# Bound to help: cooperative manipulation of objects via compliant, unactuated tails

Young-Ho Kim and Dylan A. Shell

Dept. of Computer Science and Engineering, Texas A&M University

January 30, 2018

**Abstract**

We examine the problem of moving multiple objects to goal locations by a coordinated team of mobile robots. Each robot is equipped with an unactuated, compliant chain attached as an appendage that we call its tail. Each of our robots tow objects by wrapping their tail around an item, securing it by hooking the end of the tail back onto itself, and then dragging. In addition to towing individually, any two robots wishing to operate within a tightly-knit sub-team are able to link the ends of their respective tails. These conjoined pairs can skim a region of space, clustering multiple objects together to transport several at once.

Using operators that model both forms of towing, we formulate the planning problem for collecting multiple objects and transporting them to goal locations. We propose a general framework using logical formulas to express complex tasks. This planning problem is NP-hard and so we settle for either an exhaustive enumeration or a sub-optimal plan. The combinatorics of the action choices make the former prohibitive with as few as eight robots and objects, so we explore heuristics that give satisfactory solutions in reasonable time. We analyze the performance of the proposed algorithm to give an understanding of where it expands fewer search nodes than exact search. The results include data from physical robots executing plans produced by our planner with both individuals and coupled pairs towing objects.

*'those friends thou hast, and their adoption tried*
*grapple them unto thy soul with hoops of steel. . . '*
— Shakespeare's Hamlet

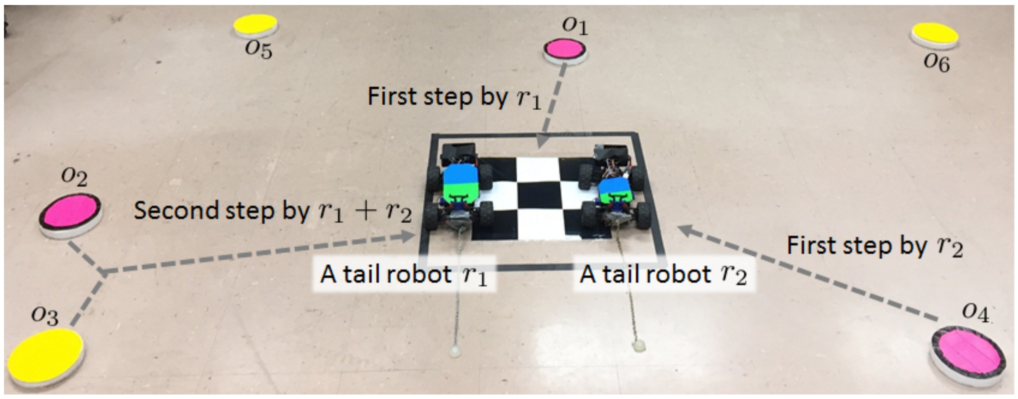
## 1 Introduction

Manipulation problems that involve transferring multiple objects to given goal locations arise in diverse applications. Familiar examples include collecting toys strewn across the floor, removing debris from a pond surface, or, more ordinarily, handling materials in a warehouse. When one can move multiple objects simultaneously, these problems have inherent parallelism which is ripe for exploitation. This paper studies robots equipped with an unactuated flexible structure attached as a tail and examines how these robots can solve multi-object collection tasks as a team.

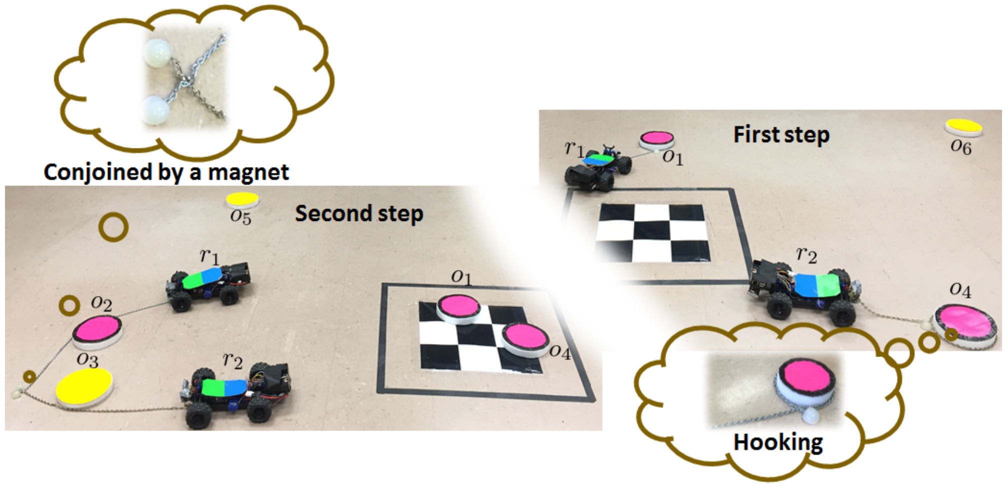
People use strings, cords, ropes, and chains in myriad of ways every day—they are used to bind and secure, they are used to connect and pull, grip and whip. Flexible linear structures are extremely versatile. At present, robots use strings and ropes only rarely since flexible structures remain difficult to model. Yet the advantages for manipulation are apparent: when wrapped to encircle a payload, ropes and strings can restrain objects through the interplay of tension and friction, and, best of all, little need be known about the geometry of the objects being manipulated once the cord is constricted.

The preceding observations have motivated research on distributed robotic manipulation [Rus et al., 1995, Kube and Bonabeau, 2000, Fink et al., 2007, 2008, Wilson et al., 2014], and also manipulation with ropes, going back to the (often overlooked) paper of Yamashita et al. [1998] and also that of Donald et al. [2000]. The extensive body of research dealing with tethers and robots connected by ropes to their environment [Sinden, 2000, Hert and Lumelsky, 1994, Igarashi and Stilman, 2010, Kim et al., 2014, Teshnizi and Shell, 2016, McGarey et al., 2016], knotting [Saha and Isto, 2006, Augugliaro et al., 2015, Wang and Balkcom, 2016] and wrapping [Hill et al., 2015] are also related, and so is work on robotic tails [Chang-Siu et al., 2011, Briggs et al., 2012, Rone and Ben-Tzvi, 2014, Libby et al., 2016]. Recent contributions to this area, such as those of Cheng et al. [2008], Fink et al. [2011], Bhattacharya et al. [2015, 2011], have demonstrated multiple robots using ropes to move objects. The existing multi-robot work has either a rope affixing robots to the objects they manipulate (exemplified by Cheng et al. [2008]), or has a pair of robots connected by a rope for skimming on the surface [Bhattacharya et al., 2015] or descending a cliff [Huntsberger et al., 2003] coordinately. In all the prior work, the rope is attached by the experimenter beforehand and remains permanently attached throughout.

The present paper first extends a robot capable of manipulating single objects with its passive tail [Kim and Shell, 2017] to coordinated manipulation problems for multi-robot systems. We assume that all our mobile robots have an inelastic tail. The ends of their tails can be linked and unlinked each other. Then, this paper examines the setting where the robots begin unattached and are able to attach or detach autonomously. The robots must determine whether it is most effective to work as individuals, or to pair off to form a sub-team with another robot, or some mixture, possibly switching from one form to the other. In this way, we unify both of the preceding approaches. Further, we examine a class of general goal predicates, allowing for much richer task specifications



(a) Two robots with flexible tails construct a plan to transport four objects: The first step is for the robots to move $o_1$ and $o_4$ to the goal. Acting individually and independently, these actions are carried out concurrently. The second step involves the pair of robots linking tails and, together, bringing $o_2$ and $o_3$ to the goal.



(b) To perform the first step, each robot tows an item on its own, hooking its tail around the object and releasing it at the goal. In the second step, the robots execute a primitive that physically couples the pair, they then surround and drag $o_2$ and $o_3$ to the goal cooperatively.

Figure 1: Consider the problem of moving four objects to the chequered region, with at least two being pink. To minimize cumulative distance, the robots balance working as a tightly-knit pair versus operating separately. Pairs have the advantage of greater towing capacity, while individuals may fetch distant objects concurrently.

than related manipulation work.

Figure 1 provides an illustrative instance of the type of manipulation problem we tackle. The top image (Figure 1(a)) shows the initial state of the environment and a plan that the robots can execute to solve the problem. In this case, execution involves two steps. The robots begin by manipulating objects individually by securing their tails around the objects (Figure 1(b), right). Once the objects have been delivered to the goal region and released, the robots connect their tails and cooperate in dragging multiple objects to the goal (Figure 1(b), left). More generally, when working as a conjoined pair, the robots will have greater towing capacity; on the other hand, two robots operating separately have the advantage of being able to simultaneously collect far-flung objects.

Picking what we believe to be an appropriate level of abstraction, we formulate the planning problem for Multi-Object Collection via Cooperative Towing (MOCCT) as a discrete optimal path planning problem, whose solution is a sequence of paths for all robots that minimizes the total manipulation cost. First, we show that the even small MOCCT problem instances result in huge search spaces—searching the space of all possible motion combinations for multiple robots and objects is prohibitive. Consequently, in this paper we are compelled to introduce heuristic simplifications. Our experimental results suggest that, despite needing only a tiny fraction of the computational time, our heuristic algorithm yields reasonable solutions. Secondly, we identify an example where small perturbations to the initial positioning of objects produces drastic changes to the optimal plan, so that the strategy employed must undergo qualitative modifications to retain optimality. We examine the performance of our approach on this example. Thirdly, we show that it is possible for the proposed algorithm to deal with scenarios far more general than those that our particular robots in physical experiments have indicated. We show how to include mixed classes of objects, complex goal specifications and constraints.

This paper contributes the following:

- We study a coordinated towing system where: (1) all robots can be separated or conjoined, (2) these operations are chosen automatically by the algorithm, and (3) may change during execution, i.e. are dynamic—sometimes the robots work as individuals, sometimes they operate within tightly-knit sub-teams.
- We present the first physical demonstration of multiple robots solving manipulation problems in this way.
- We formalize the planning problem in a framework that uses generic predicates to describe object properties, enabling specifications of complex coordination tasks. We believe this to be the first of such generality employed in multi-robot manipulation settings.
- We propose a heuristic algorithm which simplifies the search space and show that the efficiency gains are substantial for the problem instances even of moderate size. Specifically, the opportunistic algorithm is able to save between one and two orders of magnitude running-time over standard A$^\star$ search (with 8 objects, 2 robots, and single goal) with negligible loss in solution quality.

In the sections that follow, after discussing the relationship between our work and the broader literature (in Section 2), we formulate the MOCCT problem in Section 3 and show that it is NP-hard in Section 4. Section 5 presents two algorithms to solve MOCCT problems, and Section 6 first describes the evaluation of the algorithms, then physical robot experiments, and finally illustrates the generality of the formulation via some complex goal specifications. Section 7 provides additional discussion, collecting several important practical considerations and lessons learned from our experience with physical hardware, including some limitations and issues encountered. The last sections conclude and highlight some future work.

## 2 Related Work

As the previous section has motivated the problem, this section focuses on relating our approach to the technical literature. The problem we consider involves the dynamic formation of sub-teams of pairs which move objects together, but when this facet of the problem is removed, the result can be formulated as a multi-vehicle routing problem [Dantzig and Ramser, 1959, Lenstra and Kan, 1981]. Specifically when modeling bounded towing capacity, the movement of objects on the graph representation that we use fits, the capacitated multi-vehicle routing variant studied by Ralphs et al. [2003] closely. In the robotics literature, the most directly relevant work in this class is that of Mathew et al. [2015b] who transport objects with multiple heterogeneous robots. (Also, their reduction to the TSP (Travelling salesman problem) was a valuable guide for us in producing the NP-hardness result in Section 4.)

The present paper is perhaps best characterized as task planning for multiple robots, as distinct from the concerns of multi-robot path planning (see, e.g., van den Berg et al. [2009], Luna and Bekris [2011], Wagner and Choset [2015]); though, in recent years, this boundary has begun to be blurred by work such as Turpin et al. [2014], Solovey and Halperin [2014], Vega-Brown and Roy [2016]. We note, particularly, that Solovey and Halperin [2014] introduced the $k$-color planning problem recently, which deals with interchangeable tasks; our work allows for object changeability as expressible by colors, but also more sophisticated specifications too. Recently, several researchers [Hung et al., 2014, Nedunuri et al., 2014, Dantam et al., 2016] have explored Satisfiability Modulo Theories (SMT)-solvers [De Moura and Bjørner, 2008, Bouton et al., 2009] for combining task and motion planning. SMT allows first-order logic formulas to be extended to give a powerful, expressive and high-level interface for representing constraints in robotics problems [De Moura and Bjørner, 2011]. We use these sorts of formulas to express constraints and general goals that describe complex object transportation problems.

In order for our robots to link their tails, they rendezvous with one another. Both docking and rendezvous planning [Roh et al., 2008, Mathew et al., 2015a, Kim et al., 2012] have been previously studied for coordination of multi-robot systems, though not for physically linking robots.

### 2.1 Loosely-coupled and tightly-coupled coordination

Discussions of multi-robot systems often use qualifiers like 'loosely-coupled' or 'tightly-coupled' to indicate the frequency of information exchange. In the present work, when the robots operate as conjoined pairs, they are literally physically coupled so that the motions of one robot directly constrains the other. The cases where robots operate as separate individuals certainly involves less direct interaction. In this regard, this work represents an instance where the planner is automatically making decisions about whether loose or tight coordination is best, rather than having a single form of coordination predetermined beforehand.

## 3 Problem Description

### 3.1 Problem Setup and Notation

Let the set $\mathcal{O} = \{o_1, o_2, \ldots, o_n\}$ represent $n$ objects which can be transported by our $m$ robots: $\mathcal{R} = \{r_1, r_2, \ldots, r_m\}$. We will write $\mathit{pos}^t(o_i)$ to denote the position of object $i$ at time $t$, (and similarly for robots); $t \in \mathbb{N}_0$ is a temporal index, which starts at zero. We will use the term *world configuration* at time $t$, to describe the positions of all objects and robots at that instant. A set of goal locations, $\mathcal{G} = \{g_1, \ldots, g_k\}$, is given. By extending *pos* in the natural way, $\mathit{pos}^t(g_i) = X^G_i, \forall t$ gives $g_i$'s (fixed) position. We also find it helpful to write the initial positions of the objects as $X_i$ with $X_i = \mathit{pos}^0(o_i)$, and robots' initial locations with $X^R_i = \mathit{pos}^0(r_i)$. Finally, we collect all these initial positions into the set $\mathcal{X} = \{X_1, \ldots, X_n, X^R_1, \ldots, X^R_m, X^G_1, \ldots, X^G_k\}$. To describe the mass of each object, we will write the function $\mathit{mass}(o_i)$ to denote the mass for $o_i$, then let the set $\mathcal{D} = \{\mathit{mass}(o_1), \ldots, \mathit{mass}(o_n)\}$.

Along with positions, each robot's state represents that it is either operating alone or has its tail connected to another robot. Let $\mathit{match}^t(r_i) = r_j$, with $i = j$ if and only if $r_i$ is operating solo and with $i \neq j$ if and only if $r_i$ and $r_j$ are coupled together to form a conjoined pair. We partition $\mathcal{R}$ into $\mathcal{R}^t_s \cup \mathcal{R}^t_p$, where solo robots comprise set $\mathcal{R}^t_s$, and the pairs $\mathcal{R}^t_p$.

Further, assume two sets of logical predicates are given:

$\mathfrak{L}_{\text{static}}$ contains predicates, like $\text{P} : \mathcal{O} \rightarrow \{\text{True}, \text{False}\}$, describing static properties of the objects; e.g., $\text{PINK}(\cdot)$, $\text{CYLINDER}(\cdot)$, $\text{WOOD}(\cdot)$, $\text{BOOK}(\cdot)$, $\text{BALL}(\cdot)$, etc.

$\mathfrak{L}_{\text{pos}}$ with properties defined as $\text{P}' : \mathcal{O} \times \mathcal{X} \rightarrow \{\text{True}, \text{False}\}$, that include position information; e.g., $\text{LOCATEDAT}(\cdot, \cdot)$.

A set of desired final states is specified via a logical formula, $\mathbf{F}$, written in terms of $\mathfrak{L}_{\text{static}}$ and $\mathfrak{L}_{\text{pos}}$, which, when satisfied in some world configuration, distinguishes it as a goal. Rather than belaboring the point by providing a complete Backus-Naur Form characterization of the formulas, it suffices to state that the standard logical connectives ($\neg$, $\wedge$, $\vee$) can be used. Additionally, SMT also permits one to write relationships between the cardinalities of sets (and natural numbers) with $\leq, <, \neq$, etc. Note that the goal locations $\{X^G_1, X^G_2, \ldots, X^G_k\}$ enter into specifications as part of the domain of predicates in $\mathfrak{L}_{\text{pos}}$.

Continuing informally, we provide semantics by mapping to the domain of discourse $U$. Let the set $\mathbb{O} \triangleq \{\mathit{obj}_1, \mathit{obj}_2, \ldots, \mathit{obj}_n\}$ represent $n$ objects. Let the set $\mathbb{L} \triangleq \{\mathit{loc}_1, \mathit{loc}_2, \ldots, \mathit{loc}_v\}$ represent $v$ named locations.

We define interpretation $\mathcal{I}$ at time $t$, which maps constant symbols to elements, and predicate symbols to relations on the domain of discourse as:

$$\mathcal{I}^t = \begin{cases} obj_i \in \mathbb{O} \mapsto o_i \in \mathcal{O}, \\ loc_i \in \mathbb{L} \mapsto X_i \in \mathcal{X}, \\ \text{LOCATEDAT} \mapsto \{\langle o_i, X_k \rangle \mid o_i \in \mathcal{O}, X_k \in \mathcal{X},\ pos^t(o_i) = X_k\}, \\ \text{etc.} \end{cases} \tag{1}$$

As robots and objects are moved, the interpretation undergoes dynamics to model these transitions as they occur. The predicate and constant symbols provide a discrete representation of the spatial configurations, and they evolve in discrete time. Next, we relate these elements to a graph representation on which search algorithms operate.

## 3.2 The MOCCT Problem Formulation

We formulate the MOCCT problem as a task planning problem using high-level motion primitives [LaValle, 2006] for the robots' movements. The sequence of primitives comprises a motion plan having semantics defined in terms of a configuration-space (c-space) motion (details are below).

Throughout this paper, manipulation proceeds, whether by a single robot or conjoined pairs, by encircling a payload and then towing it to some position. Once multiple objects are brought together, the robots are incapable of separating them again.

*Modeling assumption:* We impose the natural restriction that all the objects at each $X \in \mathcal{X}$ must be moved together. It is useful to have notation for the set of all objects at some location, with the mnemonic for a bundle, we write $b^t_X = \{o_i : \mathcal{O} \mid pos^t(o_i) = X\}$, where $X \in \mathcal{X}$; the $X$ for $b^t_X$ is the bundle's site. One consequence of the preceding modeling assumption is that we do not consider plans of sequences longer than the total number of objects; we denote this maximum $\bar{T}$. Another consequence is that we can talk meaningfully about multiple objects at $X \in \mathcal{X}$, because we treat them as indivisible thereafter so their centroid suffices to characterize the state that is important to the robots.

The search space is now completely specified. Search proceeds over world configurations, with each element of the search space being a set of bundles, their associated sites, and the robots' state. We define planning operators which transition one world configuration to another. It is probably easiest to understand the world configurations and the valid transitions between them by visual means. Figure 2 illustrates the scenario of Figure 1(a). Bundles of objects appear in braces at each location. If, at time $t$, elements from site $X_i$ are moved to $X_j$, the result of the operation will be $b^{t+1}_{X_j} = b^t_{X_i} \cup b^t_{X_j}$. The figure abstracts away several details, but these are clarified by looking at the planning operators in terms of the motion primitives the robots use to mediate their operation in the environment. In the following four primitives, we use $\mathbf{r}$ to denote a set of robots, either a singleton $\mathbf{r} = \{r_k\}$, $r_k \in \mathcal{R}^t_s$, or a pair $\mathbf{r} = \{r_k, r_\ell\}$, $\mathbf{r} \subseteq \mathcal{R}^t_p$.

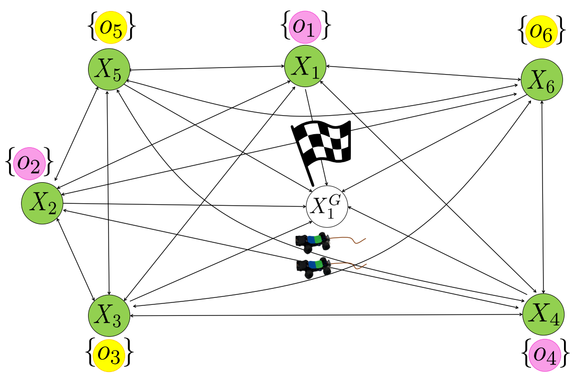
1. TRANSIT$(\mathbf{r}, X_i)$: This primitive returns a path $\Delta_1$ for moving robot(s) in $\mathbf{r}$ to $X_i$ without colliding with obstacles and objects. It changes the robot(s) locations to $X_i$, initializing their orientation so they are ready to execute TRANSFER$(\cdot)$.

2. TRANSFER$(\mathbf{r}, X_i, X_j)$: This primitive returns a path $\Delta_2$ for collision-free motion of robot(s) in $\mathbf{r}$ so that all $o_k \in b^t_{X_i}$ arrive at $X_j$. This changes the locations of both robot(s) and object(s) to $X_j$.

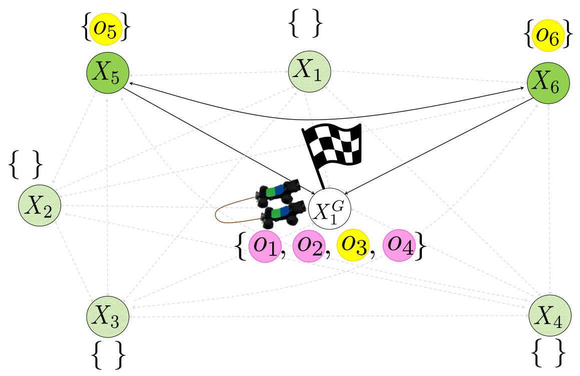
3. DOCK$(\mathbf{r}, X_i)$: For $\mathbf{r} = \{r_k, r_\ell\}$, $\mathbf{r} \subseteq \mathcal{R}^t_s$, this primitive returns a path $\Delta_3$ which, when executed, links $r_k$ and $r_\ell$ at $X_i \in \mathcal{X}$. This changes the configuration of robots, and places both $r_k, r_\ell$ in $\mathcal{R}^t_p$, $match^t(r_k) = r_\ell$ and $match^t(r_\ell) = r_k$.

4. SPLIT$(\mathbf{r})$: This primitive returns a path $\Delta_4$ for two robots $\mathbf{r} = \{r_k, r_\ell\}$, $\mathbf{r} \subseteq \mathcal{R}^t_p$, to split their previously joined tails at their current location. The operator is the inverse of DOCK. When executed, unlinks $r_k$ and $r_\ell$ at $X_i \in \mathcal{X}$. This changes the configuration of robots, and places both $r_k$, $r_l$ in $\mathcal{R}^t_s$, $match^t(r_k) = r_k$ and $match^t(r_\ell) = r_\ell$.

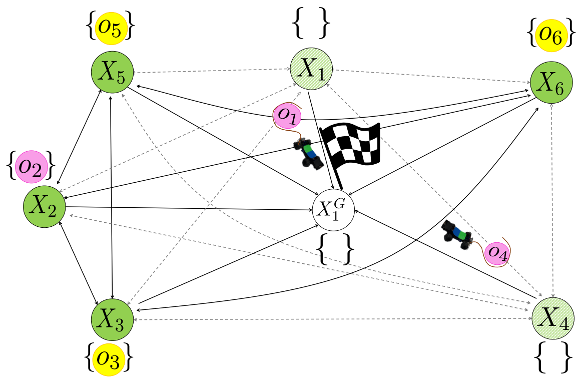
If the requirements on $\mathbf{r}$ are not met in 3. and 4., the SPLIT or DOCK primitive cannot be used in that context. The primitives are sequenced in triples to give high-level operators which transform world configurations. We write $\pi^t(\mathbf{r}, X_i, X_j)$, for a sequence that moves the objects at $X_i$ to $X_j$: either $\pi^t(\mathbf{r}, X_i, X_j) = [\Delta_4, \Delta_1, \Delta_2]$ for $|\mathbf{r}| = 1$, or otherwise $\pi^t(\mathbf{r}, X_i, X_j) = [\Delta_3, \Delta_1, \Delta_2]$ for a pair of robots, i.e. $|\mathbf{r}| = 2$, if such collision-free paths exist



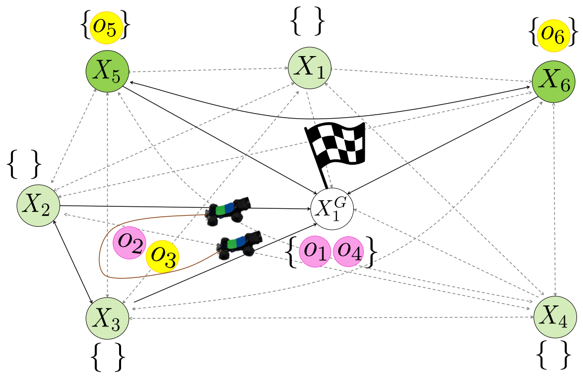
(a) The initial configuration at $t = 0$.



(b) A goal configuration at $t = T$.



(c) First step: two solo robots transfer $o_1$ and $o_4$ to the goal concurrently.



(d) Second step: a pair transfer $o_2$ and $o_3$ to the goal working together.

Figure 2: A directed graph representation for the example in Figure 1(a). A green circle indicates a vertex $V_{X_i}$ and a curly bracket shows a set of objects $b^t_{X_i}$. Here (a) represents an initial configuration while (b) is a goal configuration; (c) shows the first step in Figure 1(a), while (d) shows the second step in Figure 1(b). Heuristics mean some edges (dotted) are unlikely to be explored.

and can be found by our motion planner. In the preceding, the $\Delta_x$'s give paths for the subset of robots involved, namely, $\mathbf{r}$. When $\pi^t(\cdot,\cdot)$'s have been constructed for all $m$ robots in $\mathcal{R}$, which we write as $\Pi^t(\cdot,\cdot)$, the result is a path in the joint c-space for the team.

Let $J$ be a deterministic cost function estimating, say, the time to execute a path. Then, to define a collective cost $\mathcal{J}$, we lift $J$ to $\Pi^t$:

$$\mathcal{J}(\Pi^t) \quad = \underset{\pi^t(\mathbf{r},X_i,X_j)\in\Pi^t}{\textsc{Combine}} \begin{cases} J(\Delta_4) + J(\Delta_1) + J(\Delta_2) & \text{if } |\mathbf{r}| = 1, \\ J(\Delta_3) + J(\Delta_1) + J(\Delta_2) & \text{if } |\mathbf{r}| = 2, \end{cases} \tag{2}$$

where COMBINE is a function that describes the way concurrent operations are aggregated for the optimization objective being considered. We use COMBINE in two ways:

- COMBINE = $\max$ for robots that stay idle until others complete their tasks.
- COMBINE = $\sum$ for computing a cumulative total navigation distance for all robots as the cost function.

A few additional elements must be introduced into the model. Firstly, let $L^i$ be the tail length, a physical parameter, for each robot $i$. Secondly, we define two predicates to deal with towing constraints as follows:

$$\begin{cases} C_1 : \mathcal{R} \times 2^{\mathcal{O}} \rightarrow \{\text{True, False}\}, \\ C_2 : \mathcal{R} \times \mathcal{R} \times 2^{\mathcal{O}} \rightarrow \{\text{True, False}\}, \end{cases} \tag{3}$$

where $C_1(r_i, \mathcal{O}_p = \{o_1, \ldots, o_k\})$ is true if hooking actions by a robot $r_i$ can move all objects in $\mathcal{O}_p$ and $C_2(r_i, r_j, \mathcal{O}_p = \{o_1, \ldots, o_k\})$ is true if dragging primitives by pairs $r_i$ and $r_j$ can move all objects in $\mathcal{O}_p$.

The following makes explicit assumptions which have been tacit up to this point:

(i) All our mobile robots have an inelastic tail.

(ii) Robots can link and unlink the ends of their tails, but only to form a pair.

(iii) The tail cannot pass through any of the objects when the tail contacts the objects.

(iv) The robots' and the objects' states are observable, while the tail configurations are not.

In the work hereafter, we are not interested in infeasible plans and so consider systems where there is at least one robot (or one pair of robots) that can manipulate objects, ultimately leading to a state satisfying $\mathbf{F}$.

**General Problem Definition**: A Multi-Object Collection via Cooperative Towing (MOCCT) Planning Problem: Given $\mathcal{O}, \mathcal{R}, \mathcal{G}, \mathcal{X}, \mathcal{D}, J, \mathfrak{L}_{\text{static}}, \mathfrak{L}_{\text{pos}}$, and $\mathbf{F}$, COMBINE, with $C_1$ and $C_2$ for all robots, compute a sequence of paths $(\bar{\Pi}^0, \ldots, \bar{\Pi}^T)$ which minimizes the total delivery cost, where the capacity constraints are never violated and the goal predicate $\mathbf{F}$, is satisfied in the configuration at time $T$. Formally:

$$[\bar{\Pi}^0, \ldots, \bar{\Pi}^T] = \underset{[\Pi^0,\ldots,\Pi^T]}{\arg\min} \sum_{t=0}^{T} \mathcal{J}(\Pi^t), \tag{4}$$

subject to

$$\models_{\mathcal{I}^T} \mathbf{F}, \tag{5}$$

$$T \leq \bar{T}, \tag{6}$$

$$\left\{ \begin{array}{ll} C_1(r_k, b^t_{X_i}) = \text{True}, & \mathbf{r} = \{r_k\} \\ C_2(r_k, r_p, b^t_{X_i}) = \text{True}, & \mathbf{r} = \{r_k, r_p\} \end{array} \right\}, \forall t \in \{0, \ldots, T\}, \forall \pi^t(\mathbf{r}, X_i, X_j) \in \Pi^t. \tag{7}$$

The discrete formulation leads to a single-objective combinatorial optimization problem, where the search space contains the feasible, cooperative paths for all robots.

## 4 NP-hardness of the MOCCT Problem

We show that, quite separately from the satisfiability of $\mathbf{F}$, the preceding combinatorial optimization problem is NP-hard. To do so we define a Trivially Satisfiable Multi-Object Collection via Cooperative Towing problem (TS-MOCCT), which is simplified in that all objects are to be moved to a single goal. Thus, the logical formula for TS-MOCCT has the special trivial structure $\mathbf{F}_{TS} := \forall i, \text{LOCATEDAT}(o_i, X^G_1)$.

To show the hardness of the TS-MOCCT problem, we will show that (i) an instance of the Vehicle Routing Problem (VRP), known to be NP-hard [Lenstra and Kan, 1981], can be reduced to an instance of TS-MOCCT, and (ii) an optimal TS-MOCCT solution can be used to generate an optimal VRP solution.

We describe the VRP [Dantzig and Ramser, 1959, Lenstra and Kan, 1981] briefly: an amount of some commodity $d_i$ is to be delivered to each customer $o'_i \in O$ where $i \in \{1, \ldots, n'\}$ from a central depot $g'_0$ using $m'$ independent delivery vehicles $R = \{r'_1, \ldots, r'_{m'}\}$. Let $p_i$ be the position of customer $o'_i$. The central depot $g'_0$ is at $p_0$. Delivery is to be accomplished with the minimum total cost. $\mathcal{J}'(p_i, p_j)$ denotes the transit cost from $p_i$ to $p_j$. The goal of VRP is to find a partition of $n'$ customers into $m'$ cycles $\{s_1, \ldots, s_{m'}\}$ whose only intersection is the depot node (starting and ending at the central depot). Overall, VRP = $\langle \mathcal{O}' = \{o'_1, \ldots, o'_{n'}\}, \mathcal{R}' = \{v_1, \ldots, v_{m'}\}, \mathcal{G}' = \{g'_0\}, \mathcal{X}' = \{p_0, p_1, \ldots, p_{n'}\}, \mathcal{D}' = \{d_1, \ldots, d_{n'}\}, \mathcal{J}' \rangle$

**Lemma 1** *The* TS-MOCCT *problem is NP-hard.*

*Proof* : We give a polynomial-time transformation of VRP $= \langle \mathcal{O}', \mathcal{R}', \mathcal{G}', \mathcal{X}', \mathcal{D}', \mathcal{J}' \rangle$ into the TS-MOCCT problem $= \langle \mathcal{O}, \mathcal{R}, \mathcal{G}, \mathcal{X}, \mathcal{D}, C_1, C_2, J, Combine \rangle$ We map VRP inputs to an instance of the MOCCT problem as follows:

1. $n = n'$, then $\mathcal{O} = \mathcal{O}'$ and $\mathcal{D} = \mathcal{D}'$.

2. We set $C_2 =$ true and $C_1 =$ false for all robots.

3. We consider only pairs of robots, which was already enforced by picking $C_1$ = false. We let $m = 2m'$, so $\mathcal{R} = \{r_1, \cdots, r_{2m'}\}$.

4. We have a single goal, then $\mathcal{G} = \{g'_0\}$ for the TS-MOCCT problem.

5. We have the set of positions $\mathcal{X} = \{X_1, \ldots, X_{n'}, X^R_1, \ldots, X^R_m, X^G_1\}$, where $X_i = p_i$ for $i \in \{1, \ldots, n'\}$. The robots' initial locations $X^R_i$ map to $p_0$ for all $i$. The goal location $X^G_1$ maps to $p_0$.

6. The cost function from $X_i$ to $X_j$ for a pair of robots is $J(\Delta_3) + J(\Delta_1) + J(\Delta_2)$, which results in the collective cost, $\mathcal{J}(X_i, X_j)$, mapping to $\mathcal{J}'(p_i, p_j)$.

7. We set COMBINE = $\sum$ for computing a total navigation distance for all robots.

This transformation defines all inputs of the VRP problem in terms of the TS-MOCCT problem.

Next, we show that an optimal TS-MOCCT solution corresponds to the optimal VRP solution. We consider the optimal TS-MOCCT solution $\{\bar{\Pi}^0, \ldots, \bar{\Pi}^T\}$ for $m'$ pairs of robots. The optimal solution has that each pair of robots can start and end at the goal location. No optimal solution involves any pair of robots returning to the goal during travel (except the very start and end). Recall that the TS-MOCCT problem imposes the constraint (as a modeling assumption) that all the objects at $X$ must be moved together. Thus, the TS-MOCCT solution consists of $m'$ cycles (potentially empty) for each pair of robots with no repeated nodes except the goal location $X^G_1$. We rewrite the solution in terms of the $m'$ cycles:

$$\begin{aligned}[\bar{\Pi}^0, \ldots, \bar{\Pi}^T] = [\bar{\Pi}_1(X_0, X_i)^0, \ldots, \bar{\Pi}_1(X_j, X_0)^T], \ldots,\\ [\bar{\Pi}_{m'}(X_0, X_k)^0, \ldots, \bar{\Pi}_{m'}(X_p, X_0)^T]\end{aligned} \tag{8}$$

Since we have exactly $m'$ pairs, we can see that the cost functions are equivalent when we rewrite the summation over cycles.

$$\sum_{k=1}^{m'} \sum_{X_i, X_j \in \bar{\Pi}_k} \mathcal{J}(\bar{\Pi}_k(X_i, X_j)) = \sum_{k=1}^{m'} \sum_{(p_i, p_j) \in s_k} \mathcal{J}'_k(p_i, p_j). \tag{9}$$

Thus, an optimal solution of the TS-MOCCT problem must also be an optimal solution of the VRP.

**Theorem 1** *The* MOCCT *problem is NP-hard*

*Proof* : From Lemma 1, we proved the TS-MOCCT problem is NP-hard. The MOCCT problem is the generalized version of the TS-MOCCT, which can have more goal locations with complex goal conditions,TS-MOCCT $\subseteq$ MOCCT. Thus the MOCCT problem is NP-hard.

## 5 Algorithms for the MOCCT problem

The space of all possible motion combinations for multiple robots and objects grows exponentially and so exhaustive search is prohibitive even for small instances. Consequently, we are compelled to introduce heuristic simplifications. In this section, we describe two algorithms we used for the MOCCT problem:

(1) We seek to produce solutions of reasonable quality in limited time via A$^\star$ search. The planner minimizes time to complete the task by searching over the full solution space, but is guided by heuristics.

(2) We propose a more efficient algorithm which prunes some of the choices available to the algorithm in (1), but which risks potentially overlooking plans with minimum cost.

## 5.1 The Basic Heuristic Search Algorithm

The algorithms operate on a tree structure describing the objects and robots configuration: each node has the robots' current states (positions in $\mathcal{X}$ and $match^t(\cdot)$), and the objects' current states (positions in $\mathcal{X}$). Figure 3 shows how the search tree is constructed by using the example of Figure 1. The initial configuration $\mathcal{N}_0$ (shown in Figure 2(a)) is the root node in Figure 3. Then, we expand a tree node based on all possible choices satisfying the constraints (*e.g.,* tail length and capacities).

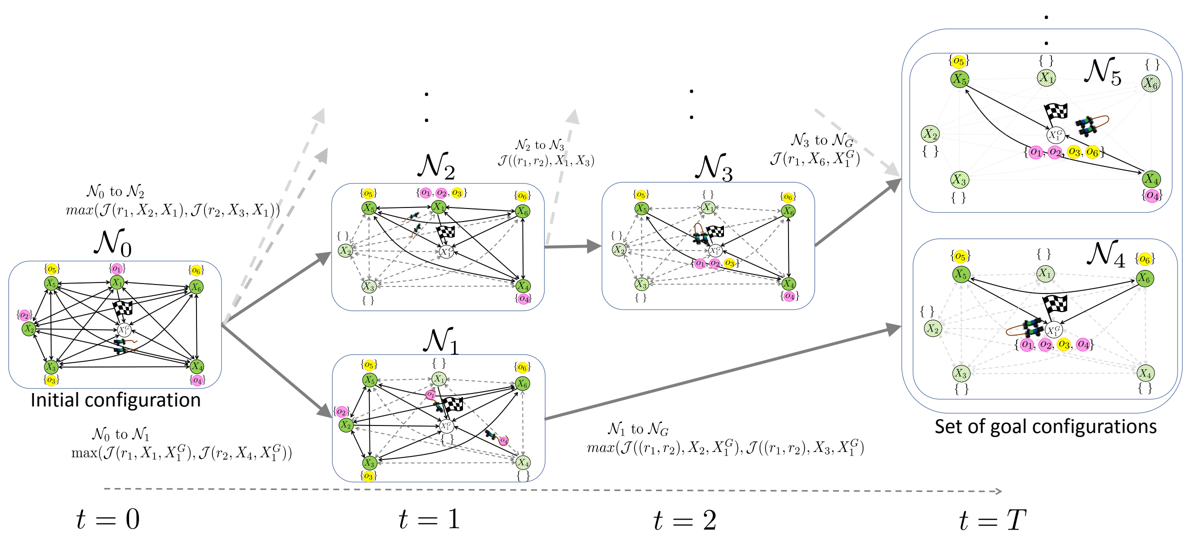
An example helps to understand the expansion of tree nodes: take the two sequential steps of Figure 1. The first step shows that $r_1$ transfers $o_1$ at $X_1$ to the goal, $X_1^G$, while $r_2$ transfers $o_4$ at $X_4$ to the goal, $X_1^G$. This transitions from $\mathcal{N}_0$ to $\mathcal{N}_1$ in Figure 3, where the transition cost is $\max(\mathcal{J}(r_1, X_1, X_1^G), \mathcal{J}(r_2, X_4, X_1^G))$. The second step shows a pair of robots $(r_1, r_2)$ transferring $b_{X_2}$ and $b_{X_3}$ to the goal, that is, a transition from $\mathcal{N}_1$ to $\mathcal{N}_4$ in Figure 3, with cost $\max(\mathcal{J}\left((r_1, r_2), X_2, X_1^G\right), \mathcal{J}\left((r_1, r_2), X_3, X_1^G\right))$.

We see that there exist multiple goal configurations satisfying $\mathbf{F}$ (in this problem instance we require four objects at the goal location, with at least two objects that are pink). For example, $\mathcal{N}_0$, $\mathcal{N}_2$, $\mathcal{N}_3$, and $\mathcal{N}_4$ are transitions that reach another goal configuration in Figure 3.

Assuming the towing capacity (maximum mass) is not exceeded, a pair of robots operating together is capable of moving more than one set of objects at multiple locations to $X_j$. In executing the second step from Figure 2(d), the pair move $o_2$ from $X_2$ and $o_3$ from $X_3$ to $X_1^G$ in one action. The search tree must include these kinds of actions because they represent part of the real cost savings that one obtains from pairs. Thus, to add extra edges departing $X_i$ of this sort, we look for nearby $X_j$'s, sorting these in ascending order by Euclidean distance. Let $\hat{B}_{X_i}$ be the set of objects nearby $X_i$. Here "nearby" includes only those no more than the length of the tail away.

Figure 4 gives an example of how these nearby objects introduce new choices. Assume the pair pick a route to transfer object(s) at $X_1$ to the goal $X_1^G$. There are four subsets of $\hat{\mathcal{B}}_{X_1}$ in Figure 4. For example, the 'subset 1' line shows one sibling, indicating $\hat{\mathcal{B}}_{X_1} = [o_1, o_5]$, in ascending order. Similarly, the 'subset 2' line shows three siblings, indicating $\hat{\mathcal{B}}_{X_1} = [o_1, o_5, o_4]$. The 'subset 3' line shows four siblings, indicating $\hat{\mathcal{B}}_{X_1} = [o_1, o_5, o_4, o_3]$. Lastly, the 'subset 4' line shows five siblings, $\hat{\mathcal{B}}_{X_1} = [o_1, o_5, o_4, o_3, o_2]$. We define a cost for this action, which we take to be the perimeter of the convex hull of $\hat{B}_{X_i}$. In Figure 4, the cost of the 'subset 4' is the total distance of the outer positions $X_1$, $X_2$, and $X_3$. The transfer cost for $o_4$ and $o_5$ is zero because they are inside of the convex hull, so are transported effectively for free. Similarly, the cost of the 'subset 3' is the total distance among $X_1$, $X_4$, and $X_3$, and no cost is incurred for $o_5$.

The basis of Algorithm 1 is A$^\star$ search, extended to include an implicit search for these additional nearby objects. The first phase, in line 4, computes the set of nearby objects, $\hat{\mathcal{B}}_{X_i}$, for all $X_i \in \mathcal{X}$, satisfying the tail length



$\mathcal{N}_0$ to $\mathcal{N}_2$
$max(\mathcal{J}(r_1, X_2, X_1), \mathcal{J}(r_2, X_3, X_1))$
$\mathcal{N}_0$
Initial configuration
$\mathcal{N}_0$ to $\mathcal{N}_1$
$\max(\mathcal{J}(r_1, X_1, X_1^G), \mathcal{J}(r_2, X_4, X_1^G))$
$\mathcal{N}_2$
$\mathcal{N}_1$
$\mathcal{N}_2$ to $\mathcal{N}_3$
$\mathcal{J}((r_1, r_2), X_1, X_3)$
$\mathcal{N}_3$
$\mathcal{N}_3$ to $\mathcal{N}_G$
$\mathcal{J}(r_1, X_6, X_1^G)$
$\mathcal{N}_1$ to $\mathcal{N}_G$
$max(\mathcal{J}((r_1, r_2), X_2, X_1^G), \mathcal{J}((r_1, r_2), X_3, X_1^G)$
$\mathcal{N}_5$
$\mathcal{N}_4$
Set of goal configurations
$t = 0$
$t = 1$
$t = 2$
$t = T$

Figure 3: Construction of a plan by searching a tree from the left ($t = 0$) to the right ($t = T$). Particular sets for each $\mathcal{O}$, $\mathcal{R}$, and $\mathcal{G}$ comprise the nodes.

constraints and the capacity constraints of robots. Then, it starts to enumerate the possible sets (lines 12–20). Given $m$ robots, it constructs the possible combinations of either role (independent or as half a pair). Then, it finds all combinations of choices for $\mathbf{r}$, called $\hat{A} \in A$, which a kind of satisfy the capacity constraints in Equation (7). In line 14 of Algorithm 1, the set is increased by including nearby objects (following the sequential ordering of $\hat{\mathcal{B}}_{X_i}$) until the maximum capacity $C_2$ is exceeded, and $\mathbf{F}$ is impossible to achieve. For example, in Figure 4, we pick the basis set $\{o_1\}$ first, and then increase the range by accumulating an element of $\hat{\mathcal{B}}_{X_i}$ to have the red area $[o_1, o_5]$, the blue area $[o_1, o_5, o_4]$, then the green one $[o_1, o_5, o_4, o_3]$, and finally the purple set $[o_1, o_5, o_4, o_3, o_2]$. In line 15 of Algorithm 1, we have to match a set of robots with a set of choices. We use the Hungarian method [Kuhn, 1955] to find the optimal matching. In line 16, the maximum (not sum of) distances of $\mathbf{r}$ at the current step is used because all the robots are working in parallel, and we add the heuristic estimate: the total distance between $X_i$ and goals, $\sum_{i\in[1,n]} \min_{j\in[1,k]}(pos^t(o_i), X_j^G)$.

## 5.2 The Opportunistic Neighborhood Search (ONS) Algorithm

To reduce the search cost, we reduce the number of outgoing choices induced by the set operation on nearby objects described in the previous section. We do this by being greedy in a manner we feel represents a kind of opportunism. This algorithm is a modification of the basic algorithm, in which we do not search over all possible subsets of objects in line 14, we instead pick the maximum sized set of objects satisfying constraints. This is efficient if we have objects that are located within the boundary of the tail length.

## 6 Experiments

The basic heuristic search (called *Basic*) and the opportunistic neighborhood search (called *Opportunistic*) algorithms produce solutions for moderate numbers of objects and robots in a reasonable time. *Opportunistic* produce solutions for even larger number of objects and robots, though precise difference in costs depend on the constraints (*e.g.,* a longer tail with larger capacities). All algorithms were executed on an Intel Core i7 3.2GHz, and implemented in Matlab with a SMT-solver [Bouton et al., 2009]. For comparison purposes, we also implemented an uninformed breadth first search, which we call *Exact*.

We have three goal predicates to validate our approach:

1. $\mathbf{F}_1$: transferring all PINK and CYLINDER objects to $X_1^G$.

$$\mathbf{F}_1 := \forall i, \textsc{pink}(o_i) \wedge \textsc{cylinder}(o_i) \wedge \textsc{locatedAt}(o_i, X_1^G). \tag{10}$$

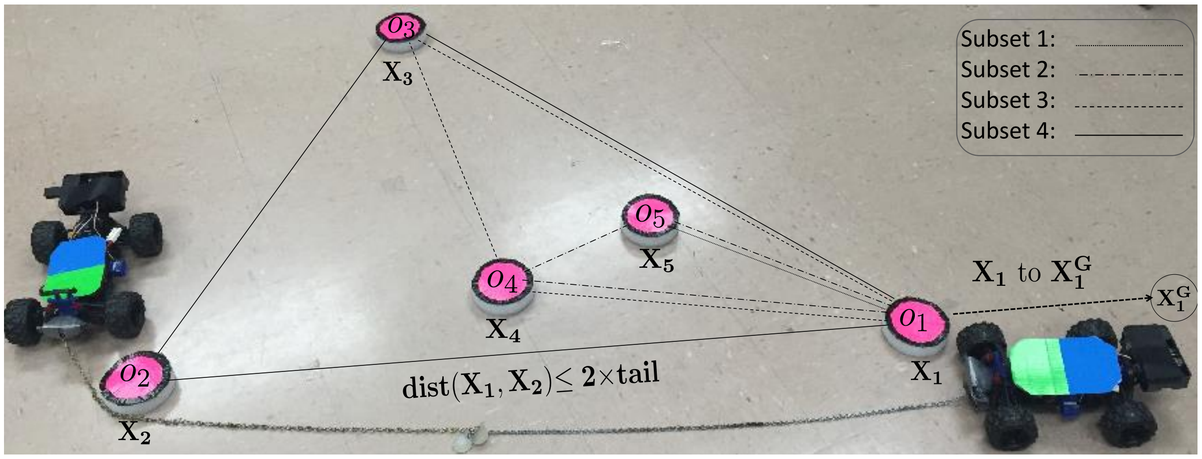


Figure 4: Five objects induce several choices. There are multiple feasible subsets of $\hat{\mathcal{B}}_{X_1}$: The ‘subset 1’ line shows $[o_1, o_5]$. The ‘subset 2’ line shows $[o_1, o_5, o_4]$. The ‘subset 3’ line shows $[o_1, o_5, o_4, o_3]$. Finally, the ‘subset 4’ line shows $[o_1, o_5, o_4, o_3, o_2]$.

**Algorithm 1** Basic_Search

```
1: INPUT: $\mathcal{O}, \mathcal{R}, \mathcal{G}, \mathcal{X}, \mathcal{J}, \mathcal{R}, \mathcal{D}, \mathbf{F}$
2: OUTPUT: $(\bar{\Pi}^0, \ldots, \bar{\Pi}^T)$
3: Q = $\mathcal{N}_0 = [pos^0(\mathcal{O}) \cup pos^0(\mathcal{R}) \cup pos^0(\mathcal{G})]$ and ClosedSet = $\varnothing$
4: Compute $\hat{\mathcal{B}}_{X_i}, \forall i$
5: while Q $\neq \varnothing$ do
6:   $\mathcal{N}_{curr} = POP(Q)$
7:   if $\mathbf{F} =$ true then
8:     return $(\bar{\Pi}^0, \ldots, \bar{\Pi}^T)$ from $\mathcal{N}_0$ to $\mathcal{N}_{curr}$
9:     break
10:  end if
11:  ClosedSet.add($curr$)
12:  for each of possible robot settings $\mathbf{r}$ of $\mathcal{R}$ do
13:    for each possible motions $\hat{A}$ of $A$ do
14:      for each possible $B_{X_i}$ of $\hat{\mathcal{B}}_{X_i}$ do
15:        $\mathcal{J}$ = Assignment($\mathbf{r}$,$\hat{A} \cup B_{X_i}$ )
16:        $\mathcal{N}_{near}$.cost =max($\mathcal{J}$) + h($\mathcal{N}_{near}$, $X_k^G$)
17:        Q = Q $\cup$ $\mathcal{N}_{near}$.
18:      end for
19:    end for
20:  end for
21: end while
```

2. $\mathbf{F}_2$: transferring four objects to $X_1^G$, where two objects are at least pink and $\textsc{num}(X)$ returns the number of objects at $X$.

$$\begin{aligned} \mathbf{F}_2 := \forall i, \forall j, i \neq j, \textsc{pink}(o_i) \wedge \textsc{pink}(o_j) \wedge \textsc{locatedAt}(o_i, X_1^G) \\ \wedge \textsc{locatedAt}(o_j, X_1^G) \wedge \textsc{num}(X_1^G) = 4. \end{aligned} \tag{11}$$

3. $\mathbf{F}_3$: transferring objects to $X_1^G$, $X_2^G$, or $X_3^G$, such that $0 < \textsc{num}(X_2^G) < \textsc{num}(X_1^G)$ and $\textsc{num}(X_3^G) = \textsc{num}(X_1^G) + \textsc{num}(X_2^G)$.

$$\begin{aligned} P_1 &:= 0 < \textsc{num}(X_2^G) < \textsc{num}(X_1^G), \\ P_2 &:= \textsc{num}(X_3^G) = \textsc{num}(X_2^G) + \textsc{num}(X_1^G), \\ \mathbf{F}_3 &:= P_1 \wedge P_2. \end{aligned} \tag{12}$$

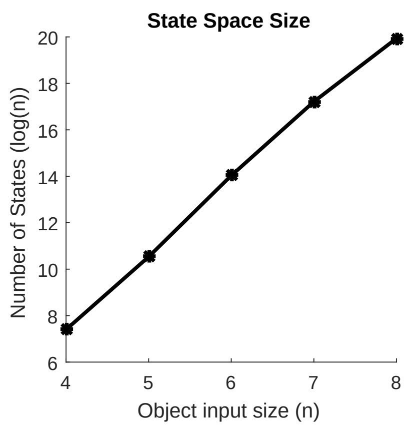
We use $\mathbf{F}_1$ for Section 6.1 and Section 6.2. Goals $\mathbf{F}_2$ and $\mathbf{F}_3$ are used to examine how the planner performs on more complex tasks, which is presented in Section 6.3.

## 6.1 Evaluation of algorithm running-time and performance

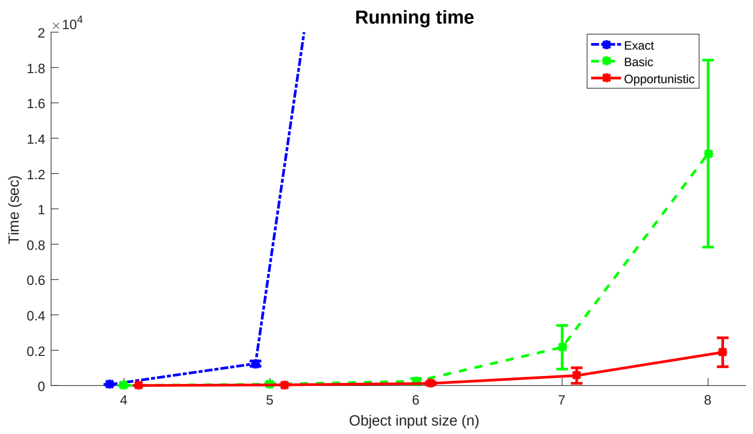
First, we show how the combinatorial aspects of the MOCCT problem means it has an enormous state-space even with few objects. The plot in Figure 5(a) shows this pictorially. For example, even 6 objects with fixed parameters (a tail length $L = 100\ cm$, $m = 2$, $C_1 = 1$, $C_2 = 3$, and *mass*$(o_i) = 1$ for all $i$) has over one million states.

Second, we evaluate the efficiency and performance of the two algorithms by randomly generating environments and measuring plan cost and computational time. The evaluation was conducted with the following fixed parameters: two tail robots, $C_1 = 1$, $C_2 = 3$, *mass*$(o_i) = 1$ for all $i$, and $L = 100\ cm$. All environments were $3\ m \times 3\ m$ open space regions, with a single goal. To test the algorithms, we increased $n$ from 4 to 8, maintaining a fixed ($m = 2$) number of robots. We generated 10 random instances for each number of objects.

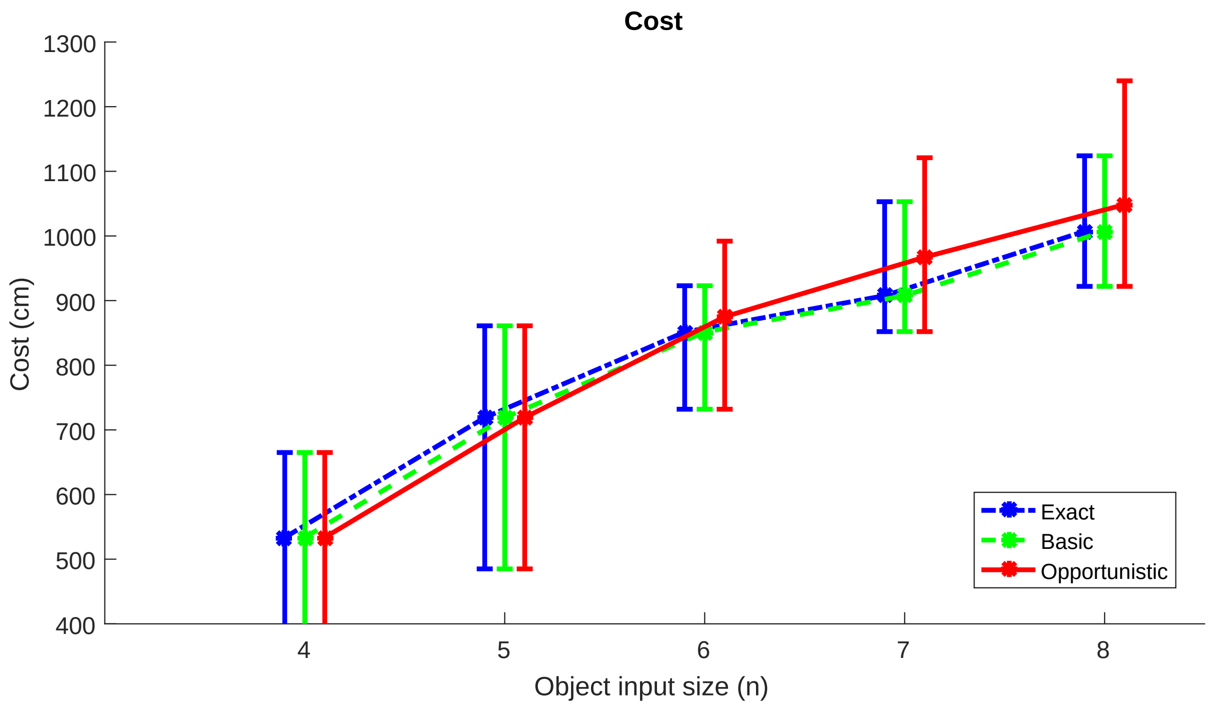
As visible in Figure 5(b), *Basic* finds an optimal solution in a reasonable time on small instances ($n \leq 6$); even a small problem has a huge search space (approximately a hundred thousand states when $n = 8$). In our tests, *Opportunistic* reduces a search space as reflected in its running-time, while the quality of solutions it finds remains high (see Figure 5(c)).



(a) State-space sizes for five settings



(b) Running times for five settings



(c) Solution costs for five settings

Figure 5: State-space sizes, running times, and solution costs of the MOCCT problem.

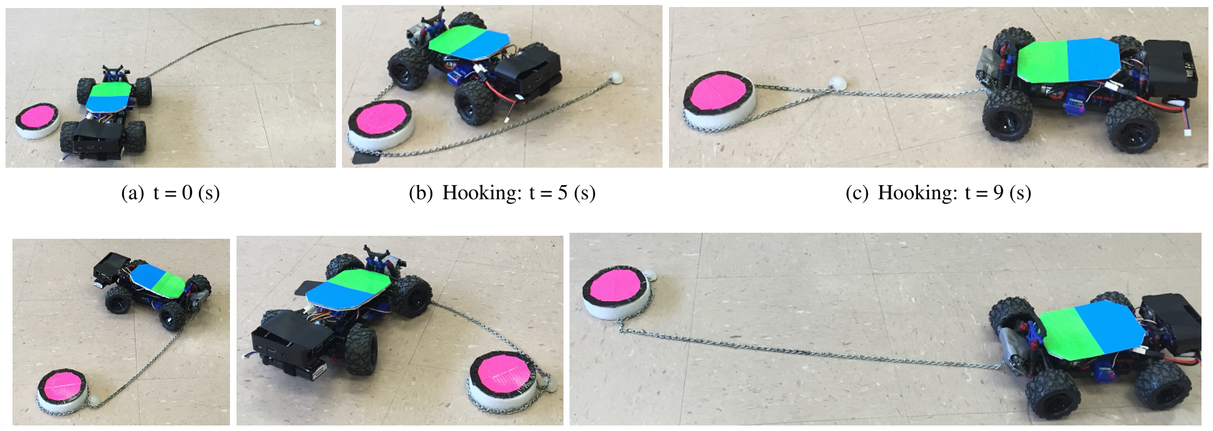
In light of these data, plans sought for planning problems in the remainder of the paper will use the ONS algorithm.

## 6.2 Physical robot experiments

Next, we demonstrate that the proposed cooperative manipulation approach works in practice.

### 6.2.1 System setup

We used two RC cars, controllable at velocities of approximately $0.4\ m/s$. An embedded computer on the car controls the robot and communicates with a separate computer integrated with an overhead tracking system which


(a) t = 0 (s) (b) Hooking: t = 5 (s) (c) Hooking: t = 9 (s)

(d) t = 0 (e) Unhooking: t = 3 (s) (f) Unhooking: t = 12 (s)

Figure 6: The robot winds its tail around the object with a clockwise direction through (a), (b) and (c). To release the hooked object, the robot drives in the opposite direction, seen in (d), (e) and, (f).

localizes the objects and the robots. Our software uses the ROS framework. All experiments are conducted in our test arena of size $5\ m \times 5\ m$.

### 6.2.2 Robot motion model and actions

We assume a simple car model for the robot. We generate the robot trajectories via Dubins curves that allow the shortest path in simple obstacle environments as in Giordano and Vendittelli [2009]. A Dubins curve consists of circular curves and linear motions, thus all elements of primitives in this paper are reduce to these atomic actions.

For a single robot, we use a hooking action in the TRANSIT$(\cdot)$ primitive, which sets up the robot's orientation to work on the TRANSFER$(\cdot)$ primitive. To produce this hooking action, we put a $\sim 3\ cm$ diameter half sphere at the end of the tail. Near each semi-sphere there is also a magnet. First, the robot winds around the object, crossing near the end of its tail (Figure 6(b)). Then, the half sphere works like a buckle (Figure 6(c)). Thereafter, the robot moves the hooked object via the TRANSFER$(\cdot)$ primitive. To release hooked objects, the robot turns around to the object reversing the direction of the winding motions (Figure 6(d) to Figure 6(f)).

For a pair of robots, we developed a DOCK$(\cdot)$ primitive. Firstly, given a subtask (move objects from $X_i$ to

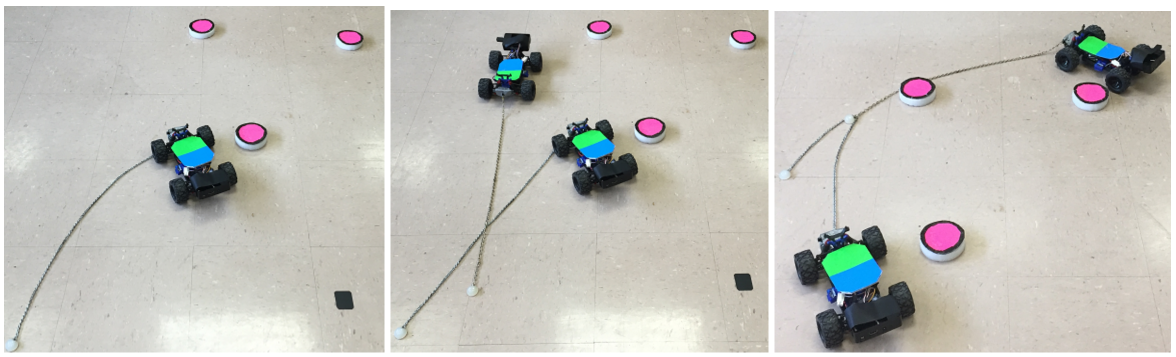

(a) t = 0 (s) (b) Docking t = 8 (s) (c) Docking t= 12 (s)

Figure 7: Initializing docking in (a). One robot stays at the given $X$. The second robot crosses the tail of the first. Since each tail, made of chain, has a magnet at the end, the two tails join naturally so long as the robot follows the contour of the convex hull via (b) to (c).

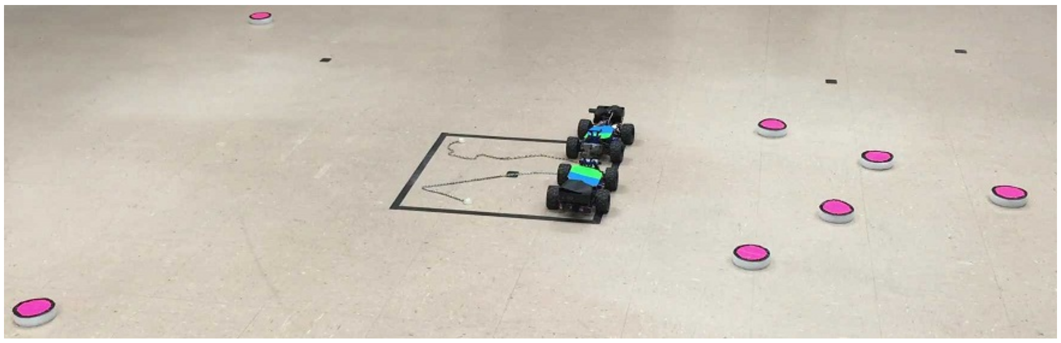


Figure 8: We have seven pink cylindrical objects to be towed to the center of the room with two robots.

$X_j$), one robot stays near $X_i$, initializing the robot's orientation for the $\text{TRANSFER}(\cdot)$ primitive (Figure 7(a)), then the second robot crosses the tail of the first (Figure 7(b)). Finally, the tails become connected (because each tail's magnet attaches to the other) as the robot follows the contour of the convex hull consisting of the outer positions $X_i$. See Figures 7(b) and 7(c). There after, we use the $\text{TRANSFER}(\cdot)$ primitive to move the gathered objects. To split conjoined tails, we simply execute an extra motion: one robot stops while the other moves past until the length of the tails is exceeded, breaking their connection.

#### 6.2.3 Three planning settings

To examine the value of robots operating individually, or as pairs, or opting to change roles dynamically, we conducted an experimental comparison with our physical robots. We considered three settings:

(1) The *single-only* setting has robots operating individually only.

(2) The *pair-only* setting has one pair of robots permanently connected together.

(3) In the *both* setting, the planner uses robots in either role and permits the making and breaking of pairs.

#### 6.2.4 Physical robot experiments: comparison of planners

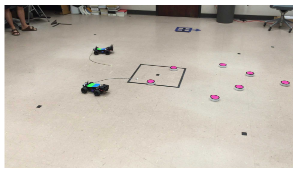
Figure 8 shows the basic setup: all trials with physical robots had $n = 7$ objects and two tail robots where $C_1 = 1$, $C_2 = 5$, $\textit{mass}(o_i) = 1$ for all $i$, and $L = 100\ cm$. The ONS algorithm was used to compute plans for all three planning settings. The total planning time for the *single-only* plan was 349 seconds, visiting a total of $4404$ states, while the *pair-only* plan took 123 seconds with $1684$ states and *both* took 24 seconds with $691$ states.

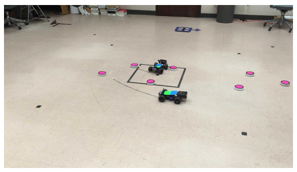
Figures 9 and 10 permit visual comparison of the plans in the three settings: *single-only*, *pair-only*, and *both* are shown in the first row, the second row, and the third row, respectively. For each row, from left to right, we see the snapshots of the physical experiments over time.

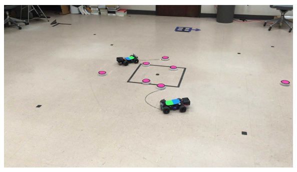
The *single-only* plan uses two robots acting individually and concurrently. There are $T = 4$ transitions in the solution, completing this task in $285$ seconds. The snapshot at the time of the final object's arrival appears in Figure 10(a). The robots and object motions are visible: blue lines record the robots' trajectories and the pink lines are the objects' motions. Since each robot needs space to release a hooked object, the final position of objects isn't compact, but is distributed near the goal boundary.

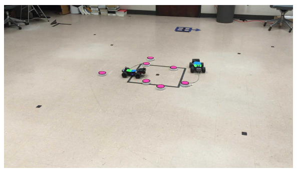
The *pair-only* plan uses the robots as a pair throughout. First, they move a single object at upper-left shown in Figure 9(e). Next, they move a single object at lower-left shown in Figure 9(f). Finally, they move five objects together in Figure 9(h). Figure 9(g) shows the moment when the gathered objects are released. There are the $T = 3$ transitions in the solution. This solution of the *pair-only* plan completes the task in 225 seconds. A snapshot at the final time is in Figure 10(b).

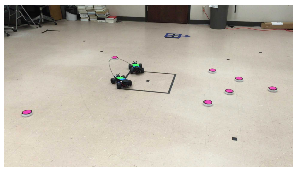
The *both* plan involves the two robots first towing objects individually using a hook primitive, and then connecting their tails to drag multiple objects cooperatively. There are $T = 2$ transitions in the solution. Firstly, two robots move objects at upper-left and lower-left shown in Figure 9(i), and then they release objects at the goal (Figure 9(j)) individually (but simultaneously). Lastly, they use a docking primitive (Figure 9(k)), and drag five

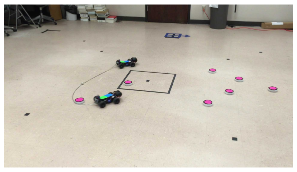

(a) The *single-only* plan, 97 $(s)$: two single tail robots release the hooked objects individually.

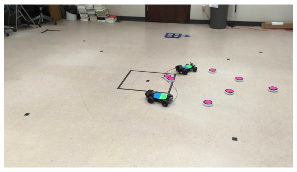

(b) The *single-only* at 134 $(s)$: two single tail robots release the next objects individually.

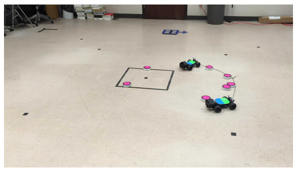

(c) The *single-only* at 238 $(s)$: two single tail robots release the next object.

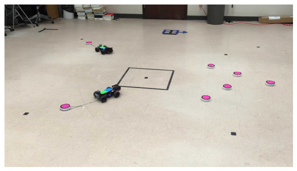

(d) The *single-only* at 280 $(s)$: one single tail robot releases the hooked object alone at the goal.

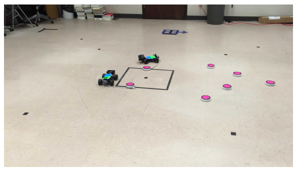

(e) The *pair-only* plan at 70 $(s)$: one pair of robots moves one object (upper-left) to the goal together.

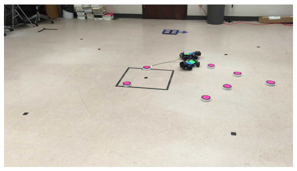

(f) The *pair-only* plan at 110 $(s)$: one pair of robots moves one object (lower-left) to the goal together.


(g) The *pair-only* at 143 $(s)$: this shows the procedure of releasing objects by a pair of robots.


(h) The *pair-only* at 216 $(s)$: one pair of robots move five objects to the goal together.


(i) The *both* plan at 70 $(s)$: two single tail robots move two objects individually.


(j) The *both* plan at 106 $(s)$: two single tail robots release two objects individually using an unhooking action.


(k) The *both* plan at 160 $(s)$: two single tail robots use a docking primitive, and then conjoined via the contact of each tail.

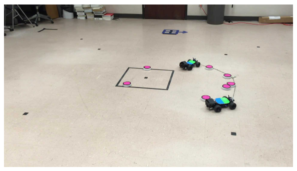

(l) The *both* plan at 210 $(s)$: a conjoined two robots drags five objects together simulatneously.

Figure 9: Comparison of executions of plans, with robots operating individually, as pairs, or both. (The frames here are visible as full video appearing as supplementary material.)

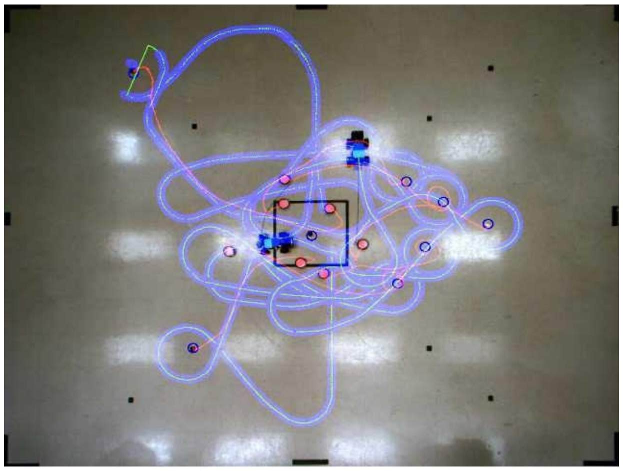
objects to the goal together (Figure 9(l)). This solution of the *both* plan completes the task in 223 seconds. The snapshot at the final time is in Figure 10(c).

A detailed analysis of all the results appears in Figure 11, which shows how the algorithm's expected costs relate to reality; the data suggest that the estimates suffice for providing a preference over paths. The blue bar is the estimated cost (in units of distance traversed) obtained by the proposed algorithm, while the red bar is a real cost (seconds elapsed) based on real test (mean and variance with ten trials). Obviously calibration could relate these two cost metrics—but the data show that ordering between plans is already satisfactory. The data appear to indicate that the *pair-only* and the *both* settings have overall performance that is very similar. In fact, the value of the concurrent execution in the *both* is partly obscured in the graph because the *both* plan must pay the cost of the docking procedure, whereas the *pair-only* plan need not execute the $\textsc{Dock}(\cdot)$ primitive.

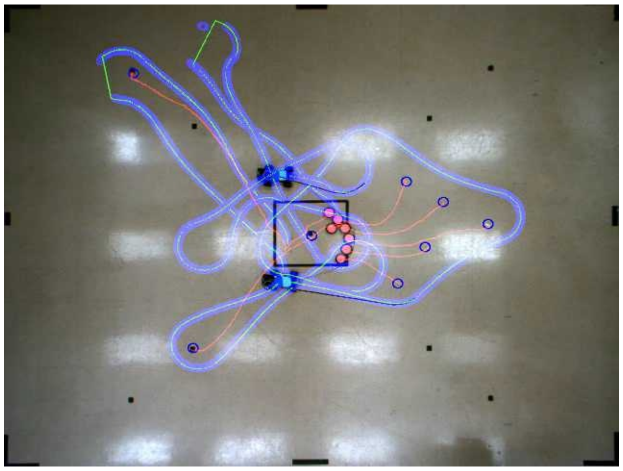
Additionally, the primitives that the pairs of robots use are simpler and more efficient to realize in practice than the hooking and unhooking action. It would appear that the cost of these operations is only justified in fairly extreme problem instances, with a mix of distant objects that can be fetched concurrently by robots acting solo, and also objects efficiently transported by pairs.

## 6.3 Generality

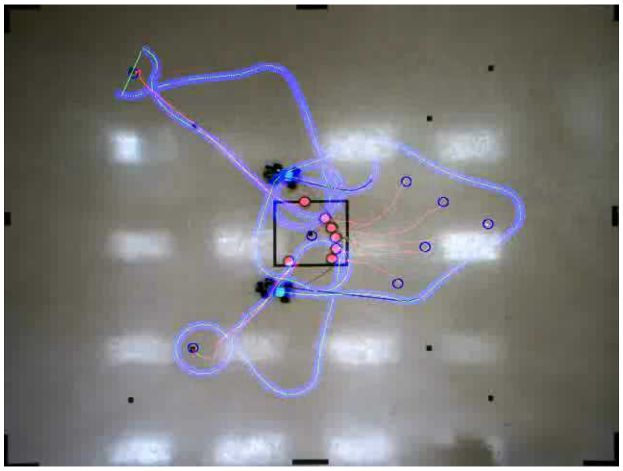
To illustrate the broader applicability of the formulation, including richer goal specifications, we conducted further evaluation of the ONS algorithm. Here, we examine goal specification $\mathbf{F}_2$ for a problem with two robots and six objects (three yellows, three pinks) where $C_1 = 1$, $C_2 = 5$, *mass*$(o_i) = 1$ for all $i$, and $L = 70\ cm$. Modifying the initial locations of the objects, we explored some related problem instances under these settings. The following



(a) *single-only* completes the task at 285 $(s)$.



(b) *pair-only* completes the task at 225 $(s)$.



(c) *both* completes the task at 223 $(s)$.

Figure 10: The view of plans in Figure 9, from the overhead tracking system. The accumulated trajectories for objects and robots are displayed: the blue lines are the robots, while the pink lines are the objects. The printed frames in here are also in full videos appears at the supplementary material.

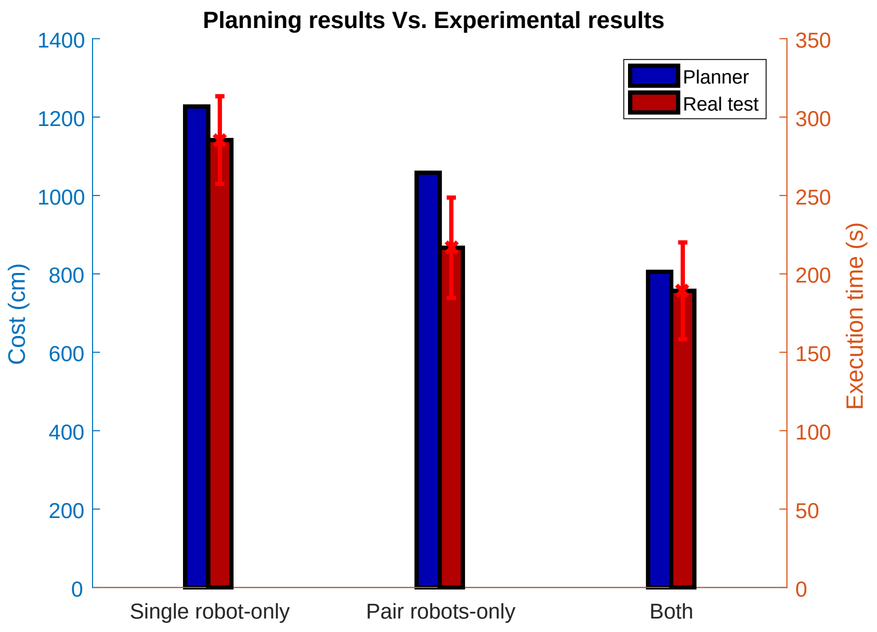


Figure 11: Planner's predicted costs and the results of experiments with physical robots.

two initial configurations are similar but result in drastically different solution classes: (*S1*) The first initial setting is shown in Figure 12(a) (visualized as a graph, akin to those used earlier to describe configurations). (*S2*) The second initial setting is shown in Figure 12(b). The only difference from *S1* is that the yellow $o_5$ is closer to the pink $o_1$, and the pink $o_4$ is a little further from the goal.

For *S1*, the first case, the plan took took 26.4 seconds and examined a total of 517 states. The initial state, in Figure 12(a), transitions to Figures 2(c) and 2(d): two robots transfer $o_1$ and $o_4$ independently and then, acting as a pair, they transfer $o_2$ and $o_3$ together.

Finding a plan for *S2* took 12.5 seconds with 223 states. The graph describing the initial conditions for C2 appears in Figure 12(b), and the plan progresses as follows: a pair of robots transfers $o_1$ and $o_5$ (Figure 12(c)) to the goal, and then they transfer $o_2$ and $o_3$ to the goal (Figure 12(d)). Both results have two transitions ($T = 2$) in their solution. Here, a small change of the positions among objects makes for qualitatively different solutions strategies. As might be expected, several parameters (such as the tail length and $C_2$) can alter the types of solutions found but detailed examples are omitted here.

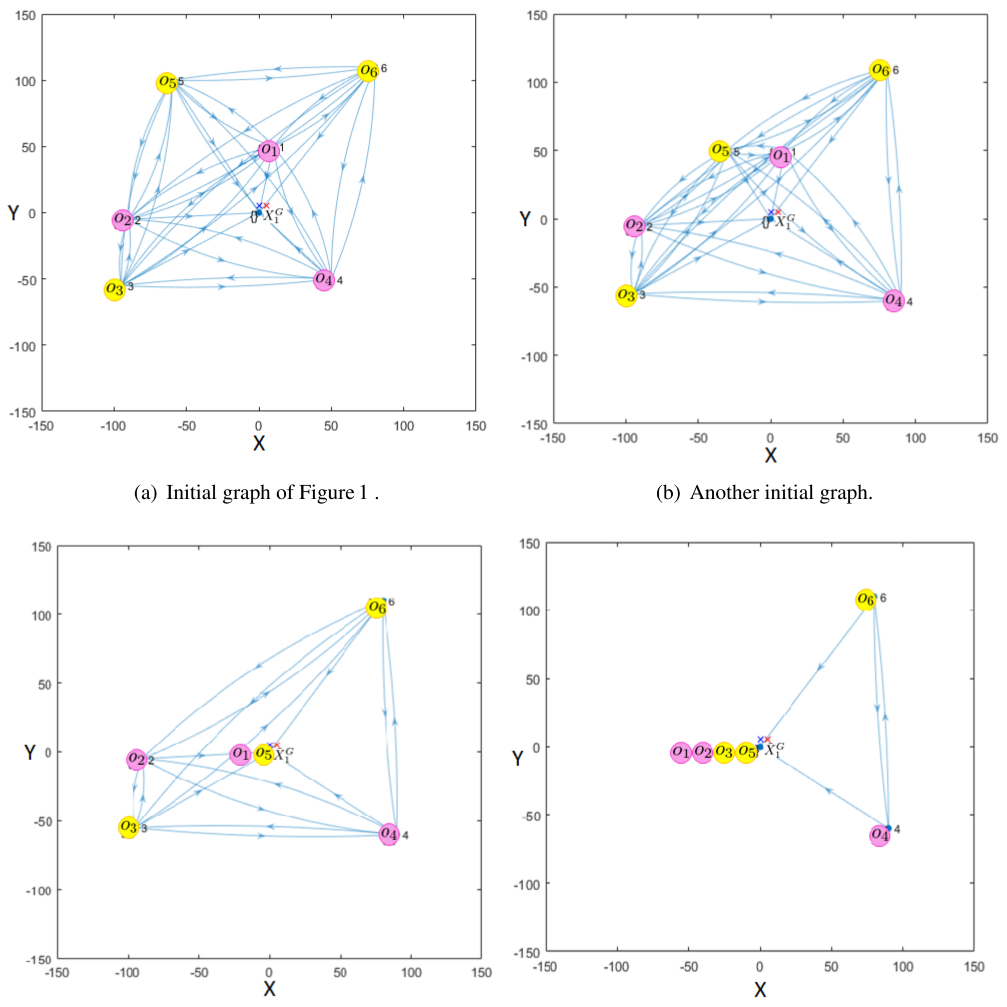
We examined the larger, complex goal predicate $\mathbf{F}_3$. The problem instance has 4 robots and 10 objects, all pink, where $C_1 = 1$, $C_2 = 5$, $mass(o_i) = 1$ for all $i$, and $L = 50\ cm$. Again the ONS algorithm was used. Our planning time for $\mathbf{F}_3$ took 1353 seconds, exploring 4469 states. The initial configuration is shown graphically in Figure 13(a), and the resulting plan appears in Figures 13(b) and 13(c). First, two single robots transfer $o_2$ and $o_5$ to $X_1^G$ independently, while one pair of robots transfer $o_3$ and $o_9$ to $X_3^G$ together. Second, one pair of robots transfers $o_6$ to $X_2^G$, while another pair of robots transfers $o_7$ to $X_3^G$. Then the final result satisfies $\mathbf{F}_3$. Here the result shows two transitions ($T = 2$) in the solution.

## 7 Discussion

It is useful to provide some additional detail on some aspects of the approach, the experiments, and the steps needed to mature the work so as to apply it outside of laboratory settings.

### 7.1 Obstacles and feasible paths

Because the preceding presentation has focused on task-planning aspects, little has been said so far about finding collision-free paths that are executed by the robots and the treatment of obstacles. In the figures, graphs have been used to give a visual representation of the world configurations in a way that helps give an intuition behind



(a) Initial graph of Figure 1 .

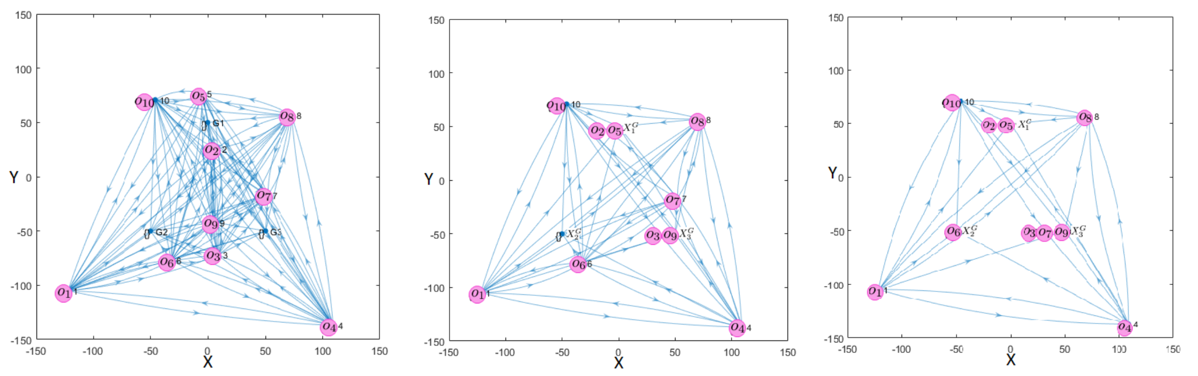
(b) Another initial graph.

(c) First step: from (b) to (c), one pair of robots transfers $o_1$ and $o_5$ to the goal $X_1^G$.

(d) Second step: from (c) to (d), one pair of robots transfers $o_2$ and $o_3$ to the goal $X_1^G$. This satisfies $\mathbf{F}_2$.

Figure 12: The simulation results are plotted using graph representation. There are three pink and three yellow objects, two robots, and one goal; (a) is for the example in Figure 1(a). Another example is shown in (b), which modified the topological relationship of (a). We show the results of the planner for the example (b) via (c) and (d). Finally, (d) satisfies $\mathbf{F}_2$.

the combinatorics of the search problem. Edges in those graphs express the possibility of some action which representations a transition from one site $X_i$ to another $X_j$. Before finding the task plan, the initial set of edges and their weights are precomputed. Then, whilst planning, the examination of possible choices requires that we perform some additional checks for the satisfaction of additional geometric constraints to ensure the resulting path is feasible. Some of the particular primitives that the robots execute need a working boundary (e.g., to move past and surround objects, space for docking, and splitting two robots). Our prior work [Kim and Shell, 2017] has detailed explanations of the types of motion primitives we use. Basically, each primitive consists of an initialization motion, a main motion, and a termination motion (with the first and third possibly being empty motions). Each is a sequence of controls generated via a Dubins curve, which transitions from one to the next. The three phases each impose constraints that are represented geometrically and which must all be satisfied if the primitive is to succeed. The procedure needed is simple: if the robot's working boundary intersects objects or other static obstacles in the configuration space, then we remove the edge representing the action from $X_i$ to $X_j$ by $\mathbf{r}$. It is worth noting that objects located very close to static obstacles may effectively be immobile because the motions needed in first phase of a primitive requires some space between objects and obstacles (e.g., sufficient room to move the tail to execute



(a) There are three goals, $X_1^G$, $X_2^G$, $X_3^G$ We use four robots and ten objects. Initially, all robots are located at $X_1^G$.

(b) First step: from (a) to (b), two single robots transfer $o_2$ and $o_5$ to $X_1^G$ individually, while one pair of robots transfers $o_3$ and $o_9$ to $X_3^G$ together.

(c) Second step: from (b) to (c), one pair of robots transfers $o_6$ to $X_2^G$, while another pair of robots transfers $o_7$ to $X_3^G$. This satisfies $\mathbf{F}_3$.

Figure 13: The simulation results of a graph representation; (a) shows ten pink objects and three goals. We used 4 robots located at $X_1^G$ initially. We show the results of the planner for the example (a) via (b) and (c). Finally (c) satisfies $\mathbf{F}_3$.

a hooking action). A consequence is that the feasibility check may need to ensure that paths are collision-free, not only for robots, but also for some portion of the length of the tail that can actually manipulate the object(s).

## 7.2 Limitations of physical robot experiments

One issue we encountered was that it was difficult to ensure that the robot maintained a fixed velocity. This is because the dragging force varies depending on whether the objects are moving (static vs. sliding friction) or not, and how many objects are moving together. Some alternative manipulation actions (e.g., the statics-based motion primitives described in our previous work [Kim and Shell, 2017]) can help improve precision of the manipulation, but, given that the focus of the work was cooperation and task planning, we did not expend a great deal of time on tuning the manipulation actions.

A second issue we encountered in practice had to do with the inexactness of the estimates of object positions. In general, the robot could wrap objects by giving a generous gap around the hull describing the (perceived) boundary of the objects. But we found it is challenging to get the robot to release objects without touching them. However, the intention is to move objects by towing only. Higher precision, thus, appears to be important for releasing objects if one wishes to ensure direct robot–object contact never occurs.

## 7.3 Considerations for the design and modeling of a practical tail for manipulation

The planning problem, as formulated in Section 3.2, abstracts away most of the details of the tail itself. The length of the tail and its relation to the size of the objects, along with towing force that the robot is able to exert, are expressed simply as constraints expressed as predicates $C_1$ and $C_2$ in (3). The roboticist simultaneously faces the question of how to design the physical tail, and how to model it effectively for planning purposes. More detailed consideration is thus warranted. The present discussion seeks to provide a useful starting point.

As the robot attempts to move a stationary object, there are three forces at play: the force produced by the robot $F_r$, the tail's static friction $F_t$, and the object's static friction $F_o$. Consider two coefficients of friction[1]: let $\mu_t$ be the coefficient of friction between a tail and the workspace floor, and let $\mu_o$ be the coefficient of friction between an object and the workspace floor. Let $L^i$ represent the tail length for a robot $i$ and $m_t$ be the mass of the tail per unit length. Then, the following condition must be satisfied if the robot is to move the object:

$$F_r \geq F_t + F_o, \tag{13}$$

where $F_t = \mu_t \cdot m_t \cdot L^i \cdot g$ and $F_o = \mu_o \cdot M_o \cdot g$.

[1]For sake of conciseness, we adopt the terminology and notation of Kim and Shell [2017].

The predicate $C_1$ also depends on an encircled object's perimeter $d_o$. Both the hooking and releasing actions need the extra length $d_r$ to surround and release the secured object owing to the extent of the robot (*cf.* Figure 6). Then, to drag objects with a single robot, the following second condition for the tail length $L^i$ must be satisfied:

$$L^i \geq d_o + d_r. \tag{14}$$

The same reasoning holds for pairs of conjoined robots and the equations are analogous: predicate $C_2$ depends on (1) the force induced by two robots, (2) the perimeter of the clustered object boundaries, (3) the total length of two conjoined tails. Then, (13) can be modified by using an effective $F_r$ for two robots and $F_t$ for two tails. Also, (14) should now have $d_o$ as the perimeter of the convex hull of the clustered objects, and $d_r$ can be extended for two robots correspondingly. Note that (14) gives a lower bound on $L^i$ owing to the relationship between tail length, mass, and forces produced robots. However, (13) gives an upper bound which depends on the object sizes and tail length. In our experience, the tail should be substantially shorter than the bound in (13). If the robots have tails of excessive length, the steps to initialize hooking, releasing, and docking actions become infeasible as obstacles will hinder motion greatly. In such cases, edges connecting locations (such as that from $X_i$ to $X_j$) are removed by the planner, and the problem becomes infeasible.

Though we have not made any statement about tail stiffness thus far, it can be treated as a curvature constraint (see Teshnizi and Shell [2016]). Let $\kappa$ be the non-zero radius of the smallest circle into which it can be bent. Curvature plays a role because (1) one robot must wrap objects for hooking; (2) a pair of robots, at most, come back together when dragging. This requires tails to have length at least $2\pi\kappa$, thus, the following third condition must be satisfied:

$$L^i \geq 2\pi\kappa. \tag{15}$$

Finally, one may model heterogeneous robot capabilities via $C_1$ and $C_2$ as well. If one has robots with different dragging forces, robots $r_i$ and $r_j$, say, where $r_i$'s dragging force is greater than $r_j$'s. A straightforward approach is to assume that robot $r_i$ reduces its the dragging force to tow objects in concert with $r_j$, and the robot's do not exploit tail-object friction. With different $\kappa$ values for the two robots, a sufficient condition is that we take two $\kappa$ values, where each is proportional to the length of its tail, respectively. Finally, we can compute the minimum length of tails to make a circle with different $\kappa$ values. Thus (15) becomes

$$L^i \geq 2\pi(\omega_i\kappa_i + \omega_j\kappa_j), \tag{16}$$

where $\omega_i$ is a tail length ratio for $r_i$ to the overall tail length and $\omega_j$ is defined analogously ($\omega_i + \omega_j = 1$).

# 8 Conclusion

This paper considers the problem of moving multiple objects to given goal locations with a coordinated team of mobile robots. Building on our prior work in which we developed a robot capable of manipulating objects with a rope-like structure affixed as a tail [Kim and Shell, 2017], here we explore how multiple robots of this type can cooperatively transport objects. To do this, robots can act as solo agents by wrapping their tail around items, securing the end, and towing. Or, additionally, pairs of robots can join their tails and act as a single unit in sweeping out an area. We pose and study the planning problem in which robots, during their execution, can dynamically choose to make or break pairs autonomously.

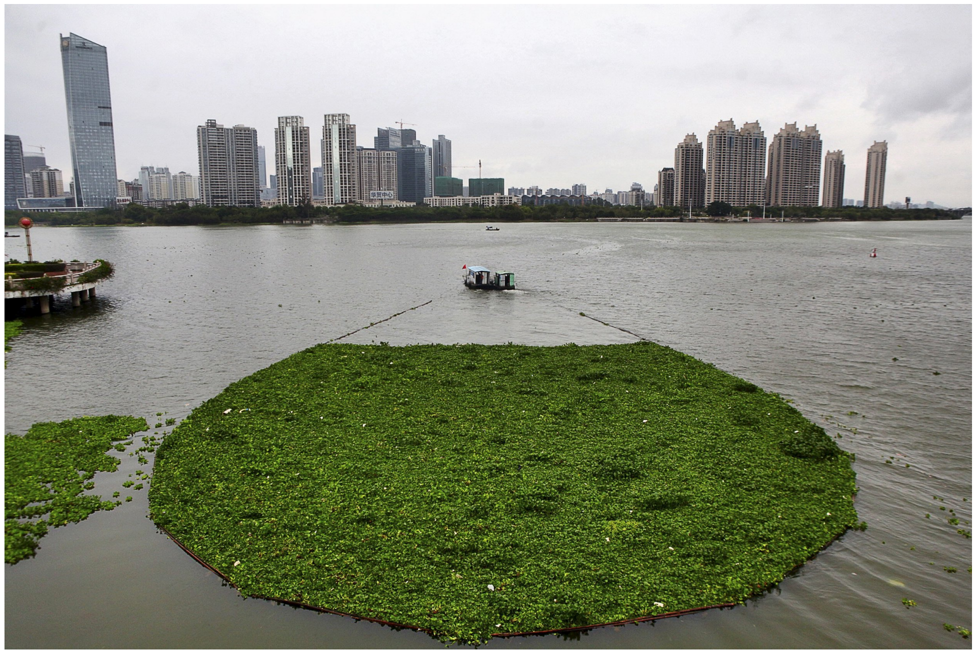
This paper describes a planner which, ultimately, outputs paths that multiple robots execute in order to encircle, move, and release objects so that they reach some final arrangement. The manipulation goal itself is specified in terms of logical predicates, enabling rich tasks to be described. We proved the MOCCT problem to be NP-hard even when it is cheap to determine whether a satisfying final arrangement exists. Next, we compared two practical algorithms for the problem, illustrating the value of a heuristic which clusters objects opportunistically. We describe the first physical demonstration of multiple robots solving a manipulation problem in this way.

## 8.1 Future work and outlook

Our research was originally motivated by aquatic applications: tethers and cables are particularly useful for manipulating objects floating on water. As well-illustrated in Figure 14, both single and pairs of human piloted boats have been used for water surface clean up operations. The importance of clean freshwater for our survival suggests

(a) Oil skimming vessels. (origin: Davidson [2010], CC BY-NC-SA 2.0)

(b) Plastic pollution in the oceans. (origin: Zak Noyle/AFrame [Noyle, 2014])



(c) Two boats, connected by a flexible cable, drag water hyacinth. (origin: Reuters [2015])

Figure 14: Tethered robots can skim oil, clean up garbage on the surface

that it is worth examining how autonomous surface vehicles could best be deployed for these sorts of applications. Moreover, given the vast physical extent of many bodies of water, we believe that surface vehicles —operating sometimes individually and sometimes as coupled pairs, as the occasion demands— would be of value.

The present work abstracts away the low-level details of the primitive motions, and also purposefully avoids detailed use of frictional and other physical models, because we intend for the techniques to carry over to vehicles operating on the surface of water. The degree to which this is possible remains to be seen.

## Acknowledgements

This research was funded, in part, by the National Science Foundation under grants IIS-1302393 and IIS-1453652. The authors are grateful for this support.

## References

Federico Augugliaro, Emanuele Zarfati, Ammar Mirjan, and Raffaello D'Andrea. Knot-tying with flying machines for aerial construction. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5917–5922, Hamburg, Germany, September 2015.

Subhrajit Bhattacharya, Hordur Heidarsson, Gaurav S. Sukhatme, and Vijay Kumar. Cooperative Control of Autonomous Surface Vehicles for Oil Skimming and Cleanup. In *Proceedings of the International Conference on Robotics and Automation*, pages 2374–2379, Shanghai, China, May 2011.

Subhrajit Bhattacharya, Soonkyum Kim, Hordur Heidarsson, Gaurav S. Sukhatme, and Vijay Kumar. A topological approach to using cables to separate and manipulate sets of objects. *The International Journal of Robotics Research*, 34(6):799–815, 2015.

Thomas Bouton, Diego Caminha B. de Oliveira, David Déharbe, and Pascal Fontaine. veriT: An Open, Trustable and Efficient SMT-Solver. In Renate A. Schmidt, editor, *Automated Deduction – CADE-22: 22nd International Conference on Automated Deduction, Montreal, Canada, August 2-7, 2009. Proceedings*, pages 151–156. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

Randall Briggs, Jongwoo Lee, Matt Haberland, and Sangbae Kim. Tails in Biomimetic Design: Analysis, Simulation, and Experiment. In *Proceedings of the International Conference on Intelligent Robots and Systems*, Vilamoura, Algarve, Portugal, October 2012.

Evan Chang-Siu, Thomas Libby, Masayoshi Tomizuka, and Robert J. Full. A lizard-Inspired Active Tail Enables Rapid Maneuvers and Dynamic Stabilization in a Terretrial Robot. In *Proceedings of the International Conference on Intelligent Robots and Systems*, San Fransisco, CA, USA, September 2011.

Peng Cheng, Jonathan Fink, Vijay Kumar, and Jong-Shi Pang. Cooperative Towing With Multiple Robots. *Journal of Mechanisms and Robotics*, 1(1), 2008.

Neil T. Dantam, Zachary K. Kingston, Swarat Chaudhuri, and Lydia E. Kavraki. Incremental Task and Motion Planning: A Constraint-Based Approach. In *Proceedings of Robotics: Science and Systems Conference*, Michigan, USA, June 2016.

George B. Dantzig and John H. Ramser. The truck dispatching problem. *Management Science*, 6(1):80–91, 1959.

James Duncan Davidson. Boats skim oil spilled by BP from the surface of the Gulf of Mexico., 2010. URL `http://www.ens-newswire.com/ens/aug2010/2010-08-06-01.html`.

Leonardo De Moura and Nikolaj Bjørner. Z3: An efficient SMT solver. In *Proceedings of International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, pages 337–340, Budapest, Hungary, March 2008.

Leonardo De Moura and Nikolaj Bjørner. Satisfiability modulo theories: Introduction and applications. *Commun. ACM*, 54(9):69–77, September 2011.

Bruce Donald, Larry Gariepy, and Daniela Rus. Distributed Manipulation of Multiple Objects using Ropes. In *Proceedings of the International Conference on Robotics and Automation*, pages 450–457, San Francisco, CA, USA, April 2000.

Jonathan Fink, Nathan Michael, and Vijay Kumar. Composition of Vector Fields for Multi-Robot Manipulation via Caging. In *Proceedings of Robotics: Science and Systems Conference*, Atlanta, Georgia, USA, July 2007.

Jonathan Fink, M. Ani Hsieh, and Vijay Kumar. Multi-Robot Manipulation via Caging in Envirionments with Obstacles. In *Proceedings of the International Conference on Robotics and Automation*, pages 1471–1476, Pasadena, CA, USA, May 2008.

Jonathan Fink, Nathan Michael, Soonkyum Kim, and Vijay Kumar. Planning and control for cooperative manipulation and transportation with aerial robots. *The International Journal of Robotics Research*, 30(3):321–328, 2011.

Paolo Robuffo Giordano and Marilena Vendittelli. Shortest Paths to Obstacles for a Polygonal Dubins Car. *IEEE Transactions on Robotics*, 25(5):1184–1191, 2009.

Susan Hert and Vladimir Lumelsky. The Ties that Bind: Motion Planning for Multiple Tethered Robots. In *Proceedings of International Conference on Robotics and Automation (ICRA)*, 1994.

Lucas Hill, Thomas Woodward, Hitoshi Arisumi, and Ross L. Hatton. Wrapping a target with a tethered projectile. In *Proceedings of International Conference on Robotics and Automation (ICRA)*, pages 1442–1447, Seattle, WA, USA, May 2015.

William N. N. Hung, Xiaoyu Song, Jindong Tan, Xiaojuan Li, Jie Zhang, Rui Wang, and Peng Gao. Motion planning with satisfiability modulo theories. In *Proceedings of International Conference on Robotics and Automation (ICRA)*, pages 113–118, May 2014.

T. Huntsberger, P. Pirjanian, A. Trebi-Ollennu, H. Das Nayar, H. Aghazarian, A. J. Ganino, M. Garrett, S. S. Joshi, and P. S. Schenker. Campout: a control architecture for tightly coupled coordination of multirobot systems for planetary surface exploration. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 33(5):550–559, Sept 2003.

Takeo Igarashi and Mike Stilman. Homotopic Path Planning on Manifolds for Cabled Mobile Robots. In *Proceedings of International Workshop on the Algorithmic Foundations of Robotics*, Singapore, December 2010.

Soonkyum Kim, Subhrajit Bhattacharya, and Vijay Kumar. Path Planning for a Tethered Mobile Robot. In *Proceedings of the International Conference on Robotics and Automation*, Hong Kong, China, May 2014.

Young-Ho Kim and Dylan A. Shell. Using a compliant, unactuated tail to manipulate objects. *IEEE Robotics and Automation Letters*, 2(1):223–230, January 2017.

Young-Ho Kim, Sang-Wook Lee, Hyun S. Yang, and Dylan A. Shell. Toward autonomous robotic containment booms: visual servoing for robust inter-vehicle docking of surface vehicles. *Intelligent Service Robotics*, 5(1): 1–18, 2012.

C.Ronald Kube and Eric Bonabeau. Cooperative transport by ants and robots. *Robotics and Autonomous Systems*, 30(1–2):85 – 101, 2000.

Harold Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistic Quarterly*, 2(1-2): 83–97, 1955.

Steven M. LaValle. *Planning Algorithms*. Cambridge University Press, New York, NY, USA, 2006.

Jan K. Lenstra and Alexander H. G. Rinnooy Kan. Complexity of vehicle routing and scheduling problems. *Networks*, 11(2):221–227, June 1981.

Thomas Libby, Aaron M. Johnson, Evan Chang-Siu, Robert J. Full, and Daniel E. Koditschek. Comparative Design, Scaling, and Control of Appendages for Inertial Reorientation. *IEEE Transactions on Robotics*, 32(6): 1380–1398, Dec 2016.

Ryan Luna and Kostas E. Bekris. Efficient and complete centralized multi-robot path planning. In *Proceedings of International Conference on Intelligent Robots and Systems (IROS)*, pages 3268–3275, San Francisco, California, USA, Sept 2011.

Neil Mathew, Stephen L. Smith, and Steven L. Waslander. Multirobot Rendezvous Planning for Recharging in Persistent Tasks. *IEEE Transactions on Robotics*, 31(1):128–142, 2015a.

Neil Mathew, Stephen L. Smith, and Steven L. Waslander. Planning paths for package delivery in heterogeneous multirobot teams. *IEEE Transactions on Automation Science and Engineering*, 12(4):1298–1308, Oct 2015b.

Patrick McGarey, Kirk MacTavish, François Pomerleau, and Timothy D. Barfoot. The line leading the blind: Towards nonvisual localization and mapping for tethered mobile robots. In *Proceedings of International Conference on Robotics and Automation*, pages 4799–4806, Stockholm, Sweden, May 2016.

Srinivas Nedunuri, Sailesh Prabhu, Mark Moll, Swarat Chaudhuri, and Lydia E. Kavraki. SMT-based synthesis of integrated task and motion plans from plan outlines. In *Proceedings of International Conference on Robotics and Automation (ICRA)*, pages 655–662, May 2014.

Zak Noyle. No, its NOT Photoshopped, 2014. URL `https://www.hometowndumpsterrental.com/blog/no-its-not-photoshopped`.

Ted K. Ralphs, Leonid Kopman, William R. Pulleyblank, and Leslie E. Trotter. On the capacitated vehicle routing problem. *Mathematical Programming*, 94(2):343–359, 2003.

Reuters. Polluted Waters of China, 2015. URL `http://www.reuters.com/news/picture/polluted-waters-of-china?articleId=USRTR4WTPW`.

Se-Gon Roh, Jae Hoon Park, Young Kouk Song, KwangWoong Yang, Moosung Choi, Hong-Seok Kim, Hogil Lee, and Hyouk Ryeol Choi. Flexible Docking Mechanism Using Combination of Magnetic Force with Error-Compensation Capability. In *Proceedings of IEEE Conference on Automation Science and Engineering*, Washington DC, USA, August 2008.

William S. Rone and Pinhas Ben-Tzvi. Continuum Robotic Tail Loading Analysis for Mobile Robot Stabilization and Maneuvering. In *Proceedings of International Design Engineering Technical Conferences & Computers and Information in Engineering Conference*, Buffalo, New York, USA, August 2014.

Daniela Rus, Bruce Donald, and Jim Jennings. Moving furniture with teams of autonomous robots. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 1, pages 235–242, Aug 1995.

Mitul Saha and Pekka Isto. Motion Planning for Robotic Manipulation of Deformable Linear Objects. In *Proceedings of the International Conference on Robotics and Automation*, Orlando, USA, May 2006.

Frank W. Sinden. The Tethered Robot Problem. *International Journal of Robotics Research*, 9(1):122–133, 2000.

Kiril Solovey and Dan Halperin. k-color multi-robot motion planning. *The International Journal of Robotics Research*, 33(1):82–97, 2014.

Reza H. Teshnizi and Dylan A. Shell. Planning motions for a planar robot attached to a stiff tether. In *Proceedings of International Conference on Robotics and Automation*, pages 2759–2766, Stockholm, Sweden, May 2016.

Matthew Turpin, Kartik Mohta, Nathan Michael, and Vijay Kumar. Goal assignment and trajectory planning for large teams of interchangeable robots. *Autonomous Robots*, 37(4):401–415, 2014.

Jur van den Berg, Jack Snoeyink, Ming Lin, and Dinesh Manocha. Centralized Path Planning for Multiple Robots: Optimal Decoupling into Sequential Plans. In *Proceedings of Robotics: Science and Systems*, Seattle, USA, 2009.

William Vega-Brown and Nicholas Roy. Asymptotically optimal planning under piecewise-analytic constraints. In *Proceedings of International Workshop on the Algorithmic Foundations of Robotics*, San Fransisco, December 2016.

Glenn Wagner and Howie Choset. Subdimensional expansion for multirobot path planning. *Artificial Intelligence*, 219:1–24, 2015.

Weifu Wang and Devin Balkcom. Tying knots precisely. In *Proceedings of International Conference on Robotics and Automation (ICRA)*, 2016.

Sean Wilson, Theodore P. Pavlic, Ganesh P. Kumar, Aurélie Buffin, Stephen C. Pratt, and Spring Berman. Design of ant-inspired stochastic control policies for collective transport by robotic swarms. *Swarm Intelligence*, 8(4): 303–327, 2014.

Atsushi Yamashita, Jun Sasaki, Jun Ota, and Tamio Arai. Cooperative manipulation of objects by multiple mobile robots with tools. In *Proceedings of the 4th Japan-France/2nd Asia-Europe Congress on Mechatronics*, pages 310–315, Fukuoka, Japan, 1998.