Unmanned Systems, Vol. 0, No. 0 (2016) 1–17
© World Scientific Publishing Company

Heterogeneous Task Allocation and Sequencing via Decentralized Large Neighborhood Search

Armin Sadeghi[a,*] , Stephen L. Smith[a]

[a]*Department of Electrical and Computer Engineering, University of Waterloo, Waterloo ON, N2L 3G1 Canada*

This paper focuses on decentralized task allocation and sequencing for multiple heterogeneous robots. Each task is defined as visiting a point in a subset of the robot configuration space – this definition captures a variety of tasks including inspection and servicing. The robots are heterogeneous in that they may be subject to different differential motion constraints. Our approach is to transform the problem into a multi-vehicle generalized traveling salesman problem (GTSP). To solve the GTSP, we propose a novel decentralized implementation of large-neighborhood search (LNS). Our solution approach leverages the GTSP insertion methods proposed in [1] to repeatedly remove and reinsert tasks from each robot path. Decentralization is achieved using combinatorial-auctions between the robots on tasks removed from robot's path. We provide bounds on the length of the dynamically feasible robot paths produced by the insertion methods. We also show that the number of bids in each combinatorial auction, a crucial factor in the runtime, scales linearly with the number of tasks. Finally, we present extensive benchmarking results to characterize both solution quality and runtime, which show improvements over existing decentralized task allocation methods.

Keywords: autonomous robots; non-holonomic robots; decentralized control.

1. Introduction

Task allocation and sequencing is a fundamental component of multi-robot operation and has been studied extensively [2, 3]. The problem consists of finding an assignment between tasks and robots along with an ordering of the tasks assigned to each robot. The objective is typically to minimize the average time, maximum time, or energy consumption of performing all tasks. A wide variety of task types and robot models have been considered in the literature, and are reviewed in [2, 3]. Our focus is on tasks that require a robot to visit a location in the workspace. In this area, the literature can be divided based on 1) single or multiple robot, 2) centralized or decentralized, and 3) vehicle dynamics or simple motion.

For a single robot the problem is simply one of task sequencing. If the robot does not have dynamics, then computing an ordering of task locations is a traveling salesman problem (TSP), for which very successful heuristic and approximation algorithms exist [4, 5]. A simple class of TSP algorithms are insertion heuristics [6], which operate by repeatedly inserting a new vertex into a partial tour. For metric graphs, two such heuristics, nearest and cheapest insertion, provide 2-factor approximations to the optimal tour [6].

For a single robot with dynamics, the Dubins vehicle model in which vehicle paths have bounded curvature is commonly studied. Early papers on the Dubins TSP include [7, 8]. In [9], a method for solving the Dubins TSP was proposed based on conversion to the generalized traveling salesman problem (GTSP). In the generalized traveling salesman problem, the cities are partitioned into disjoint sets, and he goal is to find a tour that visits one city in each set. The GTSP can be solved via a reduction to the TSP [10], or directly using GTSP solvers [11–13]. A similar conversion to the GTSP was proposed for planning tours for a robotic arm in [14], for a single Dubins TSP with neighborhoods [15] and for high-level task sequencing problems [16]. Sequencing problems have also been considered for differential drive and Reeds-Shepp models [17].

For multiple robots without dynamics, the centralized problem can be posed as a multi-vehicle TSP. In [18], a reduction is given from the multi-vehicle TSP to the single vehicle TSP for the min-sum objective. An approximation algorithm has also been recently developed for the objective of minimizing the maximum path length among vehicles [19].

The problem of computing multiple shortest tours

*Email addresses: *a6sadegh@uwaterloo.ca* *(Armin Sadeghi)*, *stephen.smith@uwaterloo.ca* *(Stephen L. Smith)*.

through a set of disk neighborhoods in the plane was studied in [20]. In [15], the more general problem of computing multiple shortest tours through arbitrary neighborhoods and with Dubins' vehicle dynamics was considered. The approach proposed for general neighborhoods was to discretize each neighborhood, and then convert the problem to an instance of the GTSP. In this paper we utilize a similar approach to discretizing neighborhoods, but we develop a decentralized algorithm that utilizes both auctions between robots, and local optimization of individual robot routes.

The decentralized problem for multiple robots without dynamics is commonly solved using market-based auctions [21–23]. When there are an equal number of robots and tasks, the problem is commonly referred to as the assignment problem, which can be solved efficiently [24, 25]. When there are more tasks than robots, there are two main auction-based approaches: 1) allocating a single task per auction using independent bids on each task [23, 26], or 2) bidding on subsets of tasks such that all tasks are allocated in a single auction, known as combinatorial auctions [27,28]. In auction algorithms with bids on individual tasks, a bid is independent of the other tasks in the auction. Therefore, the winner determination problem in these auctions is easy to address. However, typically, the cost for a robot to complete a task depends on the other tasks in its route. By placing bids on subsets of tasks, robots can capture the true cost of a subset of tasks, which may differ from the sum of bids on the individual tasks [29].

The advantage of bidding on subsets of tasks is that all tasks are allocated in a single auction, while the main drawback is the additional computational complexity [21], since the number of subsets grows exponentially with the number of tasks. Moreover, winner determination in the combinatorial auction is in general an NP-hard problem [30]. Therefore, there are several techniques to limit the number of subsets [27] and heuristics to approximate the winner determination problem [31]. A successful auction-based approach that bids on subsets for allocation and sequencing is the consensus-based bundle algorithm (CBBA) [21], which uses consensus algorithms to spread bids between robots. Each robot generates a single subset of tasks and bid on the tasks in the subset.

For several task allocation algorithms [21, 29], there are constant factor approximation guarantees that provide bounds on the worst-case performance. However, the performance of these algorithms are typically significantly better than their bounds in practice. In fact, a constant factor approximation can be achieved using very simple auction algorithms [29]. Therefore, in this paper we not only seek to develop an auction algorithm that ensures a constant factor worst-case performance guarantee, but also propose a powerful local optimization technique that further improve the quality of each robot tour. Our approach utilizes an optimization framework called large neighborhood search (LNS) [32], which has been successfully applied to several vehicle routing problems [33, 34] and the GTSP [13]. The high-level idea is to begin with a candidate solution and then repeatedly perform destroy and repair procedures. If the cost of the new solution satisfies an acceptance criterion, then it is accepted and the procedure is repeated.

In this paper, we focus on the decentralized task allocation and sequencing for heterogeneous robots with differential motion constraints. The CAPT algorithm [35] provides a solution when the number of tasks equals the number of robots, and thus sequencing is not required. Existing studies on the decentralized task allocation for multiple robots with dynamics constraints propose decoupling the dynamic constraints from the task allocation problem [36,37]. These algorithms consist of a task allocation phase in which robot dynamics are not considered. The allocation phase determines the tasks allocated to each robot and the sequence of performing the tasks. Since the resulting sequences may not be feasible under the dynamic constraints of the robots, the allocation phase is followed by a trajectory planning phase which converts the sequences to feasible tours. Although the decoupling of the task allocation and trajectory planning reduces the complexity, the resulting paths may suffer in the quality [38].

A key contribution of our decentralized auction-based approach is that it performs task allocation and trajectory planning concurrently, rather than decoupling the two stages. Tasks are defined as subsets of the robot configuration space, and a robot completes a task by visiting any point in the subset. Building on prior work [9,14], we transform the problem into a multi-vehicle GTSP. We leverage insertion methods for the GTSP in [1] as a bidding mechanism in the auctions. Moreover, we utilize the GTSP insertion methods to generate multiple tours for robots with dynamics, and we provide bounds on the performance of the insertion methods. In addition to the performance bound, the extensive set of experiments on the homogeneous and heterogeneous system of robots show improvement over the existing methods with decoupling approaches.

The paper is organized as follows. In Section 2, we provide background on combinatorial problems and vehicle models. In Section 3, we formulate the multi-robot task allocation problem and its relation to the GTSP. Section 4 presents insertion methods for the GTSP, along with bounds on their performance. In Section 6, we present an auction-based task allocation algorithm and in Section 7, we provide benchmarking results.

2. Preliminaries

In this section we give background on the TSP, GTSP, GTSP insertion methods and combinatorial auction problems, as well as standard vehicle models.

2.1. *Combinatorial problems*

A graph $G = (V, E, c)$ consists of a set of vertices V, a set of edges E and edge costs $c : E \to \mathbb{R}_{>0}$. A path P in G is a non-repeating sequence of vertices connected by edges. A cycle or tour T in G is a path in which the first and last vertices are equal. We can think of a path or tour as a connected subgraph of G denoted $P = (V_P, E_P)$, where $V_P \subseteq V$ and $E_P \subseteq \{(u, v) \in E \mid u, v \in V_P\}$.[39]

The traveling salesman problem (TSP): Given a complete graph $G = (V, E, c)$, the TSP is the problem of finding a tour $T = (V, E_T)$ that minimizes the sum of the edge costs on the tour $\sum_{e \in E_T} c(e)$.

The generalized traveling salesman problem (GTSP): Given a complete graph $G = (V, E, c)$ along with a partition of its vertex set into m mutually disjoint subsets $(V_1, V_2, \ldots, V_m)$, the GTSP is the problem of finding a tour $T = (V_T, E_T)$ that includes exactly one vertex from each subset V_i (i.e., $|V_T \cap V_i| = 1$ for each $i \in \{1, \ldots, m\}$) and minimizes $\sum_{e \in E_T} c(e)$.

The multi-vehicle GTSP (mGTSP): Given a complete graph $G = (V, E, c)$ with vertex partition $(V_1, V_2, \ldots, V_m)$ and a number of vehicles N_v, the mGTSP, is the problem of finding N_v tours that collectively visit each vertex set exactly once and with minimum total length. More precisely, the goal is to find tours $T^j = (V_{T^j}, E_{T^j})$, $j \in \{1, \ldots, N_v\}$ such that

(1) $V_{T^j} \cap V_{T^k} = \emptyset$ for each $j, k \in \{1, \ldots, N_v\}$;
(2) $|\cup_{j=1}^{N_v} V_{T^j} \cap V_i| = 1$ for each vertex set V_i; and
(3) $\sum_{j=1}^{N_v} \sum_{e \in E_{T^j}} c(e)$ is minimized.

The definition of the mGTSP also captures the multi-traveling salesman problem (**mTSP**) where each vertex set V_i consists of a single vertex.

Combinatorial auction problem: Given a set $A = \{a_1, \ldots, a_m\}$, a set of subsets $\{S_1, \ldots, S_n\}$ where each $S_i \subset A$, and non-negative price $p_i > 0$ for each subset S_i, the combinatorial auction problem is to find a set of subsets W that forms a partition of A and that maximizes the total value $\sum_{i|S_i \in W} p_i$.

In the task allocation problem, the elements of A are the tasks and a subset S_i is a subset of tasks. A bid p_i on a subset, offered by a robot, is the reward of accomplishing the tasks in the subset. The objective is to allocate a subset of tasks to each robot such that the robots collectively perform all the tasks and the revenue is maximized.

2.2. *Insertion Methods*

A class of insertion methods for constructing GTSP tours are presented in [1]. The insertion methods are extensions of the class of insertion methods defined for constructing TSP tours in [6]. These extensions are as follows.

Consider a GTSP graph $G = (V, E, c)$, a sub-tour[a] $T = (V_T, E_T)$, a vertex set V_i such that $V_i \cap V_T = \emptyset$, and a vertex $v \in V_i$. To insert v into T we find the edge $(u, w) \in E_T$ which minimizes the insertion cost, i.e., $c(u, v) + c(v, w) - c(u, w)$, and construct a sub-tour, denoted by $\mathrm{TOUR}(T, v)$, by deleting the edge (u, w) from T and adding the edges (u, v) and (v, w) to T.

In a GTSP graph with m vertex sets, an insertion method starts from a sub-tour T_1 with one vertex and creates a sequence of sub-tours $T_1, \ldots, T_m$ by inserting a vertex $v \in V_i$, where $V_i \cap T_i = \emptyset$ at each step, i.e.,

$$T_{i+1} = \mathrm{TOUR}(T_i, v).$$

The final tour T_m includes a vertex from each vertex set and the tour is an approximation of the optimal GTSP tour. For each insertion heuristic, a vertex set is chosen for insertion, and then the vertex in that vertex set with minimum insertion cost is inserted in the tour position that minimizes the insertion cost. To simplify the language, we refer to this insertion as "inserting a vertex set."

Insertion Heuristics: Given a sub-tour T *Nearest insertion* inserts the vertex set V_j not visited by the subtour containing the vertex with minimum distance from the tour:

$$\arg\min_{V_j} \min_{v \in V_j, u \in V_T} \{c(u, v)\}.$$

Cheapest insertion inserts the vertex set V_j containing a vertex with minimum insertion cost:

$$\arg\min_{V_j} \min_{v \in V_j, (u,w) \in E_T} \{c(u, v) + c(v, w) - c(u, w)\}.$$

Farthest insertion inserts the vertex set V_j whose closest vertex from the tour has the maximum distance from the tour:

$$\arg\max_{V_j} \min_{v \in V_j, u \in V_T} \{c(u, v)\}.$$

In this paper, we use a variation of the nearest insertion method where the method inserts the vertex set V_j containing the vertex with the minimum distance from or to the tour, i.e.,

$$\arg\min_{V_j} \min_{v \in V_j, u \in V_T} \min\{c(u, v), c(v, u)\},$$

which is helpful for non-symmetric edge costs [13].

The runtime to compute a GTSP tour with each insertion method is provided in [13], and is a straightforward extension of the runtime analysis in [6] for the TSP insertion methods. The cheapest insertion method for GTSP can be implemented to run in $O(|V| m \log m)$, while nearest and farthest insertion have runtimes in $O(|V| m)$.

2.3. *Vehicle Dynamics*

The following are three commonly-used vehicle models for which the shortest path between two configurations can be efficiently computed [40].

The **Dubins vehicle** model describes a forward moving vehicle with bounded turning radius. The equations of motion are

$$\dot{x} = v \cos\theta, \quad \dot{y} = v \sin\theta, \quad \dot{\theta} = \omega,$$

[a]A sub-tour is a cycle that visits only some of the vertex sets.

where $\omega \in [-v/R_{\min}, v/R_{\min}]$, $(x, y) \in \mathbb{R}^2$, $R_{\min}$ is the minimum turning radius of the vehicle, and v is the constant velocity.

The **Reeds-Shepp car** extends the Dubins model to allow the vehicle to travel in reverse, and is a more realistic model of a four-wheeled vehicle such as an automobile. The equations of motion are

$$\dot{x} = uv \cos\theta, \quad \dot{y} = uv \sin\theta, \quad \dot{\theta} = \omega,$$

where $u \in \{-1, 0, 1\}$ is the control input representing the forward and reverse gears.

Finally, the **differential drive robot (DD)** actuates two wheels independently and is capable of changing its heading without translation. The equations of motion are

$$\dot{x} = \frac{r(v_r + v_l)}{2}\cos\theta, \quad \dot{y} = \frac{r(v_r + v_l)}{2}\sin\theta, \quad \dot{\theta} = \frac{r(v_r - v_l)}{L},$$

where v_r and v_l, respectively, are the angular velocities of the right and left wheels, L is the distance between the wheels and r is the radius of the wheels.

In this paper, we use the term TSP with prefix of a vehicle model name (e.g., Dubins TSP) to denote a minimum length tour for the vehicle on a given set of vertices.

3. Problem Formulation and Approach

In this section, we present the task allocation problem and give a procedure for converting it into a GTSP instance.

3.1. *Task Allocation and Sequencing Problem*

Consider a group of N_r robots with differential constraints on their motion, located in a planar workspace $X \subseteq \mathbb{R}^2$. The team is given a set of N_t tasks to accomplish with minimum traveling distance. The location of a robot can be specified by an (x, y) location in X, and the configuration of the robot is a point in $Q = X \times \Theta$, where Θ describes the remaining states of the robot. For example, for the three models in Section 2.3, $\Theta = \mathbb{S}^1$ is the set of heading angles of the robot. We define a task t_i as a subset of Q, consisting of a subset of locations $X_i \subset X$ and a subset of states $\Theta_i \subset \Theta$:

$$t_i = \{(x, y, \boldsymbol{\theta}) | (x, y) \in X_i,\ \boldsymbol{\theta} \in \Theta_i\}.$$

Several common tasks fit in this definition. For example, the simple task of *visiting a location* using any configuration (i.e., any heading) is captured when X_i contains a single point and $\Theta_i = \Theta$. A constrained version where $\Theta_i \subset \Theta$ captures tasks in which only certain configurations can be used to complete the task. This task type appears in the inspection missions where a robot observes an event and the sensor is constrained to be directed in a range of headings. The more general task type captured by the definition is the task of *visiting a neighborhood* using any configuration where $X_i \subset \mathbb{R}^2$ and $\Theta_i = \Theta$. Finally, setting $X_i = \{(x_0, y_0), (x_1, y_1), \ldots, (x_k, y_k)\}$ captures *one-in-a-set tasks*, where a robot can complete a task by visiting just one of several locations. The *one-in-a-set tasks* appear in pick-up missions where an item can be picked up from multiple warehouses at different locations.

In this paper, we address the following problem: Suppose N_r robots, each starting at a fixed configuration, are given N_t tasks to perform collectively. The motion of the robots is governed with the models in Section 2.3. The robots communicate using a connected but possibly time-variant communication graph. The robots can swap tasks in their tours, only if there is an edge between the robots in the communication graph. The objective is to minimize the total time to perform all N_t tasks.

In the next section, we cast this problem as a mGTSP and in Section 6 we introduce our combinatorial auction-based decentralized method. We detail the construction of multiple tours by the GTSP insertion methods in Section 2.2. Finally, we leverage the GTSP insertion methods to create a decentralized method to swap tasks between the tours.

3.2. *Conversion to GTSP*

Our approach is to construct a GTSP graph that represents each task along with shortest tours between tasks. To do this, we discretize the configuration space at each task. The discretization of $X_i \times \Theta_i$ for task i into n points is denoted by t_i^n. For example, in [9] the Dubins TSP—in which each for task i $X_i = (x_i, y_i)$—is converted to a GTSP by selecting n equally spaced headings θ at the location (x_i, y_i). Thus, the discretized task i for this example is defined as

$$t_i^n = \Big\{(x_i, y_i, \theta_j) \mid \theta_j = \frac{2\pi}{n} j,\ \ j \in \{1, \ldots, n\}\Big\}.$$

Given N_r robots and N_t tasks, we construct a GTSP instance as follows. We define a complete graph $G = (V, E, c)$ and a partition of V into $m = N_t + N_r$ mutually disjoint subsets $V_1, V_2, \ldots, V_{N_t}, \overline{V}_1, \ldots, \overline{V}_{N_r}$, where

$$(\cup_{i=1}^{N_t} V_i) \cup (\cup_{r=1}^{N_r} \overline{V} r) = V.$$

The set V_i, where $i \in \{1, \ldots, N_t\}$, contains a vertex v for each discretized configurations of t_i^n. There is a single vertex in each vertex set $\overline{V}_r$, $r \in \{1, \ldots, N_r\}$ representing the initial configuration of each robot, namely depot. Additionally, we let $\mathbf{x}_u$ denote the location of the discretized configurations associated with the vertex u in the working space X. We will refer to $\mathbf{x}_u$ simply as the location of the vertex u. The weight of the edge $c(u, v)$ is the cost of the time optimal path between $(\mathbf{x}_u, \theta_u)$ and $(\mathbf{x}_v, \theta_v)$. The conversion to the GTSP relies on the efficient computation of optimal point-to-point paths between configurations, which can be performed for the three vehicle models in Section 2.3 [40]. In a general workspace with obstacles and other vehicle models, close to optimal paths are provided by the sampling based point-to-point planners [41, 42]. The cost function c is symmetric for the DD robot and Reed-Shepp's car and it is asymmetric for the Dubins vehicle, i.e. $c(u, v) \neq c(v, u)$.

Given the graph G, our objective is to find N_r tours $T^i = (V_{T^i}, E_{T^i}),\ i \in \{1, \ldots, N_r\}$ such that 1) each tour T^i includes the vertex in $\overline{V}_i$, 2) the tours collectively visit the N_t vertex sets exactly once and 3) the sum of the tour cost is minimized. In the next section we begin by addressing the centralized problem, and we develop a tour construction method that provides worst-case performance bounds and can be leveraged in the decentralized algorithm proposed in Section 6.

4. Constructing Tours By Insertion Methods

In this section, we analyze the performance of the GTSP insertion methods in Section 2.2 for construct tours for vehicles with differential motion constraints. We then provide an analysis of these methods for the mGTSP. Our analysis is an extension of that from [6] for the TSP, where it was shown that cheapest and nearest insertion provide 2-approximations to the optimal tour, and farthest insertion provides a $\lceil \ln n \rceil + 1$ approximation to the optimal.

4.1. *Bounds on the GTSP Tour Constructed By Insertion Methods*

In this section, we provide bounds on the cost of tours constructed by each of the GTSP insertion methods. Since each method is a generalization of the TSP insertion [6] to the GTSP, we extend the analysis in [6] to provide bounds. The results for the TSP insertion method hold only when the distances between the vertices are symmetric. Also, the TSP is characterized simply by the ordering of the vertices. The challenge in extending the TSP results is to bound an asymmetric GTSP tour cost where the cost depends not only on the ordering, but also on the vertex selected in each vertex set. In order to provide approximation factors for the insertion methods, we require the following assumptions on the edge costs.

Assumption 4.1 (Directed triangle inequality). The edge costs satisfy the directed-triangle inequality

$$c(u, v) + c(v, w) \geq c(u, w) \quad \text{for all } u, v, w \in V.$$

Assumption 4.2 (Bounded asymmetry in edge costs). There exists a constant k such that for each pair of vertices $u, v \in V$, the edge cost from u to v is within a constant of the edge cost from v to u:

$$\frac{c(u, v)}{c(v, u)} \leq k \quad \text{for all } u, v \in V.$$

Assumption 4.1 is the directed triangle inequality satisfied by the time optimal paths of the three vehicle models in Section 2.3. In Section 4.2, we show that Assumption 4.2 also holds for the time optimal paths of the three vehicle models. Note that $k = 1$ for symmetric GTSP instances.

Definition 4.3 (Task location density). We define d to be the maximum distance between the vertices in the same vertex set and ϵ to be the minimum Euclidean distance between vertices in different vertex sets. Then, we define the parameter $\rho = \frac{d}{\epsilon}$ to capture the density of the task locations.

In the problem of path planning with motion constraints, d is the maximum time to travel between the discretized configurations at a task location. Therefore, the density parameter represents the distribution of the tasks in an environment with respect to the motion constraints of the robots. Given Assumptions 4.1, 4.2 and the density parameter, we can establish bounds on a tour constructed by the GTSP insertion methods.

Proposition 4.4 (Bound on GTSP insertions). Suppose an insertion method, and let INSERT(G) be the cost of the GTSP tour constructed by an insertion method on G. Then,

$$\frac{\text{INSERT}(G)}{\text{GTSP}(G)} \leq k(1 + 2\rho)(\lceil \log m \rceil + 1).$$

Proof. Recall the directed-weighted graph $G = (V, E, c)$ from Section 2.2, and let $S \subseteq V$ be the set of vertices selected by an insertion method. Consider another complete symmetric weighted graph $G' = (S, E', c')$ on the vertex set S. Define the cost of the edge (s_i, s_j) in G' as $c'(s_i, s_j) = \min\{c(s_i, s_j), c(s_j, s_i)\}$.

Let TSP(G') be the cost of the optimal TSP tour in G'. Theorem 3 in [6] states that the TSP tour on G' constructed via any insertion method (i.e., using any insertion ordering, but inserting each vertex into its best edge of the sub-tour) is not greater than $(\lceil \log m \rceil + 1)\text{TSP}(G')$. Note that by Assumption 4.2 each edge $c(u, v)$ in the constructed tour is bounded by the $k \min\{c(u, v), c(v, u)\}$. Therefore, the GTSP tour constructed on the graph G with any insertion heuristic is

$$\text{INSERT}(G) \leq k(\lceil \log m \rceil + 1)\text{TSP}(G'). \tag{1}$$

Let GTSP(G) be the cost of the optimal GTSP tour in G and GTSP(S) be the cost of the GTSP tour in G obtained by including the vertices in S but using the vertex set ordering of the optimal GTSP tour. Then we have

$$\text{TSP}(G') \leq \text{GTSP}(S) \leq \text{GTSP}(G) + 2md. \tag{2}$$

With the definition of ϵ, we know that GTSP$(G) \geq m\epsilon$. Combining this with inequalities (1) and (2) we have,

$$\frac{\text{INSERT}(G)}{\text{GTSP}(G)} \leq k(1 + 2\rho)(\lceil \log m \rceil + 1).$$

□

For nearest and cheapest insertion methods we can improve the bound.

Proposition 4.5. Let *INSERT*(G) be the cost of the tour constructed using either nearest or cheapest insertion. Then

$$\frac{\text{INSERT}(G)}{\text{GTSP}(G)} \leq (1 + k)(1 + 2\rho).$$

Prior to the proof of the proposition, we establish a property of the nearest and cheapest insertions. Let T_i be the tour after i insertion steps and $s_i \in V_i$ be the vertex that the nearest insertion for the GTSP inserts at the step i. Let v_i be the vertex in V_i which has the closest distance from or to the tour. Without loss of generality assume that u is the closest vertex in the tour, then the cost of inserting v_i into tour T_i is at most equal to the cost of inserting v_i between u and w, i.e.

$$\mathrm{cost}(T_i, v_i) \leq c(u, v_i) + c(v_i, w) - c(u, w).$$

Lemma 4.6 (Bound on insertion cost). Cost of inserting s_i in the sub-tour T_i is

$$\mathrm{cost}(T_i, s_i) \leq (1+k)c'(p, q) \ \forall p \in V_{T_i}, q \in S \setminus V_{T_i}.$$

Proof. Consider the set S and the graph G'. The nearest and cheapest insertions insert the vertex in V_i with the minimum insertion cost, then we have,

$$\mathrm{cost}(T_i, s_i) \leq \mathrm{cost}(T_i, v_i).$$

Then, by Assumptions 4.2 and 4.1 we have,

$$\begin{aligned} \mathrm{cost}(T_i, s_i) \leq \mathrm{cost}(T_i, v_i) &\leq c(u, v_i) + c(v_i, u) \qquad (3)\\ &\leq (1+k)\min\{c(u, v_i), c(v_i, u)\}. \end{aligned}$$

Since, v_i is the closest vertex from or to the tour, then following inequality holds for the each step of insertions,

$$\min\{c(u, v_i), c(v_i, u)\} \leq c'(p, q) \ \forall p \in V_{T_i}, q \in S \setminus V_{T_i}. \quad (4)$$

From inequalities (3) and (4), we conclude the lemma. □

Recall that the cheapest insertion inserts a vertex to the sub-tour with the minimum insertion cost. The cost of inserting a vertex by the cheapest insertion into the sub-tour T_i is at most $\mathrm{cost}(T_i, s_i)$, thus the bound on the insertion cost in Lemma 4.6 holds for the cheapest insertion.

Further, we need a modification of a result in [6] for GTSP insertions as follows. The proof follows directly from the proof of Lemma 3 in [6].

Lemma 4.7 (Modified Lemma 3 in [6]). Suppose an insertion method inserting vertex s_i in sub-tour T_i such that satisfies

$$\mathrm{cost}(T_i, s_i) \leq (1+k)c'(p, q) \ \forall p \in V_{T_i}, q \in S \setminus V_{T_i},$$

then,

$$\mathrm{INSERT}(G) \leq (1+k)\mathrm{MST}(G').$$

Now we can establish the approximation factors in Proposition 4.5 for the nearest and cheapest insertions.

Proof. [Proof of Proposition 4.5] By Lemma 4.6, each insertion using the nearest or cheapest insertion methods satisfy the condition in Lemma 4.7. Therefore, the cost of the tour constructed by the insertions is not greater than $(1+k)\mathrm{MST}(G')$. Then, we have

$$\mathrm{INSERT}(G) \leq (1+k)\mathrm{MST}(G') \leq (1+k)\mathrm{TSP}(G').$$

Since, the edge costs for graph G' satisfies $c'(u, v) \leq c(u, v), \forall u, v \in S$, then, $\mathrm{TSP}(G')$ is a lower bound on the cost of the tour in G, consisting of the vertices in S and with the vertex ordering of the optimal GTSP tour. Therefore, we have,

$$\begin{aligned} \mathrm{INSERT}(G) \leq (1+k)\mathrm{TSP}(G') &\leq (1+k)\mathrm{GTSP}(S)\\ &\leq (1+k)(1+2\rho)\mathrm{GTSP}(G). \end{aligned}$$ □

4.2. *Bound on Vehicle Tour Cost*

The following result gives an upper-bound on the optimal path cost between two configurations for each of the three vehicle models in Section 2. The Dubins' bound (i) was conjectured in [8] and established in [43], while the Reeds-Shepp and DD bounds are, to the best of our knowledge, new results. The proof of Lemma 4.8 (i) can be found in [43], while the proof of (ii) and (iii) are given in Appendix 1. Recall that the Dubins' and Reeds-Shepp's cars are characterized with the minimum turning radius $R_{\min}$ and forward velocity v, and the parameters characterizing motion of DD robot are wheel radius r, distance between the wheels L and forward velocity.

Lemma 4.8 (Distance of robot configurations). Consider a robot whose dynamics are governed by one of the three vehicle models (Dubins, Reeds-Shepp, DD), v forward velocity and two robot configurations q_1 and q_2. Then, the travel time from q_1 to q_2, denoted $\mathrm{dist}(q_1, q_2)$ satisfies

$$\mathrm{dist}(q_1, q_2) \leq \mathrm{Euc}(q_1, q_2) + \mathcal{C},$$

where Euc is the time to travel then Euclidean distance between the points (x_{q_1}, y_{q_1}) and (x_{q_2}, y_{q_2}). $\mathcal{C}$ is defined in terms of the vehicle model as

(i) $\mathcal{C} = \frac{7\pi}{3v}R_{\min}$ for Dubins;
(ii) $\mathcal{C} = \frac{\pi}{2v}\frac{L}{r}$ for Differential Drive; and
(iii) $\mathcal{C} = \frac{\pi}{v}R_{\min}$ for Reeds-Shepp.

To compare the performance bound on the insertion methods to the existing approximation in Lemma 4.8 we assume that each task consists of a single workspace location, i.e., $|X_i| = 1$. In this case, the Euclidean distance between the vertices inside the same vertex set is zero, and thus $d_{\min} = \mathcal{C}$ and $\rho = \frac{\mathcal{C}}{\epsilon}$. For the case of a Dubins vehicle, the distance bounds from Lemma 4.8 allow us to provide tighter performance bounds. To this end, we redefine the edge costs in the graph G' to the Euclidean distance between the vertices, i.e., $c'(s_i, s_j) = \mathrm{Euc}(\mathbf{x}_{s_i}, \mathbf{x}_{s_j})$.

The distance between the configurations, by the Lemma 4.8, satisfies the following,

$$\frac{\mathrm{dist}(q_1, q_2)}{\mathrm{dist}(q_2, q_1)} \leq 1 + \rho.$$

Then the constant k in Assumption 4.2 becomes $1+\rho$. Thus, the bound in (1) becomes

$$\mathrm{INSERT}(G) \leq (1+2\rho)^2(\lceil \log m \rceil + 1)\mathrm{TSP}(G').$$

Let VTSP be the optimal tour cost for a vehicle model in Section 2.3 in a workspace $X = \mathbb{R}^2$, then we have the following result.

Corollary 4.9. The total tour cost constructed by the nearest or cheapest insertion is bounded by

$$\text{INSERT}(G) \leq 2(1+\rho)\text{VTSP}.$$

Proof. From Lemma 4.8 and inequality (3), the cost of inserting any vertex in the tour at step i by the nearest and cheapest insertion is

$$\text{cost}(T_i, s_i) \leq c(u, s_i) + c(s_i, u) \leq \\ 2(1+\rho)\text{Euc}(\mathbf{x}_p, \mathbf{x}_q)\ \forall p \in V_{T_i}, q \in S \setminus V_{T_i}.$$

From Lemma 4.7, we have,

$$\text{INSERT}(G) = \sum_{i=1}^{m} \text{cost}(T_i, s_i) \leq 2(1+\rho)\text{MST}(G'). \quad (5)$$

The MST(G') are the minimum spanning tree and the optimal TSP in the graph G', respectively. The minimum spanning tree and the optimal tour on the task locations are shorter than the optimal tour between the locations for the vehicles VTSP. Finally, the approximation factor for our cheapest and nearest insertion methods is as follows:

$$\frac{\text{INSERT}(G)}{\text{VTSP}} \leq \min\{(\lceil \log \text{m} \rceil + 1)(1+2\rho)^2, 2(1+\rho)\}. \quad \square$$

Note that for environments with large values of ρ and $m \geq 5$ this bound becomes $2(1+\rho)$, which is an improvement over the bound for the Dubins TSP method in [9], which was

$$\min\{(1+\rho)\log \text{m}, \frac{3}{2}(1+\rho)^2\}.$$

In the case that the minimum turning radius for the Dubins and the Reeds-Shepp's models and distance between the wheels for the DD robot are negligible compared to the distances between task location i.e. $\rho \approx 0$, the problem becomes the TSP on the task locations, and both nearest and cheapest insertion provide 2-approximations to the optimal.

5. Bounds on the Tours of Multiple Robots

In Section 4.2, we provided the approximation factors of the insertion methods for a single robot. In this section, we extend the analysis to multiple robots with various motion dynamics. The high-level idea is to reduce the mGTSP problem to a GTSP problem and construct a tour using the insertion methods.

5.1. *Transformation of mGTSP to GTP*

The problem of allocating tasks to multiple robots is formulated as an mGTSP in [44,45], and transformations are given from mGTSP to single GTSP problem. The transformation creates a GTSP instance consisting of duplicates of tasks for each robot and a duplicate of each depot. In this subsection, for completeness, we present the transformation from [44]. Our presentation includes a slight modification, by adding a dummy vertex, which we require to establish bounds on the cost of multiple tours constructed by cheapest and nearest insertion methods.

Recall from Section 3.1 that a vertex in the GTSP graph represents a configuration at a task. However, in a heterogeneous system of robots, the discretized configurations and the time to travel between two configurations are, in general, different for each robot. Therefore, we let V_i^r denote the vertex set consisting of the configurations of robot r at task i. Each vertex u is a associated with a configuration of a robot, and we let model(u) denote the model governing the motion of the robot. Two vertices u, v in V_i^r and V_i^k are considered equal if and only if they represent an identical configuration and model(u) = model(v). Since, only one configuration should be visited at task i, the vertex set V_i is defined as $V_i = \bigcup_{r=1}^{N_r} V_i^r$.

Let $\overline{V} = \{\overline{V}_1, \ldots, \overline{V}_{N_r}\}$ denote the set of depots for the robots. Each $\overline{V}_r$ is a vertex set consisting of the discretization of the configuration space at the depot location of robot r. Also, the replicas of the vertices at the depots are defined in the set $\overline{V}' = \{\overline{V}'_1, \ldots, \overline{V}'_{N_r}\}$. With a slight modification, we introduce a dummy vertex as a starting point for the insertions.

Let dist : $(V \cup \overline{V}) \times (V \cup \overline{V}') \to \mathbb{R}$ record the distance between any two robot configurations (at tasks or depots). For $u \in V \cup \overline{V}$ and $v \in V \cup \overline{V}'$ such that model(u) = model(v), we define dist(u, v) to be the minimum time required to travel from u to v using model(u). For all other pairs of vertices, we set the distance to a large number M that exceeds the total length of all robot tours:

$$M = (N_t + N_r)\max\{\text{dist}(u, v) \mid \text{model}(u) = \text{model}(v)\}.$$

The transformation of the multi-robot task allocation to the single GTSP problem is as follows. We define a complete weighted graph $G = (W, E, c)$ where

$$W = V \cup \overline{V} \cup \overline{V}' \cup \{\text{dummy}\}.$$

The edge costs $c(u, v)$ for each $u, v \in W$ are set to

(i) dist(u, v) if $u \in V_i^r, v \in V_j^k, \forall r, k \in \{1, \ldots, N_r\}$ and $\forall i, j \in \{1, \ldots, N_t\}$;
(ii) dist(u, v) if $u \in \overline{V}_r, v \in V_i^r, r \in \{1, \ldots, N_r\}$ and $i \in \{1, \ldots, N_t\}$;
(iii) dist(u, v) if $u \in V_i^r$, $v \in \overline{V}'_r, r \in \{1, \ldots, N_r\}$ and $i \in \{1, \ldots, N_t\}$;
(iv) zero if u = dummy and $v \in \overline{V}_1$;
(v) zero if v = dummy, $u \in \overline{V}'_i$ and $r \in \{1, \ldots, N_r\}$;

(vi) zero if $u \in \overline{V}'_r$, $v \in \overline{V}_{r+1}$ and $r \in \{1, \dots, N_r - 1\}$; and finally
(vii) zero if $u \in \overline{V}_r$, $v \in \overline{V}'_r$ and $r \in \{1, \dots, N_r\}$.

For all other edges, the cost $c(u, v) = M$.

Figure 1 shows an example of the G graph for a system of two heterogeneous robots and two tasks. The vertices associated with the configurations of each robot are represented with similar shapes to the depots. For clarity, edges with cost M are not included and the zero cost edges are represented with dashed lines.

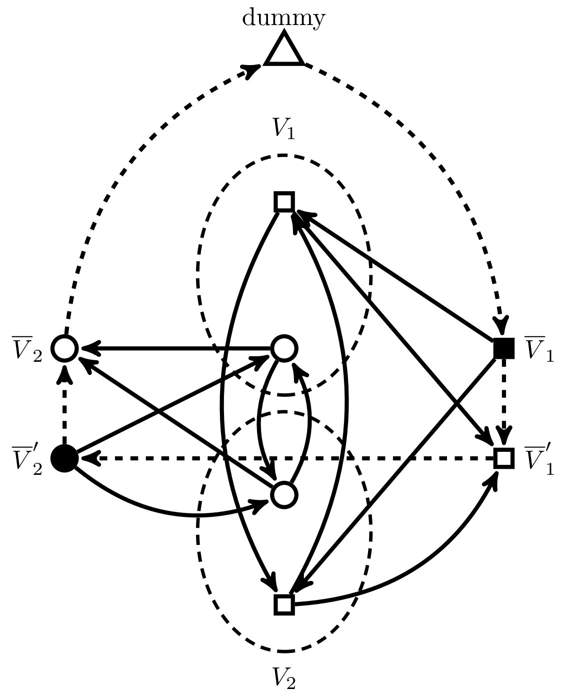

Fig. 1. Graph G, without cost M edges, for an instance with two heterogeneous robots and two tasks. Similar shapes are used to represent the depots and the configurations at task locations for each robot. The dashed edges represent the zero cost edges in G.

5.2. *Constructing Multiple Tours with Insertion Methods*

Let INSERT(G) be the cost of the tour generated by the nearest or cheapest insertions on G and TSP be the total cost of the optimal tours. Let S be the set of vertex sets selected by an insertion method. Now, we define a complete symmetric weighted graph $G' = (S, E', c')$ where the edge cost $c'(u, v)$ for $u, v \in S$ is set to

(i) Euc$(\mathbf{x}_u, \mathbf{x}_v)$ if $u, v \in V$;
(ii) Euc$(\mathbf{x}_u, \mathbf{x}_v)$ if $u \in \overline{V}_r$ and $v \in V$;
(iii) Euc$(\mathbf{x}_u, \mathbf{x}_v)$ if $u \in V$, $v \in \overline{V}'_r$ and $r \in \{1, \dots, N_r\}$;
(iv) zero if $u =$ dummy, $v \in \overline{V}_1$;
(v) zero if $v =$ dummy, $u \in \overline{V}'_r$ and $r \in \{1, \dots, N_r\}$;
(vi) zero if $u \in \overline{V}'_r$, $v \in \overline{V}_{r+1}$ and $r \in \{1, \dots, N_r - 1\}$; and finally
(vii) zero if $u \in \overline{V}_r$, $v \in \overline{V}'_r$ and $r \in \{1, \dots, N_r\}$.

For all other edges, the cost is set to M.

Prior to providing the bounds on the tours constructed by the nearest and cheapest insertion methods for multiple robots we first establish a result on lower bounds for optimal tours.

Let mVTSP be the minimum total cost of the tours for multiple robots with differential constraints in a workspace $X = \mathbb{R}^2$, and let TOUR be the set of optimal tours with optimal allocation, ordering and configurations. Then we have,

Lemma 5.1 (Lower bound on optimal tours). The length of the minimum spanning tree MST(G') on the graph G' is less than or equal to mVTSP.

Proof. Deleting an edge from each optimal tour in TOUR result in a set N_r trees. These trees collectively visit all the depots and the tasks with optimal configurations. Connecting these trees with zero costs edges to a dummy vertex creates a spanning tree on the optimal configurations. Note that the spanning tree is a lower bound on the total cost of the optimal tours. Note that for every edge (u, v) in the constructed minimum spanning tree, the cost c(u, v) is lower-bounded by the Euc$(\mathbf{x}_u, \mathbf{x}_v)$. Therefore, MST$(G')$ is at most equal to the cost of the spanning tree on the optimal configurations.

□

In Section 4.2, we defined ρ for three vehicle dynamics. Let $\rho_{\max}$ be the maximum ρ in the set of multiple heterogeneous robots.

Theorem 5.2 (Total cost of multiple tours). The total tour cost constructed by the nearest or cheapest insertion is bounded by

$$\text{INSERT}(G) \leq 2(1 + \rho_{\max})\text{mVTSP}.$$

Proof. Consider tour T_i as the sub-tour constructed by the nearest or cheapest insertions at step i on graph G. Starting from the dummy vertex, the nearest insertion method inserts the vertex at the first depot $u \in \overline{V}_1$ with distance zero from the dummy vertex (since this is the only outgoing edge from dummy at distance zero). At the next step, the closest vertex set to the tour is the replica of the previously selected depot with zero cost. The nearest insertion method selects the replica of u in $\in \overline{V}'_i$, namely u', and inserts u' in the tour (recall $c(u, u') = 0$).

The nearest insertion continues to insert a vertex in each depot and again inserts the replica of the vertex afterward. Since the vertices are inserted in a position in the sub-tour with the minimum insertion cost, each vertex $u' \in \overline{V}'_i$ is adjacent to the vertex $u \in \overline{V}_i$ in the sub-tour. Otherwise, the insertion cost in each insertion is M.

The nearest insertion creates a sub-tour consisting of all the depots and their replicas with zero cost edges. Therefore, the total sub-tour cost after adding all the depots and their replicas is zero. Note that the cheapest insertion constructs the same sub-tour by inserting vertices with zero insertion cost. Also, note that, by the definition of the zero costs edges in the graph G, the $u' \in \overline{V}'_r$ in the sub-tour is adjacent to the vertex in $\overline{V}_{r+1}$. Proceeding in the process of inserting the tasks, the nearest insertion inserts the vertices in V to the tour. Note that, by the definition of the edge costs in G, the insertion cost of a task in V between $\overline{V}'_r$ and $\overline{V}_{r+1}$ is $2M$. Similarly, the cost of inserting a vertex in between $\overline{V}'_i$ and the dummy vertex is $2M$. Therefore, the nearest or cheapest insertion methods only insert tasks in between depots and their replicas, i.e. between $\overline{V}_r$ and $\overline{V}'_r, r \in \{1, \dots, N_r\}$. Suppose, at the step ith of insertions, the insertion methods insert s_i in between $\overline{V}_r$ and $\overline{V}'_r$. By Lemma 4.6, at each insertion of the nearest and cheapest insertion methods we have,

$$\mathrm{cost}(T_i, s_i) \leq 2(1+\rho_r)\mathrm{Euc}(p,q) \ \forall p \in T_i, q \in S \setminus T_i. \quad (6)$$

By Lemma 4.7, the inequality (6) is sufficient to show that the sum of the right-hand side of Equation (6) is INSERT(G), and the sum of the left-hand side is the minimum-spanning tree on G'. Therefore, from Lemma 5.1 and inequality (6) we conclude the bound on INSERT(G). □

Obtaining N_r tours from the tour constructed by the insertion methods in G consists of two steps. First, deleting the zero cost edges and the dummy vertex outputs N_r paths. Second, deleting the vertices in the replicas of the depots and closing the paths by adding the edge from the last vertex of each path to the vertices of the depots result in N_r tours. Note that the total tour length is preserved under the extraction of N_r tours, since the added edge costs to close the paths are equal to the edge costs of the deleted edges.

So far, the algorithms of creating tours by the insertion methods were centralized. In the next section, we propose a decentralized algorithm for constructing the tours.

6. A Decentralized Auction-Based Algorithm

We now present a decentralized algorithm for task allocation and sequencing. The algorithm leverages the GTSP insertion mechanisms presented in Section 4 in a manner similar to large neighborhood search (LNS) [32].

6.1. *Large Neighborhood Search (LNS)*

The large neighborhood search (LNS) meta-heuristic is a framework initially introduced for the routing problems [32]. In LNS, the solution is repeatedly improved through destroy (deletion) and repair (insertion) operations. This method is a form of local search where at each iteration an improving solution is found by searching in a "large neighborhood" of the current solution. Typically, by searching in a larger neighborhood one is able to find higher quality locally optimal solutions. The neighborhood is defined by the heuristics used to destroy and repair solutions (in our case, the heuristics used to remove and reinsert vertices in the tour). The basic idea is to perform repeated rounds of insertions to improve the tour. First, an initial insertion method (e.g., nearest or farthest insertion from Section 2.2) is chosen and a tour is constructed. Second, a subset of vertices on the tour are deleted and then the corresponding vertex sets are reinserted into the tour using a chosen insertion method. The insertion method is chosen by a criterion (e.g., an adaptive probability density function as in [13]). In this paper, the insertion methods are chosen via the roulette wheel random selection algorithm [33]. This procedure is repeated, accepting tours when they pass an acceptance criterion (for example, if the new tour has smaller cost). In practice, one can perform LNS heuristic on the tours constructed in Section 4.1 to improve the tour quality. Recently a solver for the GTSP problem is proposed in [13] based on the LNS method. In contrast the proposed method in this paper is decentralized and addresses the multiple GTSP problem. Algorithm 1 shows a high-level description of LNS for the GTSP problem.

Algorithm 1 The LNS Framework for the GTSP

```
Input: A GTSP instance $G$
Output: A GTSP tour $T$ on $G$
1: $T \leftarrow$ Create an initial tour
2: while stopping criteria not met :
3:     Select an insertion method $I$ for the GTSP
4:     Delete a set of vertices in the tour $T$
5:     Create $T_{new}$ by reinserting vertices into $T$ via $I$
6:     if cost($T_{new}$) $\leq$ cost($T$) :
7:         $T \leftarrow T_{new}$
```

6.2. *LNS-Auction Algorithm*

The high-level idea of our decentralized implementation of LNS, namely LNS-Auction, is to delete a set of tasks from a robot's tour and then reinsert them in new robot tours via an auction. We assume that there is an initial assignment of tasks to robots that is conflict-free, i.e., each task is assigned to one robot, and that the communication graph between robots is connected for all time. In the literature, conflicts are solved by an additional consensus algorithm.

The high-level description of the auction procedure is as follows:

(i) A robot randomly decides to begin an auction by selecting a set of robots in its communication range.
(ii) The robot randomly selects a set of tasks and sequentially deletes them from its tour, takes the auctioneer role and offers the selected robots to

bid on the tasks.

(iii) The robots select an insertion method and place bids on subsets of tasks that can be inserted as a continuous segment on its tour.

(iv) Each robot sends its bids (each consisting of a subset of tasks and a corresponding bid value) to the auctioneer, who solves the combinatorial auction problem to allocate the tasks.

(v) The auctioneer communicates the result of the auction to the winning robots.

(vi) Each robot inserts the tasks it has won and locally optimizes its tour.

We now give a more detailed description of the steps. In Step (i), each robot assigns a score to each other robot and increases scores of the winning robots after each auction. The robot selection operation is a roulette wheel selection algorithm.

The bidding starts at Step (ii). Assume the auctioneer is robot r. Its bids are generated by sequentially deleting tasks from its tour T^r. Without loss of generality, let $V_1, V_2 \dots, V_d$ be the tasks deleted from the auctioneer's tour and $S_i^r = \{V_1, \dots, V_i\}, i \in \{1, 2, \dots, d\}$ be the subset of tasks until the i^{th} step of deletions.

The bidding algorithm generates a bid after each deletion, which is a pair consisting of subset of S_i^r containing V_i and a non-negative bid on the subset:

$$\text{BID}_i^r = (\texttt{bid-set}_i, \texttt{bid-value}_i).$$

The set $\texttt{bid-set}_i$ is the largest subset of S_i containing V_i that forms a continuous segment of T^r and $\texttt{bid-value}_i$ is the insertion cost of the segment of T^r corresponding to $\texttt{bid-set}_i$, see Figure 2.

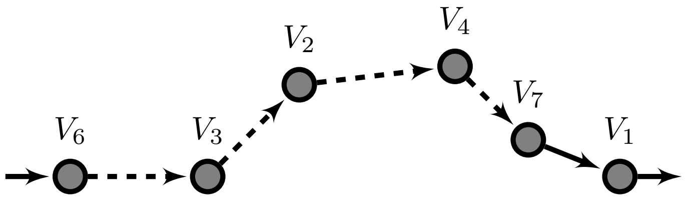

Fig. 2. Largest subset of $S_4 = \{V_1, V_2, V_3, V_4\}$ containing V_4 that forms a segment on the tour is $\{V_3, V_2, V_4\}$. The insertion cost of $\{V_3, V_2, V_4\}$ is the difference between the cost of the segment shown in dashed arrows and the cost of the edge from V_6 to V_7.

At Step (iii) robots generate subsets to bid on given their current tour and the set of tasks from the auctioneer. Given a set of tasks up for auction $\{V_1, V_2, \dots, V_d\}$, each robot in the auction selects an insertion method and sequentially inserts the tasks in their tour. The set $S_i^r, i \in \{1, 2, \dots, d\}$ is the set first i tasks inserted in the tour of robot r. Again we assume (without loss of generality) that $S_i^r = \{V_1, \dots, V_i\}$. Let T_i^r be the tour of robot r at step i of insertions. The bidding mechanism is closely related to the auctioneers bidding process in Step ii with a subtle but important difference. Unlike, the bidding for the auctioneer where the algorithm searches for the largest segment on the original tour, the algorithm for all other robots returns the largest segment in T_i^r. More precisely, in BID_i^r, the set $\texttt{bid-set}_i$ is the largest subset of S_i^r containing V_i that forms a continuous segment of T_i^r. The bid value $\texttt{bid-value}_i^r$ is the insertion cost of the segment in T_i^r.

The auctioneers goal is to find an allocation that assigns each task to exactly one robot and such that the sum of the winner bids on the subsets is minimum. Determining the winners of the auction at Step (iv) is formulated as a combinatorial auction problem whose integer programming formulation (IP) is as follows:

$$\begin{aligned}
&\text{minimize} \sum_{i,r} \texttt{bid-value}_i^r \cdot x_i^r \\
&\text{subject to} \\
&\sum_{i,r} x_i^r = 1 \quad \text{for all } i \in \{1, \dots, d\},\ r \in \mathcal{I}_r, \\
&x_i^r \in \{0,1\} \quad \text{for all } i \in \{1, \dots, d\},\ r \in \mathcal{I}_r.
\end{aligned} \tag{7}$$

where x_i^r is an integer variable equal to 1 if the bid BID_i^r is accepted and 0 otherwise, and $\mathcal{I}_r$ contains the indices of all robots participating in the auction. Note that the IP is feasible since the auctioneer's bids form a feasible solution. Several techniques are proposed to reduce the run-time of solving this IP [46] which enables fast solutions for large sizes. We show in Lemma 6.1 that the number of the variables is linear in the size of the problem, and in Section 7 we experimentally evaluate the fraction of time needed to solve the IP.

Step (v) is a simple process by the auctioneer to broadcast the result of the auction. At Step (vi) each robot inserts its winning bids in the same position on the tour it used to generate the bid. The consistency of the insertion methods in the bidding and insertion steps is required to prove the monotonic decrease in the total tour cost after each auction. Further optimization is done by robots locally. After completing the auction, each robot locally optimizes its tour by deleting tasks from its tour and reinserting then via the GTSP insertion methods of Section 4.

Let D be the maximum number of tasks offered for an auction. Assuming that each edge cost can be encoded with at most k bits, the following lemma summarizes properties of the proposed algorithm.

Lemma 6.1 (Auction algorithm properties). The decentralized auction algorithm has the following properties:

(i) The message size per robot per auction is at most $D(D \log D + k)$.

(ii) The number of the variables in the IP (7) for a set of m robots is at most $\min\{mD, 2^D - 1\}$.

(iii) The total tour cost monotonically decreases after each auction.

Proof. Proof of (i): At each step of bidding, only a single new bid is added. Therefore, in total D bids are submitted by each robot. Moreover, the cardinality of the bid-sets is at most D. Each bid value is a result of the summation of $D-1$ edge costs and each bid value is encoded by $\log D + k$ bits. Thus, the number of bits transferred in an auction per robot is at most $D(D \log D + k)$.

Proof of (ii): Note that the set of all possible combinations of tasks is the power-set of the tasks. Also, note that without loss of optimality we can replace all the bids submitted on the same bid-set by the bid with minimum bid-value. From i, we know that each robot submits at most D bids, therefore, the total number of bids in the IP (7) is equivalent to $\min\{mD, 2^D - 1\}$

Proof of (iii): Let $\text{cost}(T^r)$ be the tour cost of robot r prior to the auction. Let operation $\oplus$ represent inserting a set of tasks to a tour by an insertion method. Define the set $\text{win}(r)$ as the set of tasks that robot r has won. The bid-values are the insertion costs of the bid-sets. Therefore,

$$\text{cost}(T^r \oplus \text{win}(r)) = \text{cost}(T^r) + \sum_i \texttt{bid-value}_i^r \cdot x_i^r.$$

This holds for any robot in an auction. Let z^* denote the optimal solution to the integer program (7) and z be the objective value of assigning all tasks to the auctioneer. Thus, the total path tour is

$$\sum \text{cost}(T^r \oplus \text{win}(r)) - \sum \text{cost}(T^r) = z^* - z. \quad (8)$$

Note that assigning all the tasks to the auctioneer is a feasible solution to (7). As a consequence, the right-hand side for (8) is not greater than zero. □

7. Simulation Results

In this section, we compare our decentralized algorithm to several different existing methods. We compare to two different implementations of the Consensus-Based Bundle Algorithm (CBBA)[b] from [21]. The CBBA algorithm plays the benchmarking role in recent studies as the state of the art decentralized task allocation algorithm [47,48]. We also compare results to the Multi-Vehicle Algorithm (MVA) [37] and to a centralized algorithm [44].

7.1. *LNS-Auction Implementation Details*

The experiments featuring the LNS-Auction are initiated with a random allocation of the tasks to the robots. The number of auctions in our experiments is limited to N_t, where at each round of auctions the auctioneer is selected randomly. The auctioneer communicates with all the neighbor robots in the communication graph, and offers a number of tasks which is selected randomly between two and the number of task in the tour of the auctioneer.

In the LNS-Auction, each robot locally optimizes its tour after each round of the auction, using the method detailed in [1, Section 4]. Given a tour, this method fixes the ordering of the vertex sets, and re-optimizes the vertex chosen from each set. With a fixed ordering of the tasks, the graph G is a multi-partite graph. Therefore, in a problem with n vertices for each task, the vertex optimization method performs n breadth-first-searches [1].

Once all auctions are finished, each robot performs one final re-optimization of its tour. This optimization method is a modified implementation of Algorithm 1. The tour acceptance criteria in this optimization method is a simulated-annealing method [33] criterion. The robots randomly select a set of tasks from their tours and re-inserts by the GTSP insertion methods to re-optimize the tour. The selection of the insertion method at the final re-optimization step follows the adaptive method detailed in [32]. Each insertion method is assigned an initial score. After each iteration and if the cost of the tour has improved, the score of the insertion method is increased by a constant value. The insertion method at each iteration is selected randomly with probability proportional to the scores (i.e., roulette wheel selection). The stopping criteria for the rounds of insertions and deletions is the number of the rounds without improvement in the tour cost. This is set to 1000 in our experiments.

7.2. *Existing Methods For Comparison*

In this section, we provide a brief description of the existing approaches for the task allocation problem.

CBBA-AA: The first implementation of CBBA finds the assignment without considering dynamics for the robots. Each task consists of visiting a location with any heading, and we consider Dubins vehicle dynamics. In this implementation, we linked the CBBA and the Alternating Algorithm (AA)[8] to create feasible solutions to the Dubins vehicle after the assignment. Given an Euclidean TSP solution, the AA assigns a heading to each point based on the position of the point in the tour. Let T_{ij} be the jth task in the tour of robot i. If j is odd then heading with the AA procedure is the orientation of segment from T_{ij} to $T_{i(j+1)}$, otherwise the heading is equal to the heading assigned to the previous point in the tour.

CBBA-GLKH: In the second implementation of the CBBA, we linked the CBBA and the state of the art GTSP solver, GLKH [12]. After assigning the tasks by CBBA, the set of tasks assigned to each robot is transformed to a GTSP instance by discretization of the heading at each task location and solved separately by GLKH. The GTSP instance consists of vertices representing the configuration of the robots and the edges representing the optimal Dubins path length between each pair of configurations.

MVA: The MVA algorithm presented in [37], considers decoupling the dynamic constraints of the robots from the

[b]The CBBA code is available at http://acl.mit.edu/projects/cbba.html

assignment problem. After task assignment and sequencing by a decentralized variation of the Prim's algorithm, a heuristic method is proposed to construct feasible Dubins tours on the sequenced tasks. Assuming $2R_{\min}$ pair-wise Euclidean distance between the tasks and starting from a configuration at a depot, the algorithm plans the optimal path to the next task in the tour. The MVA algorithm is constrained to the $2R_{\min}$ distance between the points, therefore, this algorithm is augmented by an exhaustive search method for the cases where the pair-wise Euclidean distances between the tasks are smaller than $2R_{\min}$. The exhaustive search method searches all the candidate optimal Dubins paths to the next point with a fine discretization of the headings (0.1 degree) at the point and returns the Dubins path with minimum cost.

Centralized: Our method is compared to a centralized algorithm [18] in both heterogeneous and homogeneous system of robots. The problem of task allocation and sequencing for multiple robots with dynamic constraints is transformed to a GTSP problem and solved by the state-of-the-art GTSP solver [12]. The GTSP construction is detailed in Section 5.1.

The experiments are conducted on random communication graphs in which robot i and j can communicate with probability p. In each experiment, robots are initialized by randomly assigning the tasks without any conflicts. Given the initial allocation, each robot constructs a path by an insertion method.

7.3. *Random Instances*

In this section, we compare the quality of the multiple tours constructed by our decentralized implementation of LNS, namely LNS-Auction, to the various centralized and decentralized methods. The experiments are conducted on uniformly randomly generated tasks in $X = [0, 10] \times [0, 10]$, and a task is accomplished by visiting the task location with any heading, i.e., $\Theta = [0, 2\pi[$.

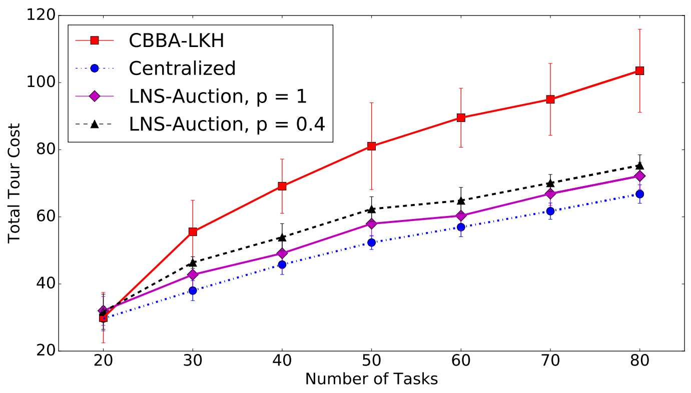

Fig. 3. Total tour cost versus the number of tasks for seven robots on uniformly randomly generated instances in a 10×10 square environment. The robots do not have any constraint on their motion $\rho = 0$. For each different number of tasks, 30 random instances are generated. The reported tour cost for each number of tasks is the average of 30 instances, each solved 10 times. The error bars represent the standard deviation of the results in different runs and instances.

Figure 3 compares the total tour cost of the LNS-Auction to CBBA-GLKH and the centralized method for different random instances. Since the published CBBA implementation constructs paths instead of tours, therefore, a post-processing on the paths by the state of the art GTSP solver, namely GLKH, is performed on the paths in order to improve the quality. The communication graphs are generated for each $p \in \{0.4, 1\}$. Note that the experiments with $p = 1$ corresponds to task allocation with all-to-all communication. At each run of the algorithm, a new communication graph is generated. The seven robots in the system are assumed to have no motion constraints. The number of auctions for the LNS-Auction algorithm is equal to the number of the tasks. Without considering any dynamics for the robots, Figure 3 shows that the LNS-Auction algorithm for different communication graphs gives considerable improvements in tour quality compared to CBBA-GLKH.

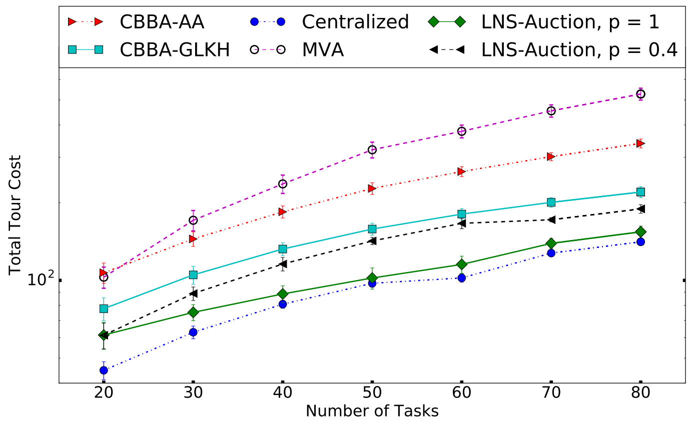

Fig. 4. Total tour cost versus the number of tasks for seven Dubins' vehicles with turning radius of 1 on uniformly randomly generated instances in a 10×10 square environment. For each different number of tasks, 30 random instances are generated. The reported tour cost for each number of tasks is the average of 30 instances, each solved 10 times. The error bars represent the standard deviation of the results in different runs and instances.

In the next experiment, we replace the robots with seven Dubins vehicles with minimum turning radius of 1. To construct the GTSP instance we discretize the heading at each location with 5 equally spaced headings. Figure 4 shows the tour costs in different random instances compared to the CBBA-AA, CBBA-GLKH, MVA and the centralized method on random instances. The maximum deviation of the tour cost of the LNS-Auction from the centralized algorithm is 37% and the average is 16%. However, the maximum deviation of the tour cost of the CBBA-GLKH from the centralized algorithm is 76% and the average deviation is 65%.

Table 1. Total tour cost on TSPLIB instances. The LNS-Auction used the communication graph with $p = 1.0$. Avg. denotes the average total tour length and Min. denotes the minimum total tour length on different runs of the LNS-Auction. Auction indicates the fraction of the total LNS-Auction time which is spent for solving I.P. (7), and Ratio indicates the average ratio of time for LNS-Auction to the CBBA-GLKH approach. The results for LNS-Auction are average of 20 experiments on each instance.

	Centralized	LNS Auction		CBBA-GLKH	CBBA-AA	Time	
		Average	Minimum			Auction	Ratio
ulysses22	35.3	65.1	46.2	99.0	103.0	0.051	5.13
att48	90.4	134.2	117.7	146.4	213.8	0.008	4.23
eil51	96.7	133.3	122.6	153.3	221.6	0.010	3.11
berlin52	91.3	127.0	119.8	169.7	234.6	0.024	2.32
st70	131.4	175.4	170.4	194.4	302.3	0.010	2.51
eil76	125.1	185.2	165.0	202.7	302.6	0.027	2.10
pr76	128.8	172.3	167.2	198.3	325.2	0.023	2.31
rat99	163.0	238.5	219.3	259.8	404.7	0.045	1.34
kroA100	170.7	246.1	244.3	263.7	429.9	0.021	0.74
kroB100	173.6	244.6	229.0	263.1	417.1	0.045	0.91
eil101	164.6	222.0	178.9	304.6	471.0	0.009	3.01
lin105	150.1	190.7	184.3	232.0	362.8	0.017	0.97
bier127	181.5	299.4	285.1	324.4	490.7	0.013	0.65
ch130	212.9	300.2	286.7	448.7	537.0	0.022	0.77
ch150	234.2	338.9	321.4	365.5	595.7	0.009	0.66
kroA150	239.8	343.4	335.0	364.6	564.3	0.001	0.81

7.4. *TSPLIB Instances*

The TSPLIB [49] provides a large library of TSP instances on which we can test the performance of our algorithm. Table 1 shows the total tour cost of the CBBA implementations, centralized and LNS approaches for a system consisting of seven Dubins vehicles with minimum turning radius of 1. To provide a fair comparison of the tour quality of each auction-based method with the centralized method, we use a complete communication graph ($p = 1$). The experiment includes several medium-size geometric instances from TSPLIB. To be consistent on the ratio of the distances to the minimum turning radius, we scale the task locations in each instance so that they lie in a 10×10 square. The initial location of the seven robots are the first seven locations in each instance. The quality of the tours constructed by LNS-Auction algorithm is compared to the two implementations of the CBBA algorithm, namely CBBA-GLKH and CBBA-AA. The published versions of CBBA, GLKH and LNS-Auction are implemented in MATLAB, `C` and Python, respectively. Although the times are not completely comparable, we provide the time ratio for LNS-Auction to CBBA-GLKH. The total time of the LNS-Auction on **ulysses22** and **kroA150** instances on a Corei5 @2.5Ghz processor are 10.14 and 62.84 seconds, respectively.

7.5. *Heterogeneous Robots*

In this section, we compare our method to a centralized method for heterogeneous systems. The experiments are conducted on uniformly randomly chosen task locations. We consider Dubins and DD vehicle dynamics with unit translational velocity. The DD robot accomplishes a task by visiting the location (x_i, y_i) with free heading, i.e., task i is defined as follows:

$$t_i = \{(x, y, \theta) | (x, y) = (x_i, y_i), \theta \in [0, 2\pi)\}.$$

The Dubins vehicle accomplishes a task by visiting any location in a unit circle centered at the task, i.e., task i is defined as follows:

$$t_i = \{(x, y, \theta) | (x - x_i)^2 + (y - y_i)^2 = R_{\min}^2, \theta \in [0, 2\pi)\}.$$

The minimum turning radius of the Dubins vehicles, $R_{\min}$, and the distance between wheels of the DD vehicle, L, are assumed to be 1. The circular neighborhoods for the Dubins vehicle are discretized with 5 random locations and 5 equally spaced headings at each location. Similarly, the configuration space at the task location for the DD vehicles are discretized with 5 equally spaced headings. Table 2 shows the total tour cost of the centralized and LNS-Auction approaches for different systems of robots and numbers of tasks. Moreover, the experiment shows the ratio of the total time of the LNS-Auction to that of the centralized approach. Each row of the table is a class of 20 random instances with the same problem parameters: that is, the number of tasks N, the number of DD vehicles, and the number of Dubins vehicles. The averages are on 10 runs of the LNS-Auction on instances of each class.

Finally, Figure 5 shows tours for our implementation of the LNS-Auction for a system of three DD vehicles and one Dubins vehicle on 20 task locations.

Table 2. Total tour cost on TSPLIB instances. The LNS-Auction used the communication graph with $p = 1.0$. N in the instance class name denotes the number of tasks, DD denotes the number of DD vehicles and Dubins denotes the number of Dubin vehicles in the system. Auction indicates the fraction of the total LNS-Auction time which is spent for solving I.P. (7), and Ratio indicates the average ratio of time for LNS-Auction to the centralized approach.

Instances	Centralized			LNS Auction			
	Average	Minimum	Time	Average	Minimum	Auction	Ratio
N10DD1Dubins1	32.6	25.5	12.8	36.4	27.1	0.22	0.09
N20DD1Dubins1	44.8	42.2	72.4	55.4	40.0	0.11	0.07
N30DD1Dubins1	57.2	51.4	222.0	70.8	55.6	0.08	0.06
N10DD3Dubins1	32.3	26.3	19.8	36.5	27.9	0.18	0.07
N20DD3Dubins1	44.9	38.5	106.2	50.5	42.4	0.13	0.05
N30DD3Dubins1	58.6	56.3	305.2	64.3	53.9	0.08	0.05

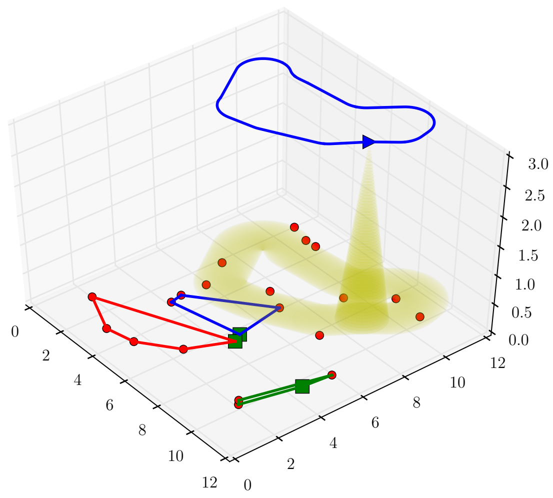

Fig. 5. Tours for a Dubin vehicle with $R_{\min} = 1$ and three DD vehicles with $L = 1$. The Dubins vehicle, shown with a triangle, completes a task by visiting a location in the neighborhood of the task. The neighborhoods are circles with radius 1. The DD vehicles, shown with squares, complete tasks with visiting the location of the tasks with free heading. The 20 task locations are randomly generated in a 10 × 10 square.

8. Conclusion

This paper considered task allocation and sequencing for heterogeneous robots with differential motion constraints. Our approach was based on transforming the problem to a multi-vehicle GTSP. We leveraged the existing insertion heuristics for the GTSP to construct dynamically feasible tours for robots, and gave bounds on their performance. We proposed a novel decentralized implementation of the LNS method for repeatedly removing and reinserting tasks. Our method leverages combinatorial auctions in order to couple task allocation and sequencing. We also proved that the auctions monotonically decrease the total tour length with limited number of bids – a crucial factor in the runtime. For future work, we are pursuing the idea of inserting tasks without explicitly constructing samples of the allowable configurations for each task. Instead, we determine the appropriate robot configuration at the time a task is inserted.

Acknowledgements

This research is partially supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) and through a collaboration with Clearpath Robotics.

Appendix A Appendices

Proof. [Proof of Lemma 4.8] Proof of (ii): Without loss of generality assume that the initial condition is $q_1 = (0, 0, 0)$, $q_2 = (x_2, y_2, \theta_2)$ and $u_r, u_l \in \{-1, 1\}$. Let α be the angle of the line segment connecting the points $(0, 0)$ and (x, y), β be the angle between the final heading and the line segment. The following are two feasible suboptimal paths:

- Rotate by α, translate a distance $\text{Euc}(q_1, q_2)$, and rotate by β; and
- Rotate by $\pi - \alpha$, translate a distance $\text{Euc}(q_1, q_2)$ with reverse gear, and rotate by $\pi - \beta$.

The total rotation in one of the paths, i.e., $\theta + \alpha$ or $2\pi - (\theta + \alpha)$, is less than or equal to π. The time for traveling on the optimal path between $\text{dist}(q_1, q_2) \leq \text{Euc}(q_1, q_2) + \frac{\pi}{2}\frac{L}{r}$.
Proof of (iii):

We begin with showing the feasible paths for the Reed-Shepp's car to change the heading by θ. Without loss of generality we assume that the initial configuration of the car is $q_1 = (0, 0, 0)$ and the destination is $q_2 = (0, 0, \theta)$. For $\theta \in]0, \pi]$, Path$_1$ in Figure A.1 consists of arcs $\overset{\frown}{OM}^+, \overset{\frown}{MN}^-$ and $\overset{\frown}{NK}^+$. Superscripts on the arcs shows the forward and reverse gear motions. Note that the triangles AED and AOD are equilateral. Therefore, we have $\angle ADE = \angle DAE$ and $\angle ODA = \angle DAO$. Thus the following holds in the triangles:

$$\angle AED + \angle EDO + \angle EAO = \angle AOD.$$

Then the distance between the configurations is as follows:

$$\begin{aligned}\text{dist}(q_1, q_2) &= l(\overset{\frown}{OM}) + l(\overset{\frown}{MN}) + l(\overset{\frown}{NK})\\ &= R_{\min}(\angle AED + \angle EDO + \angle EAO)\\ &= R_{\min}(\angle AOD) = R_{\min}\theta.\end{aligned}$$

Path$_2$ in the figure shows the feasible path to reach $q_2' = (0, 0, \theta + \pi)$, consists of arcs $\overset{\frown}{OL}^+, \overset{\frown}{LK}^-$ and $\overset{\frown}{KO}^+$, with length $R_{\min}(\pi - \theta)$. The case for $\theta \in [\pi, 2\pi[$ is similar to Figure A.1. With this result, the path construction procedure for a DD robot also applies to the Reed-Shepps model. Thus, the optimal path between the configurations $q_1 = (0, 0, 0)$ and $q_2 = (x_2, y_2, \theta_2)$ is not greater then $R_{\min}\pi + \text{Euc}(q_1, q_2)$.

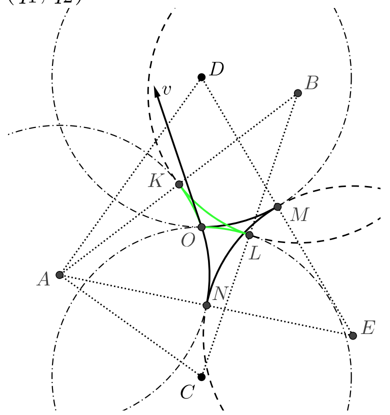

Fig. A.1. Feasible Reeds-Shepp's paths to change the heading at point O by $\theta \in]0, \pi]$ and $\pi + \theta$, Path$_1$ and Path$_2$, respectively. ☐

References

[1] M. Fischetti, J. J. Salazar Gonzalez and P. Toth, A branch-and-cut algorithm for the symmetric generalized traveling salesman problem, *Operations Research* **45**(3) (1997) 378–394.

[2] B. P. Gerkey and M. J. Matarić, A formal analysis and taxonomy of task allocation in multi-robot systems, *The International Journal of Robotics Research* **23**(9) (2004) 939–954.

[3] G. A. Korsah, A. Stentz and M. B. Dias, A comprehensive taxonomy for multi-robot task allocation, *The International Journal of Robotics Research* **32**(12) (2013) 1495–1512.

[4] K. Helsgaun, An effective implementation of the Lin–Kernighan traveling salesman heuristic, *European Journal of Operational Research* **126**(1) (2000) 106–130.

[5] B. Korte and J. Vygen, *Combinatorial Optimization: Theory and Algorithms*, Algorithmics and Combinatorics,, Vol. 21, 4 edn. (Springer-Verlag, 2007).

[6] D. J. Rosenkrantz, R. E. Stearns and P. M. Lewis, II, An analysis of several heuristics for the traveling salesman problem, *SIAM journal on computing* **6**(3) (1977) 563–581.

[7] Z. Tang and U. Ozguner, Motion planning for multitarget surveillance with mobile sensor agents, *IEEE Transactions on Robotics* **21**(5) (2005) 898–908.

[8] K. Savla, E. Frazzoli and F. Bullo, Traveling salesperson problems for the Dubins vehicle, *IEEE Transactions on Automatic Control* **53**(6) (2008) 1378–1391.

[9] J. Le Ny, E. Feron and E. Frazzoli, On the Dubins traveling salesman problem, *IEEE Transactions on Automatic Control* **57**(1) (2012) 265–270.

[10] C. E. Noon and J. C. Bean, An efficient transformation of the generalized traveling salesman problem, *INFOR* **31**(1) (1993) p. 39.

[11] D. Karapetyan and G. Gutin, Lin–Kernighan heuristic adaptations for the generalized traveling salesman problem, *European Journal of Operational Research* **208**(3) (2011) 221–232.

[12] K. Helsgaun, Solving arc routing problems using the lin-kernighan-helsgaun algorithm, tech. rep., Roskilde Universitet (2015).

[13] S. L. Smith and F. Imeson, GLNS: An effective large neighborhood search heuristic for the generalized traveling salesman problem, *Computers & Operations Research* (March 2016) Under Review.

[14] M. Saha, T. Roughgarden, J.-C. Latombe and G. Sánchez-Ante, Planning tours of robotic arms among partitioned goals, *The International Journal of Robotics Research* **25**(3) (2006) 207–223.

[15] J. T. Isaacs and J. P. Hespanha, Dubins traveling salesman problem with neighborhoods: a graph-based approach, *Algorithms* **6**(1) (2013) 84–99.

[16] F. Imeson and S. L. Smith, Multi-robot task planning and sequencing using the SAT-TSP language, *IEEE Int. Conf. on Robotics and Automation*, (2015), pp. 5397 – 5402.

[17] J. J. Enright and E. Frazzoli, The traveling salesman problem for the Reeds-Shepp car and the differential drive robot, *IEEE Conf. on Decision and Control*, (2006), pp. 3058–3064.

[18] P. Oberlin, S. Rathinam and S. Darbha, A transformation for a multiple depot, multiple traveling salesman problem, *American Control Conference*, IEEE (2009), pp. 2636–2641.

[19] M. Turpin, N. Michael and V. Kumar, An approximation algorithm for time optimal multi-robot routing, *Algorithmic Foundations of Robotics XI*, (Springer, 2015), pp. 627–640.

[20] D. Bhadauria, O. Tekdas and V. Isler, Robotic data mules for collecting data over sparse sensor fields, *Journal of Field Robotics* **28**(3) (2011) 388–404.

[21] H.-L. Choi, L. Brunet and J. P. How, Consensus-based decentralized auctions for robust task allocation, *IEEE Transactions on Robotics* **25**(4) (2009) 912–926.

[22] E. Kivelevitch, B. Sharma, N. Ernest, M. Kumar and K. Cohen, A hierarchical market solution to the min–max multiple depots vehicle routing problem, *Unmanned Systems* **2**(01) (2014) 87–100.

[23] M. G. Lagoudakis, E. Markakis, D. Kempe, P. Ke-

skinocak, A. J. Kleywegt, S. Koenig, C. A. Tovey, A. Meyerson and S. Jain, Auction-based multi-robot routing., *Robotics: Science and Systems*, **5** (2005).

[24] L. Liu and D. A. Shell, An anytime assignment algorithm: From local task swapping to global optimality, *Autonomous Robots* **35**(4) (2013) 271–286.

[25] S. L. Smith and F. Bullo, Monotonic target assignment for robotic networks, *IEEE Trans on Automatic Control* **54**(9) (2009) 2042–2057.

[26] S. Sariel and T. Balch, Real time auction based allocation of tasks for multi-robot exploration problem in dynamic environments, *Proceedings of the AAAI-05 Workshop on Integrating Planning into Scheduling*, (2005), pp. 27–33.

[27] M. Berhault, H. Huang, P. Keskinocak, S. Koenig, W. Elmaghraby, P. Griffin and A. Kleywegt, Robot exploration with combinatorial auctions, *IEEE/RSJ International Conference on Intelligent Robots and Systems*, **2**, IEEE (2003), pp. 1957–1962.

[28] L. Lin and Z. Zheng, Combinatorial bids based multi-robot task allocation method, *IEEE International Conference on Robotics and Automation*, IEEE (2005), pp. 1145–1150.

[29] M. G. Lagoudakis, M. Berhault, S. Koenig, P. Keskinocak and A. J. Kleywegt, Simple auctions with performance guarantees for multi-robot task allocation, *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, **1**, IEEE (2004), pp. 698–705.

[30] M. H. Rothkopf, A. Pekeč and R. M. Harstad, Computationally manageable combinational auctions, *Management science* **44**(8) (1998) 1131–1147.

[31] M. Mito and S. Fujita, On heuristics for solving winner determination problem in combinatorial auctions, *Journal of Heuristics* **10**(5) (2004) 507–523.

[32] D. Pisinger and S. Ropke, A general heuristic for vehicle routing problems, *Computers & operations research* **34**(8) (2007) 2403–2435.

[33] S. Ropke and D. Pisinger, An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows, *Transportation science* **40**(4) (2006) 455–472.

[34] E. Demir, T. Bektaş and G. Laporte, An adaptive large neighborhood search heuristic for the pollution-routing problem, *European Journal of Operational Research* **223**(2) (2012) 346–359.

[35] M. Turpin, N. Michael and V. Kumar, CAPT: Concurrent assignment and planning of trajectories for multiple robots, *International Journal of Robotics Research* **33**(1) (2014) 98–112.

[36] B. Chow, Assigning closely spaced targets to multiple autonomous underwater vehicles, Master's thesis, University of Waterloo (2009).

[37] S. Rathinam, R. Sengupta and S. Darbha, A resource allocation algorithm for multivehicle systems with nonholonomic constraints, *Automation Science and Engineering, IEEE Transactions on* **4**(1) (2007) 98–104.

[38] S. S. Ponda, L. B. Johnson, A. Geramifard and J. P. How, Cooperative mission planning for multi-uav teams, *Handbook of Unmanned Aerial Vehicles*, (Springer, 2015), pp. 1447–1490.

[39] B. Korte, J. Vygen, B. Korte and J. Vygen, *Combinatorial optimization* (Springer, 2002).

[40] S. M. LaValle, *Planning algorithms* (Cambridge university press, 2006).

[41] S. Karaman and E. Frazzoli, Sampling-based algorithms for optimal motion planning, *The International Journal of Robotics Research* **30**(7) (2011) 846–894.

[42] L. Janson, E. Schmerling, A. Clark and M. Pavone, Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions, *The International Journal of Robotics Research* **34**(7) (2015) 883–921.

[43] H.-S. Kim and O. Cheong, The cost of bounded curvature, *Computational Geometry* **46**(6) (2013) 648–672.

[44] P. Oberlin, S. Rathinam and S. Darbha, Today's traveling salesman problem, *Robotics & Automation Magazine, IEEE* **17**(4) (2010) 70–77.

[45] N. Mathew, S. L. Smith and S. L. Waslander, Multirobot rendezvous planning for recharging in persistent tasks, *Robotics, IEEE Transactions on* **31**(1) (2015) 128–142.

[46] R. S. Garfinkel and G. L. Nemhauser, The set-partitioning problem: set covering with equality constraints, *Operations Research* **17**(5) (1969) 848–856.

[47] J. Turner, Q. Meng and G. Schaefer, Increasing allocated tasks with a time minimization algorithm for a search and rescue scenario, *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, IEEE (2015), pp. 3401–3407.

[48] D. Di Paola, A. Gasparri, D. Naso and F. L. Lewis, Decentralized dynamic task planning for heterogeneous robotic networks, *Autonomous Robots* **38**(1) (2015) 31–48.

[49] G. Reinelt, TSPLIB – a traveling salesman problem library, *ORSA Journal on Computing* **3**(4) (1991) 376–384.

Armin Sadeghi received B.Sc. degrees in mechanical engineering and aerospace engineering from Sharif University of Technology, Iran, in 2013, and the M.A.Sc. degree in electrical and computer engineering from the University of Waterloo in 2016. From 2014 to 2016, he was Researcher at the University of Waterloo, and worked on the project of task allocation for robots with motion constraints. He is continuing his PhD studies in the Department of Electrical and Computer Engineering at the University of Waterloo.

Armin Sadeghi's research interests include motion planning, optimization and robotics.

Stephen L. Smith received the B.Sc. degree in engineering physics from Queens University, Canada, in 2003, the M.A.Sc. degree in electrical and computer engineering from the University of Toronto, Canada, in 2005, and the Ph.D. degree in mechanical engineering from the University of California, Santa Barbara in 2009.

He is currently an Associate Professor in electrical and computer engineering at the University of Waterloo, Canada. From 2009 to 2011 he was a Postdoctoral Associate in the Computer Science and Artificial Intelligence Lab at the Massachusetts Institute of Technology. His main research interests lie in the control of autonomous systems, with a particular emphasis on robot motion planning, optimization, and distributed coordination. Dr. Smith is a recipient of the 2016 Ontario Early Researcher Award, the NSERC Discovery Accelerator Supplement Award, and the 2015 Outstanding Performance Award from the University of Waterloo.