If this fails, the algorithm tries other sequences.

Algorithm 2: Lazy_Evaluation(ALGO, $\mathcal{H}$, MP)

```
1  E_b ← ∅; D ← ∅;
2  while D = ∅ ∧ time_not_exceeded do
3  |  Ω ← ALGO(H, E_b);
4  |  for ω_i, ω_{i→i+1} ∈ Ω do
5  |  |  D_i, D_{i→i+1} ← MP(ω_i), MP(ω_{i→i+1});
6  |  |  D ← (D, D_i, D_{i→i+1});
7  |  |  if D_i = ∅ then
8  |  |  |  E_b ← E_b ∪ {ω_i}; D ← ∅;
9  |  |  if D_{i→i+1} = ∅ then
10 |  |  |  E_b ← E_b ∪ {ω_{i→i+1}}; D ← ∅;
11 |  |  if D = ∅ then break ;
12 return D
```

The algorithm Algo 2 takes as input the algorithm ALGO, a heuristic $\mathcal{H}$, and a motion planner MP. $\mathcal{E}_\mathtt{b}$ keeps track of the blocked edges. The process keeps generating candidate solutions using the ALGO (Line 3). Line 5 motion plans over the candidate solution, and appends to the result (Line 6). Any failures are recorded in $\mathcal{E}_\mathtt{b}$ (Lines 8,10), and inform subsequent runs of ALGO.

Completeness: The lazy variants give up on optimality for the sake of efficiency but given enough retries the algorithms will eventually solve every problem that ALGO can. Since the motion planning happens in the order of execution, the object non-interactivity assumption is relaxed.

Smoothing: Applying velocity tuning over the solution trajectories for the individual arms relaxes the synchronization assumption by minimizing any waits that might be a by-product of the synchronization. Smoothing does not change the maximum of distances, only improves execution time. Indications that the smoothed variants of the synchronous solutions do not provide significant savings in execution time are included in the Appendix [32] for the interested reader.

It should be noted that in an iterative version of TOM, in order to explore variations in $\{\Omega\}^*$ if failures occur in finding Γ, some edges need to be temporarily blocked on $\{\Omega\}^*$. The search structures can also be augmented with NO_ACT tasks, for possible ω where one of the arms do not move.

7 Bounding Costs under Planar Disc Manipulator Model

Following from Lemma 3, the current analysis studies the dual arm costs in the randomized unit tabletop setting, with c_t as the cost measure per unit distance. For the 2-arm setting, assume for simplicity 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 $\frac{n}{2}$ objects each; then obtain two initial solutions similar to the single arm case. It is expected (Eq. 6) that these two halves should add up to approximately $(c_{pd} + 0.52c_t)n$.

From the initial 2-arm solution, we construct an *asynchronous* 2-arm solution that is collision-free. Assume 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 move operations. Because there are $\frac{n}{2}$ objects for each arm to work with, an arm 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 transfers and transits 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, let one arm wait while letting the other circling around it, which incurs a cost that is bounded by $2\pi \cdot r \cdot c_t$.

Adding up all the potential extra cost, a cumulative cost is obtained as

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

For small r, C_{dual} is almost the same as C_{single} c_t is a distance (e.g., energy) cost. Upon considering the maximum of the two arc lengths or makespan (Eq. 3), the 2-arm cost becomes $C^t_{\text{dual}} \approx (c_{pd} + 0.52c_t)\frac{n}{2} + 4n\pi r c_t$. The cost ratio is

$$\frac{C^t_{\text{dual}}}{C_{\text{single}}} \approx \frac{(c_{pd} + 0.52c_t)\frac{n}{2} + 4n\pi r c_t}{(c_{pd} + 0.52c_t)n} = \frac{1}{2} + \frac{4\pi r c_t}{c_{pd} + 0.52c_t} \ . \tag{10}$$

When r is small or when $\frac{c_t}{c_{pd}}$ is small, the 2-arm solution is roughly half as costly as the single arm solution. On the other hand, in this model a 2-arm solution does not do better than $\frac{1}{2}$ of the single arm solution. This argument can be extended to k-arms as well [32].

Theorem 1. *For rearranging objects with non-overlapping starts and goals that are uniformly distributed in a unit square, a 2-arm solution can have an asymptotic improvement of $\frac{1}{2}$ over the single arm solution.*

The synchronization assumption changes the expected cost of the solution. The random partitioning of the n objects into two sets of $\frac{n}{2}$ object with a random ordering of the objects yields $\frac{n}{2}$ pairs of objects transfers, which dominate the total cost for large n. The cost (Eq. 3) of $\frac{n}{2}$ synchronized transfers (ω_i) includes $\frac{n}{2}c_{pd}$ and $C^{\text{sync}}_{sg} \approx (\mathtt{E}(max(l_1, l_2))c_t)\frac{n}{2}$, where $\mathtt{E}(max(l_1, l_2))$ is the expected measure of the max of lengths l_1,l_2 of two randomly paired transfers. Using the *pdf*[15] of lengths of random lines in an unit square and integrating over the setup[32], results in the value of $\mathtt{E}(max(l_1, l_2))$ to be 0.66 . This means $C^{\text{sync}}_{\text{dual}} \approx (c_{pd} + 0.66c_t)\frac{n}{2} + 4n\pi r c_t$. The synchronized cost ratio is

$$\frac{C^{\text{sync}}_{\text{dual}}}{C_{\text{single}}} \approx \frac{(c_{pd} + 0.66c_t)\frac{n}{2} + 4n\pi r c_t}{(c_{pd} + 0.52c_t)n} = \frac{1}{2} + \frac{(0.07 + 4\pi r)c_t}{c_{pd} + 0.52c_t}\,. \tag{11}$$

When $\frac{c_t}{c_{pd}}$ is small, even the synchronized 2-arm solution provides an improvement of $\frac{1}{2}$. For the case when both r and c_{pd} are small, we observe that the ratio approaches 0.636.

Theorem 2. *For rearranging objects with non-overlapping starts and goals that are uniformly distributed in a unit square, a randomized 2-arm synchronized solution can have an asymptotic improvement of $\frac{1}{2}$ over the single arm solution if $\frac{c_t}{c_{pd}}$ is small, and a improvement ≈ 0.64 when both c_{pd} and r are small.*

Note on bounds: Even though the proposed simplified model does not apply immediately costs and collision volumes in general configuration spaces, experiments indicate that the speedups exist in these spaces as well.

8 Evaluation

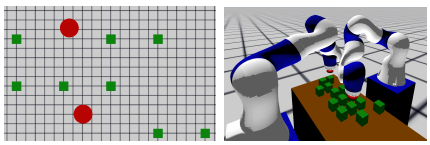

Fig. 5. *Picker* and *Manipulator* trials.

This section describes the experiments performed to evaluate the algorithms in two domains shown in Fig 5: a) simple picker and b) general manipulators. In order to ensure monotonicity, the object starts and goals do not overlap. Uniform cuboidal objects simplify the grasping problem, though this is not a limitation of the methods. 50 random experiments were limited to $300s$ of computation time. The underlying `dRRT`* motion planner is restricted to a max of $3s$ per plan. A comparison point includes a random split method, which splits $\mathcal{O}$ at random into two subsets and chooses an arbitrary ordering. Maximum of distances cost is compared to the single arm solution [18]. Computation times and success rates are reported. The trends in both experiments show that in the single-shot versions, exhaustive and `MILP` tend to time-out for larger n. Lazy variants scale much better for all the algorithms, and in some cases increase the success ratio due to retries. `TOM` has much better running time than exhaustive and `MILP`, and producing better and more solutions than random split. Overall, the results show a) our `MILP` succeeds more within the time limit than exhaustive, b) `TOM` scales the best among all the methods, and c) the cost of solutions from `TOM` is close to the optimal baseline, which is around half of the single arm cost.

Simple Picker: This benchmark evaluates two disk robots hovering over a planar surface scattered with objects. The robots are only free to move around in a plane parallel to the resting plane of the objects, and the robots can pick up objects when they are directly above them. Fig 6(*top*) all runs up to 24 objects succeeded for `TOM`. `MILP` scales better than exhaustive. Lazy random split succeeds in all cases (*bottom*). In terms of solution costs (*middle*) exhaustive finds the true optimal. `MILP` matches exhaustive and `TOM` is competitive. In all experiments, `TOM` enjoys a success rate of 100%.

General Dexterous Manipulator: The second benchmark sets up two `Kuka` arms across a table with objects on it. The objects are placed in the common reachable part of both arms' workspace, and only one top-down grasping configuration is allowed for each object pose. Here (Fig 7) a larger number of

motion plans tend to fail, so the single shot variants show artifacts of the randomness of dRRT*in their success rates. Random split performs the worst since it is unlikely to chance upon valid motion plans. Single shot exhaustive and MILP scale poorly because of expensive motion planning. Interestingly, motion planning infeasibility reduces the size of the exhaustive search tree. The solution costs *(middle)* substantiate benefits of the use of two arms. The computation times *(bottom)* again show the scalability of TOM, even compared to random split.

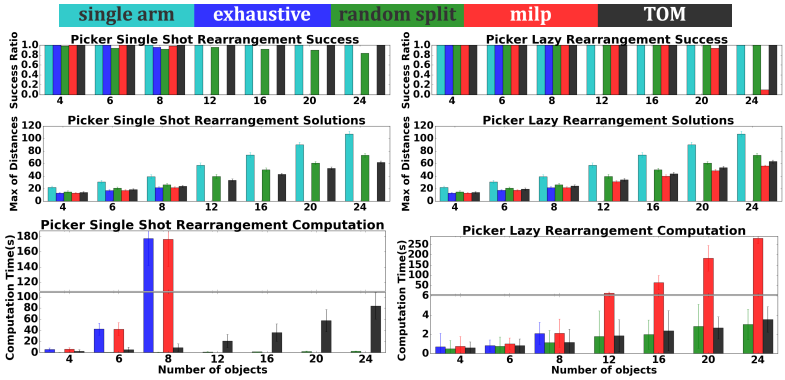

Fig. 6. *Simple Picker* results with success*(top)*, solution costs*(middle)*, and computation*(bottom)* reported for single-shot*(left)* and lazy*(right)* versions of the methods

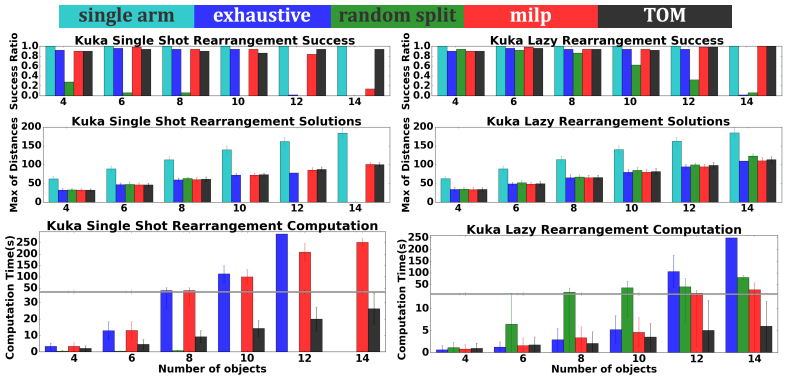

Fig. 7. Kuka results with success*(top)*, solution costs*(middle)*, and computation*(bottom)* reported for single-shot*(left)* and lazy*(right)* versions of the methods

9 Discussion

The current work demonstrates the underlying structure of synchronized dual-arm rearrangement and proposes an MILP formulation, as well as a scalable algorithm TOM that provides fast, high quality solutions. Existing efficient solvers for reductions of the dual-arm problem made TOM effective. Future work will attempt to explore the $k-$arm case and instances of non-monotone rearrangement. The incorporation of manipulation and regrasp reasoning can extend these methods to more cluttered domains.

References

1. Adler, A., de Berg, M., Halperin, D., Solovey, K.: Efficient multi-robot motion planning for unlabeled discs in simple polygons. IEEE T-ASE **12**(4) (2015)
2. Ajtai, M., Komlós, J., Tusnády, G.: On optimal matchings. Combinatorica **4**(4) (1984). DOI 10.1007/BF02579135
3. Ben-Shahar, O., Rivlin, E.: Practical pushing planning for rearrangement tasks. IEEE Transactions on Robotics and Automation **14**(4) (1998)
4. Berenson, D., Srinivasa, S., Kuffner, J.: Task space regions: A framework for pose-constrained manipulation planning. IJRR **30**(12) (2011)
5. Caricato, P., Ghiani, G., Grieco, A., Guerriero, E.: Parallel tabu search for a pickup and delivery problem under track contention. Parallel Computing **29**(5) (2003)
6. Cohen, B., Chitta, S., Likhachev, M.: Single-and dual-arm motion planning with heuristic search. IJRR **33**(2) (2014)
7. Coltin, B.: Multi-agent pickup and delivery planning with transfers. Ph.D. thesis, Carnegie Mellon University, CMU-RI-TR-14-05 (2014)
8. Dezső, B., Jüttner, A., Kovács, P.: Lemon–an open source c++ graph template library. Electronic Notes in Theoretical Computer Science **264**(5) (2011)
9. Dobson, A., Solovey, K., Shome, R., Halperin, D., Bekris, K.E.: Scalable Asymptotically-Optimal Multi-Robot Motion Planning. In: MRS (2017)
10. Edmonds, J.: Maximum matching and a polyhedron with 0, 1-vertices. Journal of research of the National Bureau of Standards B **69**(125-130) (1965)
11. Frederickson, G.N., Hecht, M.S., Kim, C.E.: Approximation algorithms for some routing problems. In: FOCS (1976)
12. Friggstad, Z.: Multiple traveling salesmen in asymmetric metrics. In: Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques. Springer (2013)
13. Galil, Z., Micali, S., Gabow, H.: An O(EV log V) algorithm for finding a maximal weighted matching in general graphs. SIAM Journal on Computing **15**(1) (1986)
14. Gharbi, M., Cortés, J., Siméon, T.: Roadmap Composition for Multi-Arm Systems Path Planning. In: IROS (2009)
15. Ghosh, B.: Random distances within a rectangle and between two rectangles. Bulletin Calcutta Math Soc. **43** (1951)
16. Gurobi Optimization, I.: Gurobi optimizer reference manual (2016)
17. Halperin, D., Latombe, J.C., Wilson, R.H.: A General Framework for Assembly Planning: the Motion Space Approach. Algorithmica **26**(3-4) (2000)
18. Han, S.D., Stiffler, N., Krontiris, A., Bekris, K., Yu, J.: Complexity results and fast methods for optimal tabletop rearrangement with overhand grasps. IJRR (2018)
19. Havur, G., Ozbilgin, G., Erdem, E., Patoglu, V.: Geometric rearrangement of multiple movable objects on cluttered surfaces. In: ICRA (2014)
20. Kirkpatrick, D., Liu, P.: Characterizing minimum-length coordinated motions for two discs. arXiv preprint arXiv:1607.04005 (2016)
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: ICRA (2016)
23. Lenstra, J.K., Kan, A.: Complexity of vehicle routing and scheduling problems. Networks **11**(2) (1981)
24. Leroy, S., Laumond, J.P., Siméon, T.: Multiple path coordination for mobile robots: A geometric algorithm. In: IJCAI, vol. 99 (1999)

25. Ota, J.: Rearrangement of multiple movable objects-integration of global and local planning methodology. In: ICRA, vol. 2 (2004)
26. 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) (2008)
27. Rathinam, S., Sengupta, R.: Matroid intersection and its application to a multiple depot, multiple tsp (2006)
28. Santaló, L.A.: Integral geometry and geometric probability. Cambridge University Press (2004)
29. Savelsbergh, M.W., Sol, M.: The general pickup and delivery problem. Transportation science **29**(1) (1995)
30. Schmitt, P.S., Neubauer, W., Feiten, W., Wurm, K.M., Wichert, G.V., Burgard, W.: Optimal, sampling-based manipulation planning. In: ICRA. IEEE (2017)
31. Shome, R., Bekris, K.E.: Improving the scalability of asymptotically optimal motion planning for humanoid dual-arm manipulators. In: Humanoids (2017)
32. Shome, R., Solovey, K., Yu, J., Bekris, K., Halperin, D.: Fast, high-quality dual-arm rearrangement in synchronous, monotone tabletop setups. arXiv:1810.12202 [cs.RO] (2018)
33. Solovey, K., Halperin, D.: On the hardness of unlabeled multi-robot motion planning. IJRR **35**(14) (2016)
34. Solovey, K., Salzman, O., Halperin, D.: Finding a needle in an exponential haystack: Discrete RRT for exploration of implicit roadmaps in multi-robot motion planning. IJRR **35**(5) (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: ICRA (2014)
37. Stilman, M., Schamburek, J.U., Kuffner, J., Asfour, T.: Manipulation planning among movable obstacles. In: ICRA (2007)
38. 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) (2013)
39. Van Den Berg, J., Stilman, M., Kuffner, J., Lin, M., Manocha, D.: Path planning among movable obstacles: a probabilistically complete approach. In: WAFR (2008)
40. Van Den Berg, J.P., Overmars, M.H.: Prioritized motion planning for multiple robots. In: IROS (2005)
41. Vega-Brown, W., Roy, N.: Asymptotically optimal planning under piecewise-analytic constraints. In: WAFR (2016)
42. Wagner, G., Kang, M., Choset, H.: Probabilistic path planning for multiple robots with subdimensional expansion. In: ICRA (2012)
43. Wilfong, G.: Motion planning in the presence of movable obstacles. Annals of Mathematics and Artificial Intelligence **3**(1) (1991)
44. Wilson, R.H., Latombe, J.C.: Geometric Reasoning about Mechanical Assembly. Artificial Intelligence Journal **71**(2) (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) (2015)
46. Yu, J., LaValle, S.M.: Optimal multirobot path planning on graphs: Complete algorithms and effective heuristics. IEEE Transactions on Robotics **32**(5) (2016)