Distributed Task Allocation and Task Sequencing for Robots with Motion Constraints

by

Armin Sadeghi Yengejeh

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2016

Abstract

This thesis considers two routing and scheduling problems. The first problem is task allocation and sequencing for multiple robots with differential motion constraints. 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, as well as one-in-a-set tasks. Our approach is to transform the problem into a multi-vehicle generalized traveling salesman problem (GTSP). We analyze the GTSP insertion methods presented in [1] and we provide bounds on the performance of the three insertion mechanisms. We then develop a combinatorial-auction-based distributed implementation of the allocation and sequencing algorithm. The number of the bids in a combinatorial auction, a crucial factor in the runtime, is shown to be linear in the size of the tasks. Finally, we present extensive benchmarking results to demonstrate the improvement over existing distributed task allocation methods.

In the second part of this thesis, we address the problem of computing optimal paths through three consecutive points for the curvature-constrained forward moving Dubins vehicle. Given initial and final configurations of the Dubins vehicle, and a midpoint with an unconstrained heading, the objective is to compute the midpoint heading that minimizes the total Dubins path length. We provide a novel geometrical analysis of the optimal path, and establish new properties of the optimal Dubins' path through three points. We then show how our method can be used to quickly refine Dubins TSP tours produced using state-of-the-art techniques. We also provide extensive simulation results showing the improvement of the proposed approach in both runtime and solution quality over the conventional method of uniform discretization of the heading at the mid-point, followed by solving the minimum Dubins path for each discrete heading.

Acknowledgments

I would like to thank my supervisor Professor Stephen Smith and express my appreciations for his patience and guidance through these years.

I would like to thank readers of this – Professor Christopher Nielsen and Professor Soo Jeon – for their time, helpful feedbacks and guidance.

I'd like to thank Frank Imeson and Ahmad Bilal Asghar for their help and insightful discussions.

Finally, I would like to thank my family for their love and support.

Dedication

To my mother, Effat.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 A Distributed Task Allocation and Sequencing Algorithm for Robots with Differential Constraints

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, 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 distributed, 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. Two such heuristics, nearest and cheapest insertion, provide 2-factor approximations to the optimal tour.

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 the goal is to find a tour that visits one city in each set. The GTSP can be solved via a reduction to the TSP, or directly using GTSP solvers [10]. A similar conversion to the GTSP was proposed for planning tours for a robotic arm in [11], for a single Dubins TSP with neighborhoods [12] and for high-level task sequencing problems [13]. Sequencing problems have also been considered for differential drive and Reeds-Shepp models [14].

For multiple robots without dynamics, the centralized problem can be posed as a multi-vehicle TSP. In [15], 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 [16]. In [12], a GTSP approach was proposed for the problem of multiple shortest tours through generalized neighborhoods under dynamic constraints of the Dubins vehicle. In this study, the discrete representation of the configuration space at the neighborhoods is converted to a GTSP.

The distributed problem for multiple robots without dynamics is commonly solved using market-based auctions [17, 18, 19]. When there are an equal number of robots and tasks, the problem is commonly referred to as task assignment [20, 21]. There are two main auction-based approaches: 1) bidding on individual tasks in each auction [22, 23], or 2) bidding on subsets of tasks in each auction, known as combinatorial auctions [24, 25].

The advantage of bidding on subsets of tasks is a faster convergence rate, while the main drawback is the additional computational complexity [17] as the number of subsets grows exponentially with the number of tasks. Moreover, winner determination in the combinatorial auction is shown to be NP-hard [26]. Therefore, there are several techniques to limit the number of subsets [24] and heuristics to approximate the winner determination problem [27]. A successful auction-based approach for allocation and sequencing is the consensus-based bundle algorithm (CBBA) [17], 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.

In the first part of the thesis we focus on distributed task allocation and sequencing for heterogeneous robots with differential motion constraints, for which prior work is limited. The CAPT algorithm [28] provides a solution when the number of tasks equals the number of robots, and thus sequencing is not required. To the best of our knowledge, the existing studies on the distributed task allocation for multiple robots with dynamics constraints propose decoupling the motion constraints from the task allocation problem [29, 30]. These algorithms consist of a task allocation and sequencing phase with assumption of Euclidean distances between the points, followed by a trajectory planning phase that converts the sequence of tasks to feasible tours. Although the decoupling of the task allocation and trajectory planning reduces the complexity, the resulting paths may suffer in the quality [31].

1.2 Optimal Dubins Path Between Three-Consecutive Points

Routing problems for non-holonomic vehicles have been studied extensively in the fields of robotics and autonomous systems [12, 7, 32, 33]. The non-holonomic motion of a forward-moving Dubins vehicle with bounded turning radius [34] is commonly studied as a model for fixed-wing aerial vehicles. A configuration of a Dubins vehicle consists of a location (x, y) in the Euclidean plane and a heading $\alpha \in [0, 2\pi)$. The motion of the Dubins' vehicle with minimum-turning radius $R_{\min}$ and control input $u \in [-1/R_{\min}, 1/R_{\min}]$ is governed by the following equations:

$$\dot{x} = \cos\alpha, \quad \dot{y} = \sin\alpha, \quad \dot{\alpha} = u.$$

Dubins [34] provided the set of candidate optimal paths between pair-wise configurations of the Dubins vehicle.

In this thesis, we focus on the Dubins path problem between three consecutive points, where headings at only the initial and final point are fixed. Our interest in this problem stems from two applications. First, given a Dubins path through a set of points, a fast solution to this problem provides a method for inserting a new point into the Dubins path with minimum additional cost. Second, we show how it can be used as a tool to perform repeated local optimizations on a Dubins path through a set of points.

Related work: Ma *et al.* [35] study the optimal Dubins paths for three consecutive points where the initial heading is fixed and the midpoint and final point have free headings. Under the assumption that the pairwise Euclidean distance between all points is at least $2R_{\min}$, the authors provide a sufficient condition for the optimal path between the points. In addition, a receding horizon algorithm is proposed to construct feasible Dubins path on an ordered set of points.

The authors of [36] formulate a family of convex optimization sub-problems to address the problem of the optimal Dubins path between a set of n ordered points with distance at least $4R_{\min}$ apart. The drawback of the approach is that the number of convex optimization sub-problems can grow to $2^{(n-2)}$ in the worst case. Their approach provides a solution to the three-point Dubins problem, but it requires solving several convex optimization problems.

A heuristic was recently proposed [37] to extend this method to the problem of Dubins paths through neighborhoods.

Another closely related problem is the Dubins TSP, where given n points, the objective is to sequence the points and choose a heading at each point such that the resulting Dubins tour length is minimum. In [8, 30, 38], approximation algorithms are proposed to assign headings to the points given the optimal ordering of the Euclidean TSP problem on the same set of points. In [38], the headings are assigned by a heuristic solution to the three-point Dubins problem considered in this thesis. This heuristic approach is adopted in [39] to insert points into the tours of multiple-Dubins vehicles.

In [9], the continuous interval of headings at each point is approximated by a finite number of samples. Each sample, along with the position of the corresponding point forms a configuration, and the problem reduces to computing a generalized traveling salesman problem (GTSP) tour that visits one configuration for each point. The authors in [40] present an experimental comparison of Dubins TSP algorithms including the GTSP approach. Recently and built on the results for the pairwise optimal Dubins interval path, Manyam and Rathinam [41] proposed a Dubins TSP algorithm based on uniform discretization of the headings at each point to intervals.

In Table 1.1, we provide references to the studies involving Dubins vehicle based on the different features of the problems. The notations in the table are defined as follows:

(i) Multiple-ordered: The problem of planning feasible Dubins vehicle path between multiple points with fixed order;

(ii) D.E.: Abbreviation of Dynamic Environment. In the studies with this notation, the tasks arrive in time;

(iii) N: Tasks are defined as visiting a neighborhoods;

(iv) Polygonal car: Dubins vehicle in a polygonal shape;

(v) (A): Autonomous underwater vehicle(AUV); and

(vi) (d): Distance constraint between the points.

1.3 Thesis Contribution

We analyze two routing problems in this thesis. For the task allocation problem, the main contribution of this thesis is to provide a distributed algorithm for task allocation and sequencing for multiple robots with dynamics. Tasks are defined as subsets of the robot configuration space. A robot completes a task by visiting any point in the subset. Building on prior work [11, 9], we transform the problem into a multi-vehicle GTSP. We provide bounds on the performance of three GTSP insertion methods proposed in [1] that are generalization of TSP insertions [6]. We then provide a distributed implementation of the insertion methods based on large-neighborhood search, and benchmark the performance of the approach on several instances from the TSPLIB [62]. Our approach utilizes an optimization framework called large neighborhood search (LNS) [63], which has been successfully applied to several vehicle routing problems [64, 65]. 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.

The focus of second part of the thesis is to provide an efficient method for computing the optimal Dubins path between three consecutive points. We present a novel analysis of the problem that relies on inversive geometry, and results in a set of equations defining the optimal heading at the mid-point. We provide a simple method to approximate the optimal heading, and give bounds on its worst-case deviation from optimal. We then present an iterative method that is guaranteed to converge to the optimal solution. In simulation, we compare our approach to the uniform discretization method of [9] in both solution quality and computation time. Finally, we show that a Dubins TSP can be solved using a coarse heading discretization followed by repeated heading optimization using our technique to achieve high-quality tours in approximately 8% of the computation time.

1.4 Organization

The thesis is organized as follows. A review of the literature on task allocation problems is presented in Chapter 2. In Chapter 3, we provide mathematical preliminaries and background on combinatorial problems, vehicle models and geometrical properties of circle inversion. Chapter 4 discusses the problem of distributed task allocation in systems of robots with motion constraints. In Chapter 5, we present our method for the optimal Dubins problem between three consecutive points. Finally, the conclusions and our directions for the future research is presented in Chapter 6.

Reference	Points			N	Heading			Dim.		O	Veh.	Other Constraint
	Point-to Point	Multiple-Ordered	Dubins TSP		Fixed	Constrained	Free	2D	3D		Multiple	
[34, 42]	+	-	-	-	+	-	-	+	-	-	-	
[8]	-	+	-	-	-	-	+	+	-	-	-	
[36]	-	+	-	-	-	-	+	+	-	-	-	$4R_{\min}$ (d)
[9, 41, 43]	-	-	+	-	-	-	+	+	-	-	-	
[35]	+	+	-	-	-	-	+	+	-	-	-	$2R_{\min}$ (d)
[40]	-	+	+	-	-	-	+	+	-	-	-	
[44, 45]	+	-	-	-	+	-	-	+	-	-	-	Steady wind
[46]	+	-	-	-	-	+	-	+	-	-	-	
[47, 48, 49]	+	-	-	-	+	-	-	-	+	-	-	
[50]	-	-	+	+	-	-	-	+	+	-	+	D.E.
[15]	-	-	+	-	-	-	+	+	-	-	+	
[51]	+	-	-	-	+	-	-	+	-	+	-	Polygonal car
[52]	+	-	-	-	+	-	-	+	-	-	-	Bounded velocity
[53]	-	-	+	+	-	-	+	+	-	-	-	
[54]	+	-	-	-	-	-	+	+	-	-	+	
[29]	+	+	-	-	-	-	+	+	-	-	+	Underwater current(A)
[55]	-	+	-	-	-	-	+	+	-	-	+	
[56]	-	+	-	+	-	-	+	+	-	-	-	
[57]	-	-	+	-	-	-	+	+	-	-	-	Uncertainty
[58, 59]	-	-	+	-	-	-	+	+	-	-	+	
[60]	-	+	-	+	-	-	+	+	-	+	-	Area Coverage
[61]	+	-	-	-	-	-	+	-	+	-	-	Implementation

Table 1.1: References to the studies involving Dubins vehicles.

Chapter 2

Literature Review

The task allocation problem for a system of agents is extensively studied in the literature. Several variations of this problem are formulated and addressed with various techniques, including exact and approximation methods. Researchers have studied the problems with complex tasks, uncertain and dynamic environments and constraints on the agents. Mainly, following aspects define a task allocation problem:

- Centralized, decentralized or distributed algorithm;
- Objective function, e.g. minimizing the maximum tour cost or the total tour cost;
- Complex or simple tasks;
- Heterogeneous or homogeneous agents;
- Dynamic or static environment;
- Exact, approximation or heuristic methods; and
- Constraint on the agents e.g. capacity, energy and motion constraints.

In this section, we have an emphasis on the studies addressing multi-agent routing problem which is the primary purpose of this thesis. Simple task in the literature is defined as visiting a location in the workspace. Several constraints on the tasks define various complex tasks in the studies in this area, however, the definitions usually include visiting locations.

2.1 Centralized Task Allocation

The centralized task allocation algorithms assume a central unit with complete awareness of the environment and tasks. The central unit plans the trajectories for each agent and broadcasts it to the agents. The centralized approaches have the computational load on a central unit (can be a non-mobile unit) and provide the opportunity to use smaller agents. In the downside, agents are required to remain in the communication range of the centralized unit. As a consequence, these algorithms limit the operation area. Moreover, these methods are susceptible to communication loss and agent failures.

A class of centralized algorithms is based on reducing the task allocation problem to well-studied problems e.g. TSP. Various factors of the problem affect the computational effort required to solve the problem. For instance, the capacity of an agent with limited energy resources indicates the maximum capability of the agent to perform tasks. The capacity is a crucial aspect of the problem in the terms of time complexity. The problem of assigning n tasks with the minimum total cost to m agents ($m > n$) with the capacity of performing one task is optimally solved in polynomial time with a reduction to the matching in bi-parted graphs [66].

In task allocation problem for mobile robots, the cost of performing a task is defined as the time to reach and complete a task. Therefore, the cost depends on the sequencing of the tasks in an agent's path. The problem of assigning tasks with the objective of minimum total cost (i.e. min-sum) is shown to be **NP**-hard and a reduction is given to the traveling salesman problem (TSP) [15, 67]. The reduction based task allocation methods, utilizing the state-of-the-art TSP solvers [4] and the computational capacity of the central unit, offer solutions with good quality for the task allocation in static environments. However, these algorithms suffer in dynamic environments, where the tasks arrive sequentially in time. These algorithms have to generate and solve a new TSP instance, including the new tasks, for each task arrival.

Several studies also consider centralized algorithms without providing reductions to the TSP [68, 69, 70, 71, 16]. These algorithms are polynomial in time with practical or mathematical performance guarantees. Moreover, they offer more flexibility to problems with dynamic environments compared to the reduction based algorithms. The collision-free trajectory assignment is another factor in designing planning algorithms. Two approaches are considered in this matter: 1) Planning paths without considering collisions and relying on the lower-level local planners to avoid obstacles; 2) consider the obstacles in the planning procedure [70, 28]. Turpin *et al.* [28] consider an obstacle-free environment and plan trajectories to avoid collisions between the agents. Assuming capacity of one task for the agents, authors proved optimality for the generated trajectories.

One of the few studies offering a mathematical guarantee for the task allocation problem capturing the minimization of the maximum tour cost (min-max) is a centralized approximation algorithm with a 5-approximation factor [16]–assuming homogeneous agents with no constraint on their capacities.

2.2 Distributed Task Allocation

The centralized algorithms are highly dependent on the communication between the central unit and the agents, however, the distributed [72, 17, 73, 74, 18, 19, 75, 76] methods allocate and sequence the tasks independent of a central unit. We do not distinguish between the terms decentralized and distributed task allocation since there are few subtleties in the definitions of the terms. A brief description of the both terms and the differences is given in [77]. To leverage the drawbacks of depending on a central unit, the decentralized and distributed algorithms distribute the computational load of planning between the agents. As a consequence of eliminating the requirement of connection to a central unit, the operation area becomes larger. The connectivity of each agent to the network of agents at any time is the most common assumption in the decentralized algorithms.

Eliminating a central unit with a complete awareness of the tasks and the assigned agents leaves agents with partial information. In order to avoid conflicts in the assignments – a result of inconsistent awareness of agents on the assignment of the tasks – two methods are proposed in the literature as follows: 1) Assuming a shared memory between the agents [72, 78] 2) reach a consensus on the information between the agents [17, 76].

A set of decentralized algorithms generating sub-optimal assignments are the market-based algorithms [17, 19, 50, 79, 80, 81]. In [19] authors give an auction based algorithm for different objective functions with approximation factors. In a system of m robots and n tasks, the algorithm is a 2-approximation for min-sum objective function and a $2m$-approximation for min-max problems. The Consensus-Based Bundle Algorithm (CBBA) [17] is a decentralized auction algorithm consisting of two phases, namely bidding and consensus. The agents place bids on the tasks with the maximum revenue and spread the bid between other agents with a consensus algorithm. For a communication graph with diameter D, the number of communications required to converge a final assignment is upper-bounded by $\min\{n, mD\}$. Moreover, the final assignment ensures 2-approximation of the total optimal path cost. The auction phase in the CBBA between different pairs of agents take place simultaneously. Later, this algorithm adjusted to have asynchronous auctions, while preserving the mathematical properties [82].

The auction algorithms are also divided based on their bidding mechanisms. In [74], authors characterize the performance of several auction structures based on the number of agents and tasks involving in each auction. In [23], several TSP-based bidding mechanism are proposed, where the agents bid on an individual task. On the other hand, several task allocation algorithms consider bids on subsets of tasks [24, 83]. The advantage of bidding on subsets of tasks is the fast convergence rate of the auctions, however the determination of the winners is the bottleneck. Authors in [84] provide a survey of the literature of the market-based task allocation problem.

Several decentralized and distributed task allocation algorithms are proposed to cover constraints on the tasks. A generalization of the simple tasks are the tasks require visiting a neighborhood [85, 50]. In [86], a distributed algorithm proposed for the assignment of grouped tasks. In [87], authors provide an extension of the decentralized algorithm CBBA to address the assignment problem of tasks with coupled constraints. Authors in [88] consider tasks requiring more than one agent to visit. The constraints in the task allocation problem are not limited to the tasks. In [50], authors assume curvature bounded motion for the agents and propose a decentralized auction-based algorithm.

Chapter 3

Preliminaries

In this chapter we provide some background on the mathematical concepts for the thesis. In Section 3.1 we give definitions from graph theory presented in [89]. Section 3.2 presents definitions of some combinatorial problems. Section 3.3 provide background on standard vehicle models and Section 3.4 presents the preliminaries on the circle inversion.

3.1 Graphs

A graph G is a pair of sets $G = (V, E)$, where $v \in V$ represents a vertex in the graph and $E \subseteq V \times V$ is the set of edges between the vertices. An edge is pointed from u to v if the ordered pair (u, v) is in E. A graph is complete if there exist an edge $(u, v) \in E$ for every $u, v \in V, u \neq v$.

Definition 3.1.1 (Weighted Graph)**.** *A graph G is a weighted graph if there is a function $c : E \rightarrow \mathbb{R}_{>0}$ assigning values to each edge $e \in E$.*

Definition 3.1.2 (Undirected Graph)**.** *A graph G is undirected if for every edge $(u, v) \in E$ there is an edge $(v, u) \in E$.*

Definition 3.1.3 (Subgraph)**.** *Subgraph of a graph G is a graph $G' = (V_{G'}, E_{G'})$ where $V_{G'} \subseteq V$ and $E_{G'} \subseteq E$. A subgraph G' is called a spanning subgraph if $V_{G'} = V_G$.*

Definition 3.1.4 (Path)**.** *A path P in graph G is a subgraph (V_P, E_P) with vertices $\{v_1, v_2, \ldots, v_{k+1}\}$ such that $v_i \neq v_j$ for $1 \leq i < j \leq k+1$ and*

$$E_P = \big\{(v_i, v_{i+1}) | i \in \{1, \ldots, k\}\big\} \subseteq E.$$

Equivalently a path can be represented as a sequence of vertices from v_1 to v_{k+1} connected by edges.

Definition 3.1.5 (Cycle)**.** *A cycle C in graph G is a subgraph (V_C, E_C) with vertices $\{v_1, v_2, \ldots, v_{k+1}\}$ such that $v_i \neq v_j$ for $1 \leq i < j \leq k+1$ and*

$$E_C = \big\{(v_i, v_{i+1}) | i \in \{1, \ldots, k\}\big\} \cup \{(v_{k+1}, v_1)\} \subseteq E.$$

A spanning cycle is called a tour.

Definition 3.1.6 (Connected Undirected Graph)**.** *An undirected graph G is connected if for every pairs of vertices $u, v \in V$ there exist a path in the graph G from u to v.*

Definition 3.1.7 (Tree)**.** *A tree $T = (V_T, E_T)$ in an undirected graph G is a connected subgraph with no cycle.*

Definition 3.1.8 (Minimum Spanning Tree)**.** *A spanning tree T in an undirected graph G is a tree such that $V_T = V$. Minimum spanning tree in a weighted graph, is a spanning tree with minimum total weight, i.e. $\sum_{e \in E_T} c_e$.*

The following section discusses number of combinatorial problems related to graph theory that we will be referring to in the thesis.

3.2 Combinatorial problems

Consider a graph $G = (V, E, c)$ consists of a set of vertices V, a set of edges E and edge costs $c : E \to \mathbb{R}_{>0}$.

The traveling salesman problem (TSP): Given a complete weighted graph G, 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 weighted 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): In the multi-vehicle GTSP, we are 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 goal is to find 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

(i) $V_{T^j} \cap V_{T^k} = \emptyset$ for each $j, k \in \{1, \dots, N_v\}$;

(ii) $| \cup_{j=1}^{N_v} V_{T^j} \cap V_i| = 1$ for each vertex set V_i; and

(iii) $\sum_{j=1}^{N_v} \sum_{e \in E_{Tj}} c(e)$ is minimized.

Combinatorial auction problem: We are given a set $C = \{c_1, \dots, c_m\}$, a set of subsets $\{S_1, \dots, S_n\}$ where each $S_i \subset C$, and non-negative price for each subset $p_i > 0$. Our goal is to find a set of subsets W taken from among $\{S_1, \dots, S_n\}$ that forms a partition of C and has maximum total value, $\sum_{i|S_i \in W} p_i$.

3.3 Vehicle Dynamics

The following are three commonly-used models for vehicle dynamics. For each model, the shortest path between two configurations can be efficiently computed [90].

3.3.1 Dubins vehicle

The model describes a vehicle with bounded turning radius. The equations of motion are

$$\begin{aligned} \dot{x} &= v \cos \theta, \\ \dot{y} &= v \sin \theta, \\ \dot{\theta} &= u. \end{aligned}$$

where $u \in [-v/R_{\min}, v/R_{\min}]$, $(x, y) \in \mathbb{R}^2$, $R_{\min}$ is the minimum turning radius of the vehicle, and v is the constant velocity. The optimal path between any pair of configurations of the Dubins vehicle is limited to 6 possible path types. Let L, R denote the turn to left and right, respectively and S denote the traversing a straight line. Any optimal path is a member of set $\{LRL, RLR, LSL, RSR, RSL, LSR\}$ [34].

3.3.2 Reeds-Shepp car

The Reeds-Shepp's car model 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.

Letting $\omega \in \{-1, 1\}$ be the forward and reverse gears, and $u \in [-v/R_{\min}, v/R_{\min}]$, the model is

$$\begin{aligned}\dot{x} &= \omega \; v \cos\theta,\\ \dot{y} &= \omega \; v \sin\theta,\\ \dot{\theta} &= u \; \omega.\end{aligned}$$

Additional to the motion primitives of Dubins vehicle, the Reed-Shepp car requires a notation of froward or reverse gear.

3.3.3 Differential drive robot (DD):

The DD robot actuates two wheels independently and is capable of changing its heading without translation. Let u_r, u_l be the angular velocities of the right and left wheels, r be the radius of the wheels and L be the distance between the them. Then, the model is

$$\begin{aligned}\dot{x} &= \frac{r}{2}(u_r + u_l)\cos\theta,\\ \dot{y} &= \frac{r}{2}(u_r + u_l)\sin\theta,\\ \dot{\theta} &= \frac{r}{L}(u_r - u_l).\end{aligned}$$

In this thesis, 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. In general we refer to the TSP problem involving a robot with differential constraint as **VTSP** where **V** represents the dynamics of the robot.

3.4 Circle Inversion

In two dimensional geometry, circle inversion [91] is a mapping of a geometric object Q with respect to a circle $\mathcal{C} = \text{circle}(O, R)$ to another object $\text{inv}(Q, \mathcal{C})$. The inverse of points, lines and circles are defined as follows:

Definition 3.4.1 (Inverse of a point)**.** *The inverse of a point P with respect to $\mathcal{C}$ is a point P' on the segment $\overline{OP}$ with distance $\frac{R^2}{|OP|}$ from P.*

Figure 3.1 depicts the geometrical method to find the inverse of point P.

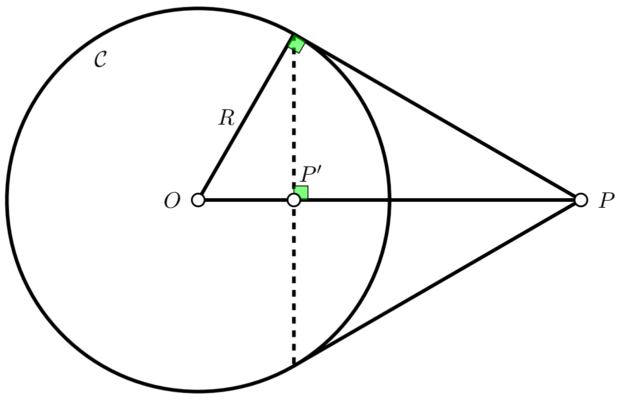


Figure 3.1: The inverse of point P with respect to circle $\mathcal{C} = \text{circle}(O, R)$.

Definition 3.4.2 (Inverse of a line)**.** *The inverse of a line l is obtained by inverse of points, with respect to $\mathcal{C}$, on the line, i.e. $\forall P \in l, \exists P' = \text{inv}(P, \mathcal{C})$ such that $P' \in \text{inv}(l, \mathcal{C})$.*

The inverse of a line is either a line or a circle. Following define the inverse of a line with respect to $\mathcal{C} = \text{circle}(O, R)$.

(i) If $O \in l$, then the inverse of the line is the line itself,

(ii) If $O \notin l$, then the inverse of the line is a circle passing through O.

Definition 3.4.3 (Inverse of a circle)**.** *The inverse of a circle Q is obtained by inverse of points, with respect to $\mathcal{C}$, on the circle, i.e. $\forall P \in Q, \exists P' = \text{inv}(P, \mathcal{C})$ such that $P' \in \text{inv}(Q, \mathcal{C})$.*

The inverse of a circle is either a line or a circle, and the inverse is well defined by inverse of three point on the line. Following conditions define the inverse of a circle with respect to $\mathcal{C} = \text{circle}(O, R)$.

(i) If $O \in Q$ and Q does not intersect $\mathcal{C}$, then the inverse is a line tangent to Q at O,

(ii) If $O \in Q$ and Q intersects $\mathcal{C}$, then the inverse is a line passing through the intersection points of Q and $\mathcal{C}$,

(iii) If $O \notin Q$, then the inverse is a circle passing through O.

With a slight abuse of terminology, we define inverse of a line segment S with respect to $\mathcal{C}$ to be the inverse of the infinite line containing the line segment S with respect to $\mathcal{C}$.

The angle between a circle and an intersecting line is defined as the angle between the line and the tangent to the circle at the intersection point. Following proposition provides a property of the circle inversion on preserving angles between intersecting lines.

Proposition 3.4.4 (Circle inversion preserves angels)**.** *The angle between two intersecting lines, l_1, l_2 equals the angle between the* $\mathrm{inv}(l_1, \mathcal{C})$ *and* $\mathrm{inv}(l_2, \mathcal{C})$*.*

Chapter 4

A Distributed Task Allocation and Sequencing Algorithm for Robots with Differential Constraints

4.1 Problem Formulation and Approach

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

4.1.1 Task Allocation and Sequencing Problem

Consider a group of N_r robots with differential constraints on their motion, located in a planar workspace $X \subset \mathbb{R}^2$. 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 above, $\Theta = \mathbb{S}^1$ 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. 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.

4.1.2 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 select discretized configurations from each task. The set of n discretized configurations from the space $X_i \times \Theta_i$ for task i is denoted by t_i^n. In [9], the Dubins TSP (in which each task i contains a single (x_i, y_i) location) is converted to a GTSP by selecting equally spaced headings θ at the location. Thus, the n discretized configurations of task i are

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

Given N_r robots and N_t tasks, we construct a GTSP as follows. We define a complete weighted 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

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

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 $\overline{V}_i$, $i \in \{1, \ldots, N_r\}$ representing initial state (depot) of the robots. 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 $(\mathbf{x}_v, \theta_v)$. We assume that these edge weights satisfy the triangle inequality. Note that, in the case $X = \mathbb{R}^2$, this assumption holds for time optimal paths of three vehicles models in Section 3.3, due to the fact that the weight of an edge between two vertices is the minimum time to travel between the corresponding configurations. However, the edge weights are non-symmetric for the Dubins vehicle, and thus only the directed triangle inequality holds [9]. In more general workspace X, where the point-to-point paths are obtained by the sampling-based planners [93, 94], we take the metric closure of the graph [92], in which the edge between two vertices is replaced with the shortest path between the vertices in the graph.

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.

Remark 4.1.1 (Computing Edge Weights). The conversion to GTSP relies on efficient computation of optimal point-to-point paths between configurations. In the case $X = \mathbb{R}^2$, each point-to-point path can be computed in constant time for the three vehicle models in Section 3.3, since there exist an only finite number of candidate shortest paths between any two configurations [90]. In a more general workspace X and other vehicle models, close to optimal paths are provided by the sampling based point-to-point planners [93, 94]. •

4.2 An Insertion Heuristic for the GTSP

A class of insertion methods for constructing GTSP tours are presented in [1]. In this section, we apply these methods to construct TSP tours for vehicles with differential motion constraints.

4.2.1 Insertion Methods

The insertion methods presented in [1] are the 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 $T = (V_T, E_T)$, a vertex set V_i such that $V_i \cap V_T = \emptyset$, and a vertex $v \in V_i$. We find the edge $(u, w) \in E_T$ which minimizes the insertion cost, i.e. $\text{cost}(u, v, w) = c(u, v) + c(v, w) - c(u, w)$, and construct a sub-tour, denoted by $\text{TOUR}(T, v)$, 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.,

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

The final tour T_m includes a vertex from each vertex set and the tour is an approximation for 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 between 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:

- *Nearest insertion* inserts the vertex set V_j containing the vertex with the 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 is maximum:
$$\arg\max_{V_j} \min_{v\in V_j, u\in V_T} \{c(u,v)\}.$$

Remark 4.2.1 (A variation of nearest insertion). In this thesis, 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.,
$$\arg\min_{V_j} \min_{v\in V_j, u\in V_T} \min\{c(u,v), c(v,u)\}.$$
•

Remark 4.2.2 (Special Case of TSP Insertions). When each vertex set V_i contains only one vertex and the costs are symmetric, the problem becomes a TSP, and the three insertion mechanisms are those from [6]. It is 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. •

Built on the time complexity analysis in [6] for the TSP insertion methods, the cheapest insertion method for GTSP can be implemented to run in $O(|V|m\log m)$. Moreover, the nearest and farthest insertion methods for GTSP run in $O(|V|m)$.

4.2.2 Bounds on the GTSP Tour Cost

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 4.2.3 (Directed triangle inequality). The edge costs satisfy the directed-triangle inequality, i.e.,

$$c(u, v) + c(v, w) \geq c(u, w) \quad \forall u, v, w \in V.$$

Assumption 4.2.4 (Bounded ratio of edge costs). The edge costs between every pair of the vertices are in constant factor k of each other, i.e.,

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

Assumption 4.2.3 is the directed triangle inequality satisfied by the time optimal paths of the three vehicle models in Section 3.3. In Section 4.2.3, we show that Assumption 4.2.4 also holds for the time optimal paths of the three vehicle models.

Let d be the maximum distance between the vertices in the same vertex set and ϵ be the minimum Euclidean distance between vertices in different vertex sets. Define $\rho = \frac{d}{\epsilon}$ as a parameter capturing the density of the task locations.

Recall the directed-wighted graph $G = (V, E, c)$ from Section 4.2.1, 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 INSERT be the cost of the tour in G constructed by a GTSP insertion heuristic and 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 $(\lceil \log m \rceil + 1)\text{TSP}(G')$. Therefore, the GTSP tour constructed on the graph G with any insertion heuristic is

$$\text{INSERT} \leq k(\lceil \log m \rceil + 1)\text{TSP}(G'). \tag{4.1}$$

Let GTSP* 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 GTSP*. Then we have

$$\mathrm{TSP}(G') \leq \mathrm{GTSP}(S) \leq \mathrm{GTSP}^* + 2md. \tag{4.2}$$

With the definition of the ϵ, we know that GTSP$^* \geq m\epsilon$. Combining this with inequalities (4.1) and (4.2) we have,

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

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

Proposition 4.2.5. *Let* INSERT *be the cost of the tour constructed using either nearest or cheapest insertion. Then*

$$\frac{\mathrm{INSERT}}{\mathrm{GTSP}^*} \leq (1+k)(1+2\rho).$$

Before providing a proof for the proposition, we require establishing a property of the insertions at each step. 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 into tour T_i, between u and w, is defined by

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

Lemma 4.2.6 (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 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 4.2.4 and 4.2.3 we have,

$$\begin{aligned} \mathrm{cost}(T_i, s_i) \leq \mathrm{cost}(T_i, v_i) &\leq c(u, v_i) + c(v_i, u) \\ &\leq (1+k)\min\{c(u, v_i), c(v_i, u)\}. \end{aligned} \tag{4.3}$$

Since, v is the closest vertex from the tour, then the following inequality follows from inequality (4.3),

$$\min\{c(u, v_i), c(v_i, u)\} \leq c'(p,q) \ \forall p \in V_{T_i}, q \in S \setminus V_{T_i}. \tag{4.4}$$

From Equations (4.3) and (4.4), we conclude the proof. □

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 $\text{cost}(T_i, s_i)$, thus the bound on the insertion cost in Lemma 4.2.6 holds for the cheapest insertion.

Now we can establish the approximation factors in Proposition 4.2.5 for the nearest and cheapest insertions.

Proof of Proposition 4.2.5. Lemma 3 in [6] states that if the insertion cost at each step is less the $(1+k)c'(p,q)\ \forall p \in V_{T_i}, q \in S \setminus V_{T_i}$, then the cost of the constructed tour is less than the minimum spanning tree $\text{MST}(G')$ on the vertices of the graph G'. Therefore, from Lemma 4.2.6 and inequality (4.2) we have,

$$\begin{aligned}\text{INSERT} \leq (1+k)\text{MST}(G') &\leq (1+k)\text{TSP}(G') \\ &\leq (1+k)(1+2\rho)\text{GTSP}^*.\end{aligned}$$

□

4.2.3 Bound on Vehicle Tour Cost

The following result gives an upper-bound on the path cost between two configurations for each of the three vehicle models in Section 3.3. The Dubin's bound was conjectured in [8] and established in [95], while the Reeds-Shepp and DD bounds are, to the best of our knowledge, new results.

Lemma 4.2.7 (Distance of robot configurations)**.** *Consider a robot whose dynamics are governed by one of the three vehicle models (Dubins, Reeds-Shepp, DD) and two robot configurations q_1 and q_2. Then, the travel time from q_1 to q_2, denoted* $\text{dist}(q_1, q_2)$ *satisfies*

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

where Euc *is the 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) $\mathcal{C} = \frac{7\pi}{3}R_{\min}$ for Dubins;

(ii) $\mathcal{C} = \frac{\pi}{2}\frac{L}{r}$ for Differential Drive; and

(iii) $\mathcal{C} = \pi R_{\min}$ for Reeds-Shepp.

Proof. The proof of (1) is established in [95]. The proof of (2), (3) is given in Appendix A.1. □

To compare the performance bound on the insertion methods to the existing approximation in Lemma 4.2.7 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 $\rho = \frac{\mathcal{C}}{\epsilon}$. Thus we arrive the following result,

For the case of a Dubins vehicle, the distance bounds from [95] 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 bound in (4.1) becomes

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

Let VTSP be the optimal tour cost for a vehicle model in 3.3 in a workspace $X = \mathbb{R}^2$, then we have

Corollary 4.2.8. *The total tour cost constructed by the nearest or cheapest insertion is bounded by*

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

Proof. From Lemma 4.2.7 and inequality (4.3), the cost of inserting any vertex in the tour at step i by the nearest and cheapest insertion is

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

From Lemma 3 in [6] we have,

$$\mathrm{INSERT} = \sum_{i=1}^{m} \mathrm{cost}(T_i, s_i) \leq 2(1 + \rho)\mathrm{MST}(G'). \tag{4.5}$$

The $\mathrm{MST}(G')$ and $\mathrm{TSP}(G')$ 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:

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

□

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

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

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.

Remark 4.2.9 (LNS via Repeated Insertions). In practice, one can perform repeated rounds of insertions to improve the tour. An initial insertion method is chosen and a tour is constructed. A subset of vertices on the tour are deleted and then reinserted into the tour using a randomly chosen insertion method. This procedure is repeated, accepting tours when they pass an acceptance criterion (for example, if the new tour has smaller cost). This is the basic idea of large neighborhood search (LNS) [63]. •

4.2.4 Insertion Heuristic for Multiple Robots

In this section, we extend the approximation factor of the insertion methods for a single robot in Section 4.2.3 to a system of multiple robots. The high-level idea is to reduce the mGTSP problem to a GTSP problem and construct a tour via the insertion methods.

In [67], a reduction is given from the problem of multiple shortest cost tours for heterogeneous robots with dynamic constraints to TSP. The reduction creates a GTSP instance consisting of the duplicates of tasks for each robot and a duplicate of each depot. The reduction is followed by a reduction from the GTSP instance to TSP [96]. Under assumption that the configurations for robots at the depots are fixed, the reduction converts the GTSP graph in Section 4.1.2 with N_r robots and V as tasks to a GTSP instance with $N_r(|V|+2)$ vertices. A reduction of the mTSP problem with homogeneous robots to the TSP is presented in [15]. The reduction only creates a duplicate of the depots, therefore, the size of the graph with N_r robots and V as tasks become $|V| + 2N_r$.

The reduction from mTSP to TSP in [15] for a system of homogeneous robots can be extended to the mGTSP problem without adding the duplicates for a system of multiple homogeneous robots. We provide the approximation factors for the insertion methods in the reduced GTSP instance and then we extend the results to the heterogeneous robots.

Let $\overline{V} = \{\overline{V}_1, \ldots, \overline{V}_{N_r}\}$ denote the set of depots for the robots. Each $\overline{V}_i$ is a vertex set consisting of the samples in the configuration space at the depot location of robot i. Also, the replicas of the vertices at the depots are defined in the set $\overline{V}' = \{\overline{V}'_1, \ldots, \overline{V}'_{N_r}\}$. Consider a dummy vertex with zero edge cost to the depots and edges of cost M to the vertices in V, where M is a large number equal to

$$|N_r||V| \max_{v_i, v_j \in V}\{c(v_i, v_j)\} \max_{v_i \in V, v_j \in \overline{V}}\{c(v_i, v_j\}\}.$$

Recall from Section 4.2.3 that the vertices in the graph represent configurations at the task locations. Let dist^r be a cost function which takes an edge (u, v) as an input and returns the minimum required time for robot r to traverse the path between two configurations associated with the vertices u and v. In a homogeneous system of robots the cost function dist^r is identical for the robots and it is denoted by dist.

The reduction of the multi-robot task allocation to the single GTSP problem is as follows:

Consider a complete weighted graph $G = (W, E, c)$ where

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

We set the edge cost $c(u, v),\ u, v \in W$ to M with the exception of the following cases:

(i) $\text{dist}(u, v)$ if $u, v \in V$;

(ii) $\text{dist}(u, v)$ if $u \in \overline{V}_i,\ i \in \{1, \ldots, N_r\}$ and $v \in V$;

(iii) $\text{dist}(u, v)$ if $u \in V$, $v \in \overline{V}'_i$ and $i \in \{1, \ldots, N_r\}$;

(iv) zero if $u = \text{dummy}$ and $v \in \overline{V}_1$;

(v) zero if $v = \text{dummy}$, $u \in \overline{V}'_i$ and $i \in \{1, \ldots, N_r\}$;

(vi) zero if $u \in \overline{V}'_i$, $v \in \overline{V}_{i+1}$ and $i \in \{1, \ldots, N_r - 1\}$; and finally

(vii) zero if $u \in \overline{V}_i$, $v \in \overline{V}'_i$ and $i \in \{1, \ldots, N_r\}$.

Let INSERT 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 $c'(u, v)$ where $u, v \in S$ is set to M with the exception of the following cases:

(i) $\mathrm{Euc}(\mathbf{x}_u, \mathbf{x}_v)$ if $u, v \in V$;

(ii) $\mathrm{Euc}(\mathbf{x}_u, \mathbf{x}_v)$ if $u \in \overline{V}$ and $v \in V$;

(iii) $\mathrm{Euc}(\mathbf{x}_u, \mathbf{x}_v)$ if $u \in V$, $v \in \overline{V}'_i$ and $i \in \{1, \ldots, N_r\}$;

(iv) zero if $u =$ dummy, $v \in \overline{V}_1$;

(v) zero if $v =$ dummy, $u \in \overline{V}'_i$ and $i \in \{1, \ldots, N_r\}$;

(vi) zero if $u \in \overline{V}'_i$, $v \in \overline{V}_{i+1}$ and $i \in \{1, \ldots, N_r - 1\}$; and finally

(vii) zero if $u \in \overline{V}_i$, $v \in \overline{V}'_i$ and $i \in \{1, \ldots, N_r\}$.

Before providing the bounds on the tours constructed by the nearest and cheapest insertion methods for multiple robots we require to establish the following result. Let mVTSP be the minimum total cost of the tours for multiple robots with differential constraints.

Lemma 4.2.10 (Lower bound on multiple optimal tours)**.** *The minimum spanning tree* MST(G') *on the graph G' is at most equal to* mVTSP*.*

Proof. Deleting an edge from each optimal tour result in a set of trees which collectively visit all the depots and the tasks. Connecting these trees with zero costs edges to a dummy vertex creates a spanning tree on the optimal configurations. Note that for every edge (u, v) in the optimal tours, the cost $\mathrm{dist}(u, v)$ is lower-bounded by the $\mathrm{Euc}(u, v)$. Therefore, MST(G') is at most equal to the cost of the spanning tree on the optimal configurations. The constructed spanning tree on the optimal configurations, via deleting non-zero cost edges from the tours and adding zero cost edges, is a lower bound for the total cost of the optimal tours. □

Theorem 4.2.11 (Homogeneous robots)**.** *The total tour cost constructed by the nearest or cheapest insertion is bounded by*

$$\mathrm{INSERT} \le 2(1 + \rho)\mathrm{mVTSP}.$$

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 selects a configuration at the first depot $u \in \overline{V}_1$ with cost zero from the dummy vertex. The closest vertex set to the tour in the next step is the replica of the previously selected depot with cost zero. Without loss of generality, assume that the nearest insertion method selects the exact replica of u in $\in \overline{V}_i'$, namely u', and inserts u' in the tour (recall $c(u, u') = 0$). Inserting the exact replica of u ensures that the tour is complete with equal starting and destination configurations.

The nearest insertion continues to insert a vertex in each depot and again without loss of generality inserts the exact replica of the vertex afterward. Due to the fact that 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 $u' \in \overline{V}_i'$ in the sub-tour is adjacent to a vertex in $\overline{V}_{i+1}$. 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 $\overline{V}_i'$ and $\overline{V}_{i+1}$ 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 $\overline{V}_i$ and $\overline{V}_i', i \in \{1, \dots, N_r\}$.

Recall from Section 4.2.3 that for each insertion of the nearest and cheapest insertions for the graph G we have,

$$\text{cost}(T_i, s_i) \leq 2(1 + \rho)\text{Euc}(p, q) \ \forall p \in T_i, q \in S \setminus T_i. \tag{4.6}$$

By Lemma 3 in [6], the inequality (4.6) is sufficient to show that the sum of the right-hand side of Equation (4.6) is INSERT, and the sum of the left-hand side is the minimum-spanning tree on G'. Therefore, from Lemma 4.2.10 we and inequality (4.6) we conclude the bound on INSERT. □

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.

For a system of multiple heterogeneous robots, the configurations at each task location and the time to traverse the edge between two configurations varies for different robots. Therefore, a vertex set V_j is defined at each task location for robot r consisting of the configurations of robot r, namely V_j^r.

By initializing the tours $\{T^1, \ldots, T^{N_r}\}$ with a sample at the depots, the nearest and cheapest insertion are redefined as follows:

- *Nearest insertion*: inserts the vertex set V_j^r containing the vertex with the minimum distance to or from the tour T^r and deletes all other duplicates of the vertex set V_j.

$$\arg\min_{V_j^r, T^r} \min_{v \in V_j^r, u \in V_{T^r}} \min\{\text{dist}^r(u, v), \text{dist}^r(v, u)\}.$$

- *Cheapest insertion*: inserts the vertex set V_j^r containing the vertex with minimum insertion cost in the tour T^r and deletes all other duplicates of the vertex set V_j.

$$\arg\min_{V_j^r, T^r} \min_{v \in V_j^r, (u,w) \in E_{T^r}} \{\text{dist}^r(u, v) + \text{dist}^r(v, w) \\ - \text{dist}^r(u, w)\}.$$

- *Farthest insertion*: inserts the vertex set V_j^r whose closest vertex has the maximum distance from the tour T^r and deletes all other duplicates of the vertex set V_j.

$$\arg\max_{V_j^r, T^r} \min_{v \in V_j^r, u \in V_{T^r}} \{\text{dist}(u, v)\}.$$

Note that at each step of inserting vertices in the tours, the insertion methods only insert one of the duplicates of the vertex set V_j.

In Section 4.2.3, we defined ρ for three vehicle dynamics. Let $\rho_{\max}$ be the maximum ρ in the set of multiple heterogeneous robots.

Corollary 4.2.12 (Heterogeneous robots). *The total tour cost constructed by the nearest or cheapest insertion is bounded by*

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

Proof. Suppose that the insertion methods, in a step of insertions, insert vertex $s_j \in V_j^r$ to the tour T^r. Also, by the definition of the nearest insertion, the distance of vertex set V_j^r to the tour T^r is minimum between all vertex sets to the tours. Therefore, the insertion cost is limited by the

$$\text{cost}(T^r, s_j) \leq 2(1 + \rho_{\max})\text{Euc}(p, q), \forall p \in \bigcup_{i=1}^{N_r} V^i(T^i), \forall q \in \bigcup_{i=1}^{N_r} V^i \setminus V^i(T^i). \tag{4.7}$$

The summation of the insertion costs on the right-hand side of Equation (4.7) is the total tour cost of the robots. Note that summation of the Euclidean distances on the left-hand side is equal to the cost of the minimum spanning forest rooted at the depots. The minimum spanning forest is a lower bound on the optimal total tour cost. □

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

4.3 A Distributed Auction-Based Algorithm

We now present a distributed algorithm for task allocation and sequencing. The algorithm leverages the GTSP insertion mechanisms presented in Section 4.2 in a manner similar to large neighborhood search (LNS) [63]. The high-level idea 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 sequentially deletes a set of tasks 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.

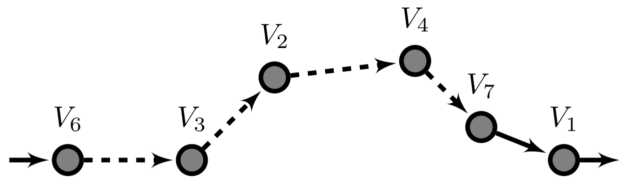


Figure 4.1: Largest subset of $S_4 = \{V_1, V_2, V_3, V_4\}$ containing V_4 that forms a segment on the tour is $\{V_3, V_2, V_4\}$. The insertion cost of $\{V_3, V_2, V_4\}$ is the difference between the cost of the segment shown in dashed arrows and the cost of the edge from V_6 to V_7.

(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) Auctioneer communicates the result of the auction to the winner agents.

(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 ith the 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 4.1).

At Step (iii) robots generate subsets to bid given their current tour and the set of tasks from the auctioneer. Given 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 tours. The set $S_i^r, i \in \{1, 2, \dots, d\}$ is the set of tasks inserted in the tour of robot r in the first i steps of insertions. 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 in step i of insertions. The bidding mechanism is closely related to the auctioneer's 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 of 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 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 summation 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:

$$\begin{aligned}
&\text{minimize} \sum_{i,r} \texttt{bid-value}_i^r \cdot x_i^r \\
&\text{subject to} \\
&\sum_{i,r} x_i^r = 1 \quad \text{for all } i \in \{1,\dots,d\},\, r \in \mathcal{I}_r, \\
&x_i^r \in \{0,1\} \quad \text{for all } i \in \{1,\dots,d\},\, r \in \mathcal{I}_r.
\end{aligned} \tag{4.8}$$

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 [97] which results solving problems with large sizes. In addition we prove that the number of the variables is linear in the size of the problem.

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.

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 4.3.1 (Auction algorithm properties)**.** *The distributed auction algorithm has the following properties:*

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

(ii) Number of the variables in the IP (4.8) for a set of m robots is not greater than $\min\{mD, 2^D - 1\}$.

(iii) Total tour cost monotonically decreases after each auction.

Proof. Proof of (i): At each step of the 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 (4.8) 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 (4.8) and z be the objective value of assigning all tasks to the auctioneer. Thus, the total path tour is

$$\sum \text{cost}(T^r \oplus \text{win}(r)) - \sum \text{cost}(T^r) = z^* - z. \tag{4.9}$$

Note that assigning all the tasks to the auctioneer is a feasible solution to (4.8). As a consequence, the right-hand side for (4.9) is not greater than zero. □

4.3.1 Variations of LNS-Auction

The auction procedure in Section 4.3 and the integer programming formulation (4.8) can be extended to cover additional properties.

Capacity Constrained Robots

The auction algorithm in Section 4.3 considers the case without binding on the capacity of the robots. The capacity is defined as the maximum number of tasks assigned to a robot. The proposed auction algorithm in Section 4.3 solves the task allocation problem with capacity-constrained robots with adjustment in the IP formulation.

Let L^r_{max} be the capacity of Robot r, where $\sum_r L^r_{\text{max}} \geq N_t$. Assuming that the initial assignment of the tasks satisfy the capacity constraint, therefore adding the following set of inequalities to the constraints of IP (4.8) ensures that the solution of IP for auctions satisfy the capacity constraints.

$$L^r + \sum_i |\texttt{bid-set}_i| x^r_i \leq L^r_{\text{max}} \quad \text{for all } r \in \mathcal{I}_r. \tag{4.10}$$

The initial assignment of the tasks are assumed to satisfy the capacity constraint, therefore, the assignment of tasks before the auction is a feasible solution to the IP (4.8) augmented by the set of constraints (4.10).

Multiple Auctioneers

The auction algorithm in Section 4.3 is a distributed implementation of Large neighborhood search. In the large neighborhood search algorithm, deleting and re-inserting tasks from different tours result in searching in a larger subset of the solution space and usually converges faster. Deleting tasks from multiple robots require multiple auctioneers. Although the auction structure remains the same with an auctioneer solving the IP (4.8), there is a subtle but important point to ensure monotonicity of the auction algorithm with multiple-auctioneers. In this implementation, similar to the original algorithm, one can show the monotonicity of the algorithm, ensuring that the assignment of tasks before the auction is a feasible solution to the constraints at the formulation (4.8). We adjust the original LNS-Auction algorithm as follows:

Let $A = \{r_1, \ldots, r_a\}$ be the subset of robots taking the auctioneer role and T^r be the tasks in the path of robot r prior to an auction. Let $T^r_{\text{del}}, r \in A$ be the set of tasks in auctioneer r's tour after deletion. In our adjustment of the LNS-Auction to cover multiple auctioneers, the bidding procedure for bidders is similar to the original approach. However, the subtlety appears on the bids submitted by the auctioneers. Non-auctioneer robots generate $\texttt{bid-set}_i$ similar to the procedure detailed in Section 4.3. The `bid-set` is the largest subset of S_i containing V_i that forms a continuous segment of T^r_i. However, the `bid-value`$_i$ is the total of the insertion cost of the segment in T_i^r and cost(T^r). Note that the auctioneers generate the bids by inserting the tasks to T_{del}^r. Let $T_{i,\text{del}}^r$ be the tour of auctioneer r after i steps of insertions.

(i) Each auctioneer $r \in A$ submits a bid on its path prior to the auction, where `bid-set` is the tasks in the T^r and `bid-value` is cost(T^r).

(ii) Each auctioneer deletes a set of task from its path and offers to other robots in the auction.

(iii) Auctioneer r inserts the tasks offered for the auction.

(iv) The `bid-set` is the union of the largest subset of S_i containing V_i that forms a continuous segment of T_{del}^r and the tasks in T_{del}^r.

(v) The `bid-value`$_i$, is the total of the insertion cost of the segment in $T_{i,\text{del}}^r$ and cost(T_{del}^r).

The adjustment (i) ensures that the assignment of the tasks prior to the auction is a feasible solution to the auction. Therefore, the monotonicity of the algorithm is preserved. Note that without applying (v), the assignment in (i) is the only feasible solution to IP (4.8).

4.4 Simulation Results

In this section, we compare our distributed algorithm to two different implementations of the Consensus-Based Bundle Algorithm (CBBA)[1] from [17], Multi-Vehicle Algorithm (MVA) [30] and a centralized algorithm [67]. The CBBA algorithm plays the benchmarking role in recent studies as the state of the art distributed task allocation algorithm [98, 99]. The first implementation, named CBBA-AA, 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 a 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}

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

to $T_{i(j+1)}$, otherwise the heading is equal to the heading assigned to the previous point in the tour.

In the second implementation of the CBBA, namely CBBA-GLKH, we linked the CBBA and the state of the art GTSP solver– GLKH [100]. After assigning the tasks by CBBA, the set of tasks assigned to each robot is converted to a GTSP instance by sampling the headings at each task location and solved separately by GLKH.

The MVA algorithm presented in [30], considers decoupling the motions of the robots from the assignment problem. After tasks assignment and sequencing by the distributed variation of the Prim's algorithm, a heuristic method is proposed to construct feasible Dubins tours on the sequenced tasks. Assuming $2R_{\min}$ distance between each pair of tasks and starting from a configuration at a depot, the algorithm plans the optimal path to the next task in the tour with a free heading. 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 that point distances are smaller than $2R_{\min}$. The exhaustive search method searches all the paths to the next point with a fine discretization of the headings (0.1 degree) and returns the Dubins path with the minimum cost.

The results of the proposed algorithm are also compared to centralized algorithm [67]. The problem of task allocation and sequencing for multiple Dubins vehicles is transformed to GTSP and solved by the state-of-the-art GTSP solver.

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.

4.4.1 Random Instances

Figure 4.2 compares the total tour cost of the LNS-Auction to CBBA-LKH and the centralized method for different random instances. Due to the fact that the published implementation of the CBBA constructs paths, after the tasks assignment by the CBBA, we perform a post-processing on the paths by the state of the art TSP solver, namely LKH, in order to improve the quality of the tours. In each instance, tasks are uniformly randomly distributed in a 10×10 square. The communication graphs are generated for each $p \in \{0.4, 1\}$. 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 4.2 shows that the LNS-Auction algorithm for

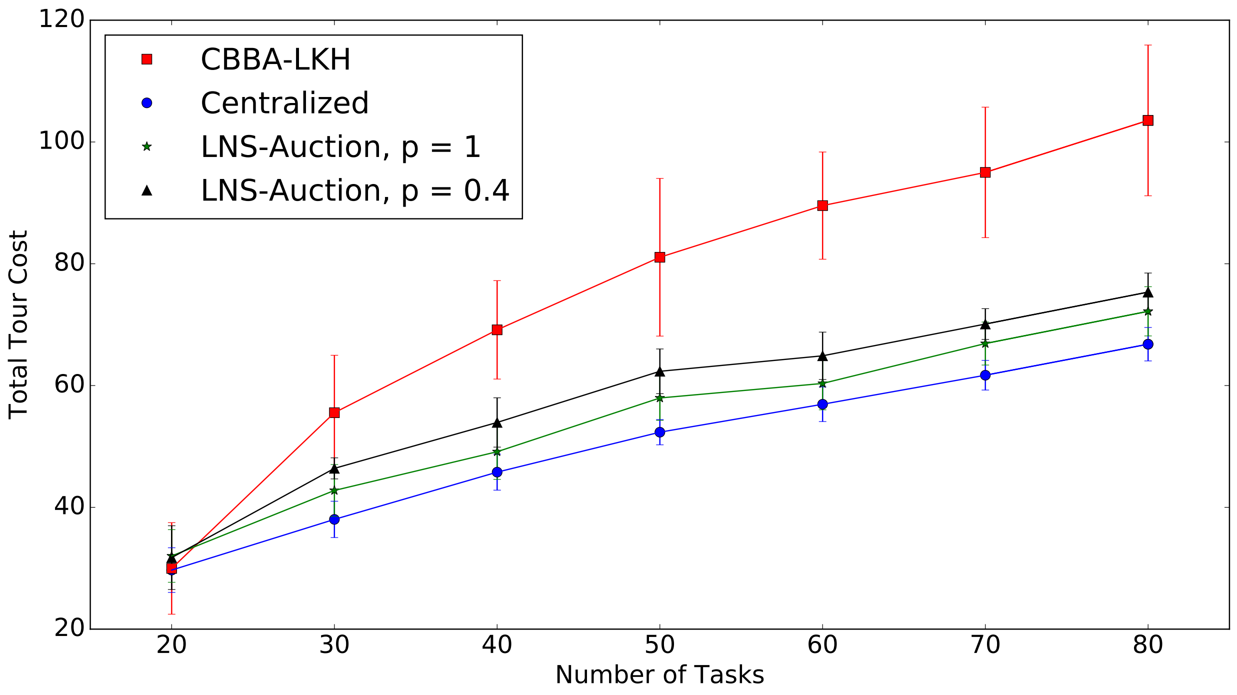


Figure 4.2: Total tour cost versus the number of tasks for seven agents on uniformly randomly generated instances in a 10 × 10 square environment. The robots do not have a differential constraint on their motion $\rho = 0$. For a different number of tasks, 30 random instances are generated. The reported 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.

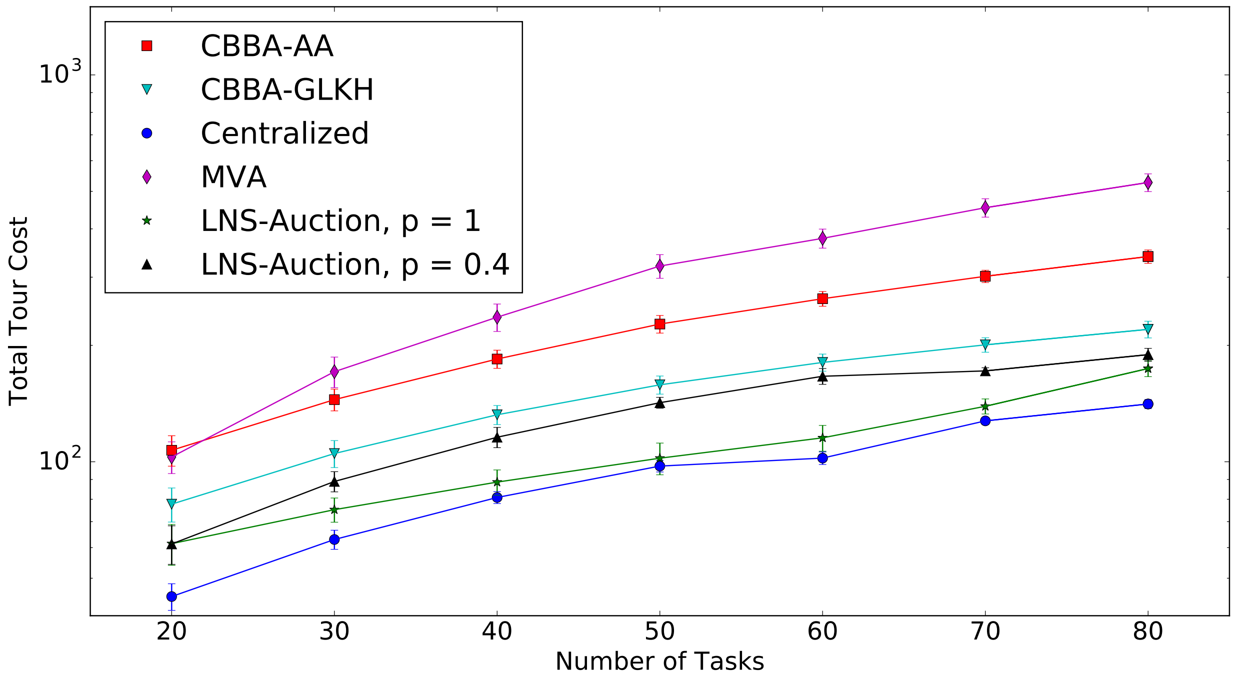
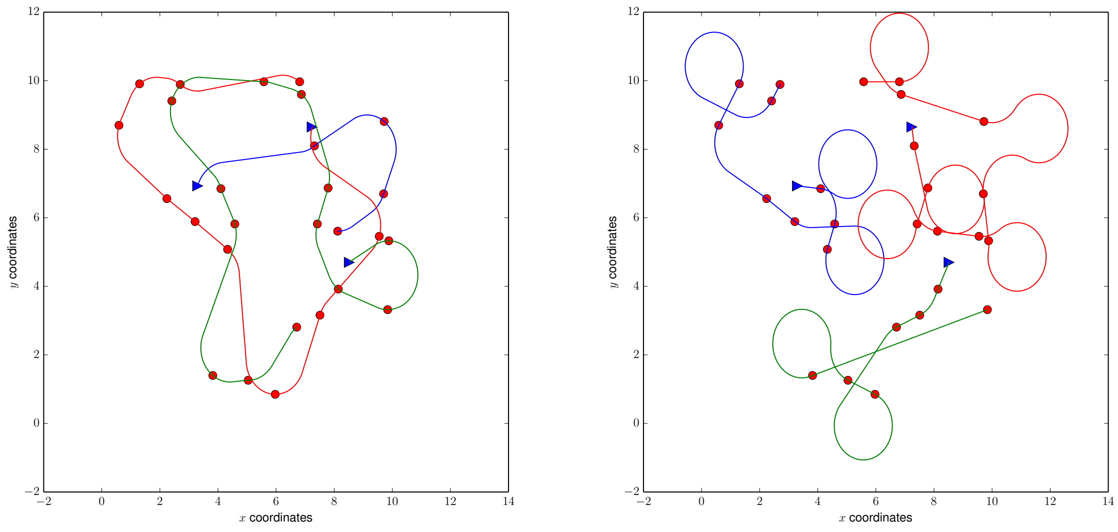


Figure 4.3: Total tour cost versus the number of tasks for seven Dubins' cars with turning radius of 1 on uniformly randomly generated instances in a 10 × 10 square environment. For a different number of tasks, 30 random instances are generated. The reported tour cost for each number of tasks is the average of 30 instances, each solved 10 times. The error bars represent the standard deviation of the results of different runs and instances.

different communication graphs gives considerable improvements in tour quality compared to CBBA-LKH.

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 4.3 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%.

Finally, Figure 4.4 shows paths for the CBBA-AA and our implementation of LNS heuristics for a system of three Dubins vehicle on 30 task locations.



(a) LNS-Auction (b) CBBA-AA

Figure 4.4: Paths for three Dubins vehicle with $R_{\min} = 1$. The 30 task locations are randomly generated in a 10×10 square.

4.4.2 TSPLIB Instances

The TSPLIB [62] provides a large library of TSP instances on which we can test the performance of our algorithm. Table 4.1 shows the total path cost of the greedy and LNS approaches for a system consisting of seven Dubins vehicles with minimum turning radius of 1. Moreover, the experiment shows the ratio of the total auction time of the LNS-Auction to that of solving the Euclidean problem with CBBA. 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.

4.5 ROS Implementation

The Robot operating system (ROS) [101] is a communication tool and protocol–widely used in Robotics. Figure 4.5 demonstrates the high-level implementation of our auction algorithm for two agents (robots). The consensus algorithm, named `server`, randomly assigns the tasks to different agents. Agent 1 (`agent_01`) in Figure 4.5 takes the auctioneer role and requests participation of other agents in the auction. Agent 2 (`agent_02`) provides the service `agent_to_agent` for auctioneers to invite it to auctions. The message in this request (`msg/tsk_list`) is the set of tasks that the auctioneer has deleted from its path and offers for the auction. Table 4.2 shows the inputs and output parameters of the auction service call. The task and bid lists are arrays of objects `Task.msg` and `Bid.msg` detailed in tables 4.3 and 4.4. Agent 2 in a response to the auction request from the auctioneer returns its availability for the auction. The agent accepts all the auction request calls unless it is involved in another auction. To avoid assigning tasks more than the capacity of robot r, the agent also returns the remaining capacity of itself, namely $L^r_{\max} - L^r$ in Section 4.3.1.

Agent 1 calls the service provided by Agent 2 to return the results of the assignment, named `update_path`. After inserting the tasks in the path of Agent 2, the local planner generates the paths towards the next goal.

	Centralized	LNS Auc.		CBBA-GLKH	CBBA-AA	Time	
		Ave.	min			Auc.	Ratio
ulysses22	35.3	**65.1**	46.2	99.0	103.0	0.32	5.13
att48	90.4	**134.2**	117.7	146.4	213.8	0.20	4.23
eil51	96.7	**133.3**	122.6	153.3	221.6	2.21	3.11
berlin52	91.3	**127.0**	119.8	169.7	234.6	0.40	2.32
st70	131.4	**175.4**	170.4	194.4	302.3	0.34	2.51
eil76	125.1	**185.2**	165.0	202.7	302.6	0.47	2.10
pr76	128.8	**172.3**	167.2	198.3	325.2	0.98	2.31
rat99	163.0	**238.5**	219.3	259.8	404.7	1.14	1.34
kroA100	170.7	**246.1**	244.3	263.7	429.9	0.47	0.74
kroB100	173.6	**244.6**	229.0	263.1	417.1	1.25	0.91
eil101	164.6	**222.0**	178.9	304.6	471.0	0.77	3.01
lin105	150.1	**190.7**	184.3	232.0	362.8	0.57	0.97
bier127	181.5	**299.4**	285.1	324.4	490.7	0.60	0.65
ch130	212.9	**300.2**	286.7	448.7	537.0	1.09	0.77
ch150	234.2	**338.9**	321.4	365.5	595.7	0.65	0.66
kroA150	239.8	**343.4**	335.0	364.6	564.3	0.92	0.81

Table 4.1: Total tour cost on TSPLIB instances. The LNS-Auction used the communication graph with $p = 1.0$. Auc. denotes the total time spent for I.P. (4.8) averaged on different trials and Ratio indicates the average time ratio of the LNS-Auction to CBBA-LKH. The results for LNS-Auction are average of 20 experiments on each instance.

Name	Input	Output

`auction_request`	`Task`	`bool Availability`
		`Bid`
		`int16 agent_id`
		`int16 capacity`

Table 4.2: The table shows the content of an agent to agent service call message.

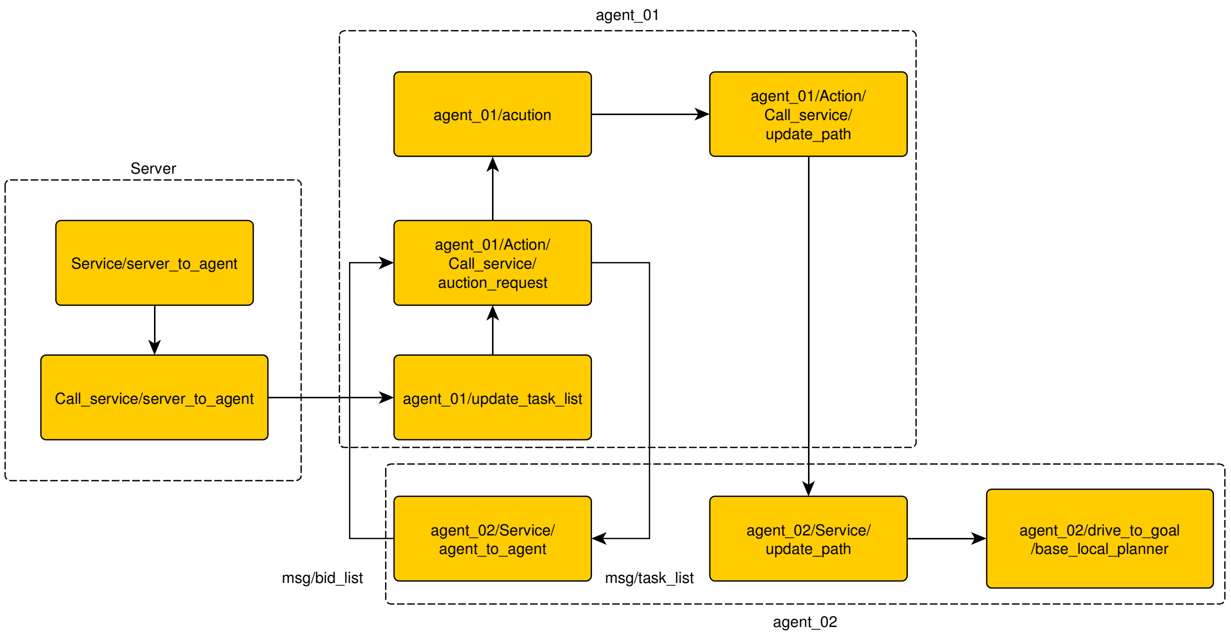

agent_01
Server
agent_01/acution
agent_01/Action/ Call_service/ update_path
Service/server_to_agent
agent_01/Action/ Call_service/ auction_request
Call_service/server_to_agent
agent_01/update_task_list
agent_02/Service/ agent_to_agent
agent_02/Service/ update_path
agent_02/drive_to_goal /base_local_planner
msg/bid_list
msg/task_list
agent_02

Figure 4.5: High-level representation of the LNS-Auction algorithm in a system of two robots.

Type	Name	Description
`PoseStamped`	`pose`	Position of a task in Euclidean space
`uint64`	`task_id`	The identification number of the task
`int16`	`task_type`	Identifying the type of task in Section 4.1.2
`int16`	`agent_id`	Currently assigned agent's id
`int16[]`	`capable_agent_list`	List of agents capable of performing the task

Table 4.3: Contents of a `Task.msg` message

Type	Name	Description
`int16`	`bid_set`	List of tasks in the bid-set
`uint64`	`bid_val`	Value of the bid

Table 4.4: Contents of a `Bid.msg` message

Chapter 5

On Efficient Computation of Shortest Dubins Paths Through Three Consecutive Points

5.1 Problem Formulation

We now formulate the problem of finding an optimal path for a Dubins vehicle (see Section 3.3) between three consecutive points.

5.1.1 Three-Point Dubins Path

Let the tuple $X_i = (\mathbf{x}_i, \alpha_i)$ denote a Dubins vehicle configuration, consisting of a point $\mathbf{x}_i$ in the Euclidean plane, and a heading $\alpha_i \in [0, 2\pi)$ at $\mathbf{x}_i$. An alternative representation of the heading at $\mathbf{x}_i$ is two circular arcs (left and right turns) containing $\mathbf{x}_i$ and tangent to the heading. Given initial and final configurations X_i and X_f, along with a midpoint $\mathbf{x}_m$ with a free heading, the three-point problem is defined by the tuple $(X_i, \mathbf{x}_m, X_f)$ and stated as follows.

Problem 5.1.1 (Three-point Dubins path). *Given a tuple $(X_i, \mathbf{x}_m, X_f)$, with pairwise Euclidean distances between the points $\mathbf{x}_i$, $\mathbf{x}_m$, $\mathbf{x}_f$ of at least $4R_{\min}$, find a heading α_m at $\mathbf{x}_m$ such that the length of an optimal Dubins path starting at X_i, passing through $X_m = (\mathbf{x}_m, \alpha_m)$, and arriving at X_f is minimum.*

From the Bellman's principle of optimality [102], the optimal Dubins path through three configurations is obtained by concatenating two optimal Dubins paths between the pairs. Given two configurations X_1 and X_2, the optimal Dubins path from X_1 to X_2 can be computed in constant time [34]. The optimal Dubins paths between two configurations is in the set $\{CCC, CSC\}$ where S is a straight line segment and C is a circular turn with minimum turning radius in either left L or right R direction. Therefore, in general the optimal path through three points is obtained by concatenating two Dubins paths as follows.

$$\{(C_1C_2C_3)_1(C_4C_5C_6)_2, (C_1C_2C_3)_1(C_4S_5C_6)_2,$$
$$(C_1S_2C_3)_1(C_4C_5C_6)_2, (C_1S_2C_3)_1(C_4S_5C_6)_2\}.$$

From [36] the set of optimal Dubins paths under $4R_{\min}$ distance assumption of Problem 5.1.1 is reduced to $(C_1S_2C_3)_1(C_4S_5C_6)_2$. The $4R_{\min}$ distance constraint is relaxed further in Section 5.6.

5.1.2 Properties of Three-Point Dubins Path

In a path of type $(C_1S_2C_3)_1(C_4S_5C_6)_2$, the arc segments C_3 and C_4 are the two incident path segments to the mid-point. In the optimal solution to Problem 5.1.1, the two arcs incident to the mid-point have equal lengths and both are in the same turning direction i.e., left turn or right turn [36]. Thus for simplicity we represent the path as $C_1S_2C_3S_4C_5$. We summarize the properties of the optimal Dubins path through three consecutive points, provided in [36], as follows.

Lemma 5.1.2 (Three-point Dubins). *Given $(X_i, \mathbf{x}_m, X_f)$, in an optimal path of type $C_1S_2C_3S_4C_5$, the line segment between $\mathbf{x}_m$ and the center of the circle associated with the optimal heading bisects the angle between the line segments S_2 and S_4.*

Substituting the left L and right R turns for each C_i in the path $C_1S_2C_3S_4C_5$, we obtain the set of 8 candidate optimal path types for Problem 5.1.1.

In [35], the authors address a variation of Problem 5.1.1 in which the final heading is also free, i.e., the problem $(X_i, \mathbf{x}_m, \mathbf{x}_f)$. The authors show that under a $2R_{\min}$ distance constraint the optimal path is of type $C_1S_2C_3S_4$, and the optimal heading bisects the angle between S_2 and S_4 as in Lemma 5.1.2. Limiting the path types in the problem $(X_i, \mathbf{x}_m, X_f)$ to $C_1S_2C_3S_4C_5$, the result of Lemma 5.1.2 applies even without the $4R_{\min}$ constraint. The proof follows directly from the proof in [35] for the problem instance $(X_i, \mathbf{x}_m, \mathbf{x}_f)$.

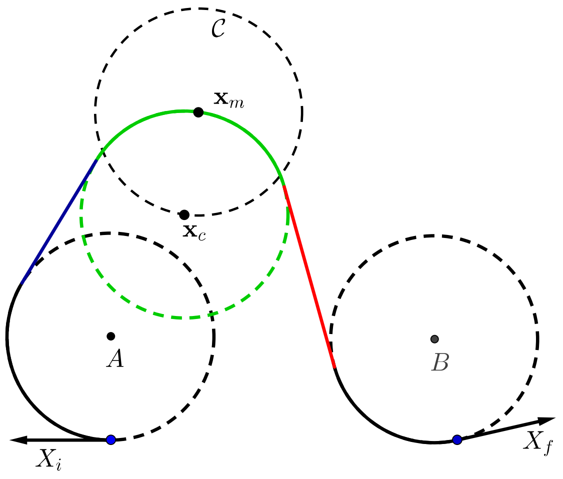


Figure 5.1: An optimal path of type $R_1S_2R_3S_4L_5$. Each component of the path is sketched in different colors.

5.2 Optimal Path and Inversive Geometry

In this section, we use inversive geometry to establish properties of optimal paths of type $C_1S_2C_3S_4C_5$, that form the basis of our solution approach to Problem 5.1.1.

5.2.1 Inversive Geometry in Dubins Paths

Figure 5.1 shows the optimal path for the case $R_1S_2R_3S_4L_5$. The points A, B and $\mathbf{x}_c$ are the centers of the circles associated with the headings at the points $\mathbf{x}_i, \mathbf{x}_f$ and $\mathbf{x}_m$ respectively. In Figure 5.1, the common tangent of the circles centered at A and $\mathbf{x}_c$ is an *outer-common tangent* and the common tangent of the circles centered at $\mathbf{x}_c$ and B is an *inner-common tangent*.

Figure 5.2 shows the inverse of the components of the path with respect to the circle $\mathcal{C}$ centered at $\mathbf{x}_m$ with radius $R_{\min}$. Each S segment in Figure 5.1 is shown as a line in Figure 5.2. The circle inversion operation (see Section 3.4) on each line generates a circle containing the mid-point $\mathbf{x}_m$, shown in Figure 5.2 in the same color. The inverse of the circle associated with the heading at $\mathbf{x}_m$ is a line passing through the two intersection points of circle($\mathbf{x}_m, R_{\min}$) and circle($\mathbf{x}_c, R_{\min}$).

The following lemma provides a sufficient condition for optimality of a $C_1S_2C_3S_4C_5$ path based on Lemma 5.1.2.

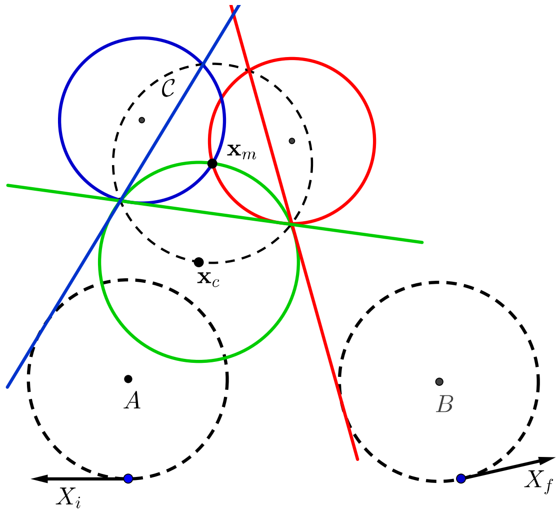


Figure 5.2: Inverse of the components of the path $R_1S_2R_3S_4L_5$ with respect to the dashed circle $\mathcal{C} = \text{circle}(\mathbf{x}_m, R_{\min})$.

Lemma 5.2.1 (Radius of inverted circles in an optimal path)**.** *In any optimal path of type $C_1S_2C_3S_4C_5$, the inverses of the line segments, S_2 and S_4, with respect to a circle centered at $\mathbf{x}_m$ with radius $R_{\min}$ are two circles of equal radius.*

Proof. Consider any optimal path of type $C_1S_2C_3S_4C_5$ — for such a path we can define the following quantities as shown in Figure 5.3. Let P be the intersection of the two line segments in the optimal path and P' be the inverse of P with respect to $\mathcal{C}$. Note that P' is the inverse of the point P and the line $\overline{PP'}$ contains $\mathbf{x}_m$ by the definition of P', thus the inverse of $\overline{PP'}$ with respect to $\mathcal{C}$ is the line itself. The point P is common in both lines S_2 and S_4, thus the point P' is lies on both circles $\text{inv}(S_2, \mathcal{C})$ and $\text{inv}(S_4, \mathcal{C})$. From Lemma 5.1.2 we know that the line $\overline{PP'}$ is the bisector of the angle between two line segments, S_2 and S_4. From Proposition 3.4.4, circle inversion preserves the angle between $\overline{PP'}$ and S_2 and angle between $\overline{PP'}$ and S_4. Therefore, the angles between the line $\overline{PP'}$ and circles $\text{inv}(S_2, \mathcal{C})$ and $\text{inv}(S_4, \mathcal{C})$ are equal. Thus we have, $\angle P'C\mathbf{x}_m = \angle P'D\mathbf{x}_m$, which implies

$$\angle P'DC = \frac{\angle P'C\mathbf{x}_m}{2} = \frac{\angle P'D\mathbf{x}_m}{2} = \angle P'CD.$$

Let Q be the intersection of the lines $\overline{CD}$ and $\overline{P'\mathbf{x}_m}$. The points $\mathbf{x}_m$ and P' are common in both circles $\text{inv}(S_2, \mathcal{C})$ and $\text{inv}(S_4, \mathcal{C})$, implying $\angle P'QC = \angle P'QD = \frac{\pi}{2}$. Thus the triangles $\triangle P'CQ$ and $\triangle P'DQ$ are equal given that they have common side $P'Q$ and two equal

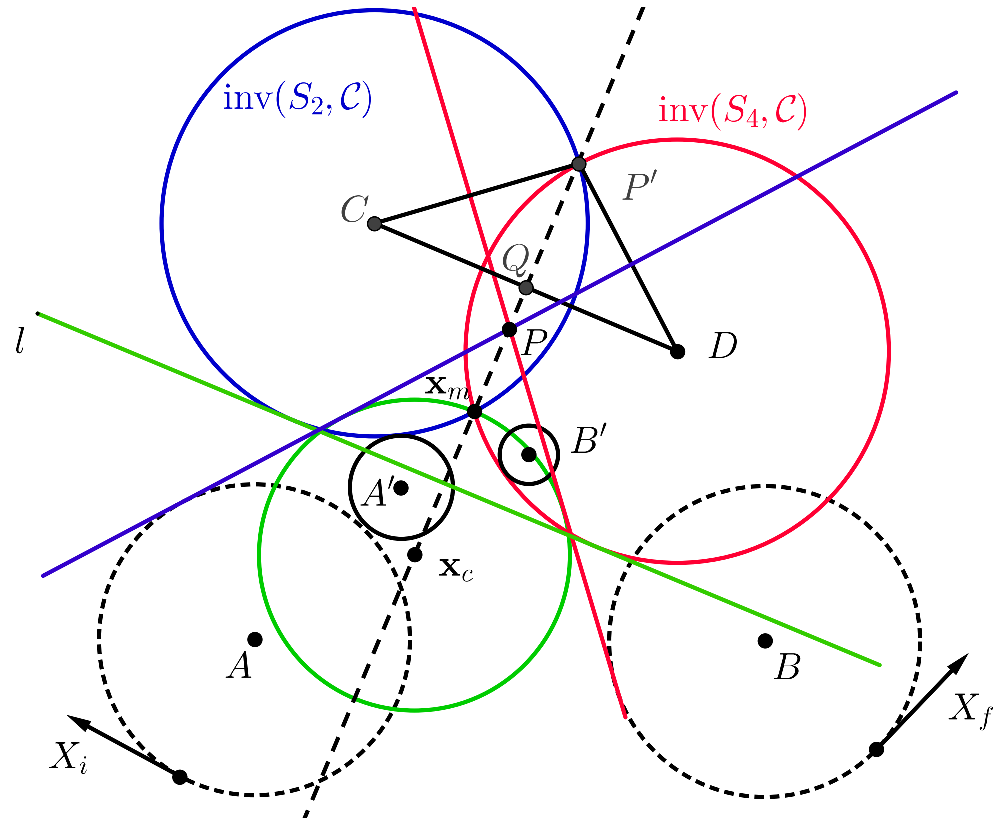


Figure 5.3: Path $R_1S_2R_3S_4L_5$ and inverse of the path components. The optimal path given in the figure with initial state X_i, final state X_f and point $\mathbf{x}_m$ to visit.

angles. Therefore, the segments CP' and DP' have equal length, implying that the circles have same radius. $\square$

5.2.2 Optimality Condition

Without loss of generality we set $R_{\min}$ to 1 in the rest of this chapter, otherwise, we scale the location of the points to satisfy the assumption. In addition, we rotate the coordinate system such that the centers A and B lie on the x-axis. Then, the optimal heading at $\mathbf{x}_m$ equals the angle between the line tangent to circle$(\mathbf{x}_c, R_{\min})$ at $\mathbf{x}_m$ and the x-axis.

Due to the $4R_{\min}$ distance constraint on the points, circle$(A, R_{\min})$ does not contain $\mathbf{x}_m$. Therefore, the inverse of circle$(A, R_{\min})$, i.e.,inv(circle$(A, R_{\min}), \mathcal{C})$, is a circle centered at point A' with radius $r_{A'}$ (see Figure 5.3). The point A' and radius $r_{A'}$ are defined as follows:

$$r_{A'} = \frac{1}{|\overline{A\mathbf{x}_m}|^2 - 1}, \quad |\overline{A'\mathbf{x}_m}| = |\overline{A\mathbf{x}_m}| r_{A'}. \tag{5.1}$$

Substituting A, A' and $r_{A'}$ with B, B' and $r_{B'}$, respectively, we can define B' and $r_{B'}$.

To derive a set of equations for the optimal heading in the path $C_1S_2C_3S_4C_5$, we require the following additional definitions:

- μ_A is 1 if $C_1 = C_3$ and -1 otherwise,
- μ_B is 1 if $C_5 = C_3$ and -1 otherwise,
- R is the radius of the circles centered at C and D in the optimal path,
- $\theta = \angle \mathbf{x}_m CD = \angle \mathbf{x}_m DC$ (see Figure 5.3),
- $\beta_1 = \angle \mathbf{x}_m AB$ and $\beta_2 = \angle \mathbf{x}_m BA$.

Following proposition provide the set of equations to obtain the optimal heading.

Proposition 5.2.2. *The optimal heading α^* at $\mathbf{x}_m$ is the unique solution to the following set of equations:*

$$\frac{1}{2(\mu_A + |\overline{A\mathbf{x}_m}| \cos(\beta_1 + \theta - \alpha^*))} = R, \tag{5.2}$$

$$\frac{1}{2(\mu_B + |\overline{B\mathbf{x}_m}| \cos(\beta_2 + \theta + \alpha^*))} = R, \tag{5.3}$$

$$\frac{1}{2(1 - \sin(\theta))} = R. \tag{5.4}$$

Proof. Figure 5.4 shows the triangles $\triangle CA'\mathbf{x}_m$ and $\triangle DB'\mathbf{x}_m$ of Figure 5.3. The length of the segment $\overline{CA'}$depends on the line segment S_2. If the line segment S_2 is an inner common tangent then circle$(A', r_{A'})$ is contained in circle(C, R) and share a common tangent, therefore, $|\overline{CA'}|$ equals $R - r_{A'}$. On the other hand, if the line segment S_2 is an outer common tangent the circle$(A', r_{A'})$ is tangent to circle(C, R) from outside and the length of the segment $|\overline{CA'}|$ is $R + r_{A'}$. The same applies to the segment $\overline{DB'}$ with respect to the segment S_4. The circles centered at C and D in Figure 5.3 are tangent to line l. Therefore, the distance of the centers C and D from the line l is R. Since the line l passes through the intersection points of the two equally sized circles centered at $\mathbf{x}_m$ and $\mathbf{x}_c$, the distance

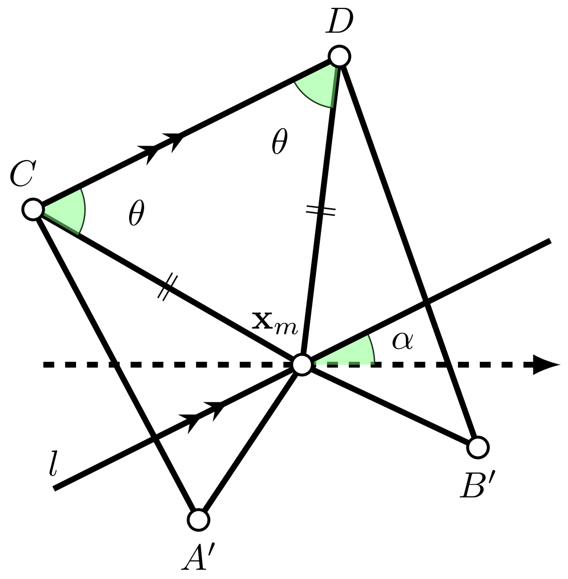


Figure 5.4: Triangles $\triangle CA'\mathbf{x}_m$ and $\triangle DB'\mathbf{x}_m$ of Figure 5.3. Line l contains $\mathbf{x}_m$ and is parallel to the direction of the heading at $\mathbf{x}_m$.

of $\mathbf{x}_m$ from the line between C and D is $R - \frac{R_{\min}}{2}$ which results Equation (5.4). Finally, the cosine law for triangles $\triangle C\mathbf{x}_m A'$ and $\triangle D\mathbf{x}_m B'$ in Figure 5.4 result Equations (5.2) and (5.3), respectively. $\square$

The set of unknowns in Equations (5.2), (5.3) and (5.4) are R, α and θ, where α is the optimal heading at $\mathbf{x}_m$. Unfortunately, we have been unsuccessful in obtaining a closed form solution to these set of trigonometric equations. In Section 5.3.1, we leverage Proposition 5.2.2 to bound the optimal heading at the mid-point. Moreover, we propose a geometric method to approximate the heading, followed by an iterative procedure to converge to the optimal.

5.3 Three-point Dubins Algorithm

In this section, we propose a simple method to find the optimal path in the problem instance $(X_i, \mathbf{x}_m, X_f)$. First, leveraging the properties in Section 5.2, we propose a method to find an approximate midpoint heading.

5.3.1 Approximation Method

In this section, we propose an approximation of the optimal heading at the mid-point $\mathbf{x}_m$. We assume that the pair-wise distances of $\mathbf{x}_i$, $\mathbf{x}_m$ and $\mathbf{x}_f$ go to infinity. Then, the length of segments $\overline{A\mathbf{x}_m}$ and $\overline{B\mathbf{x}_m}$ go to infinity which, by Equation (5.1), implies $|\overline{A'\mathbf{x}_m}|$ and $|\overline{B'\mathbf{x}_m}|$ approach zero. From Lemma A.2.1 (see Appendix A.2), the radius of the circles $\mathrm{inv}(S_2, \mathcal{C})$ and $\mathrm{inv}(S_4, \mathcal{C})$ is bounded from below by $\frac{1}{4}R_{\min}$. Therefore, the angles $\angle \mathbf{x}_m CA'$ and $\angle \mathbf{x}_m DB'$ (see Figure 5.4) approach zero and the angles $\angle C\mathbf{x}_m A$ and $\angle B\mathbf{x}_m D$ go to $\frac{\pi}{2}$.

Therefore, in terms of the angles $\beta_1 = \angle \mathbf{x}_m AB$, $\beta_2 = \angle \mathbf{x}_m BA$, $\theta = \angle \mathbf{x}_m CD$, we have

$$\beta_1 - \alpha + \theta = \frac{\pi}{2}, \quad \beta_2 + \alpha + \theta = \frac{\pi}{2}.$$

From these equations, we can approximate the heading α at the mid-point α by

$$\bar{\alpha} = \frac{\beta_1 - \beta_2}{2}. \tag{5.5}$$

The following result establishes the maximum error between $\bar{\alpha}$ and the optimal heading α^*. The proof is given in Appendix A.2.

Proposition 5.3.1 (Maximum error of approximated heading)**.** *For the optimal path of Problem 5.1.1, the following holds for the optimal heading at* $\mathbf{x}_m$*:*

$$\left|\alpha^* - \frac{\beta_1 - \beta_2}{2}\right| \leq \zeta.$$

For the optimal path P^*, the bound ζ is defined as

(i) $\zeta = 0$ if $|\overline{A\mathbf{x}_m}| = |\overline{B\mathbf{x}_m}|$ and P^* is RSRSR, LSLSL, RSLSR, or LSRSL;

(ii) $\zeta = \frac{\pi}{9}$ if P^* is RSRSR or LSLSL;

(iii) $\zeta = \frac{\pi}{5}$ if P^* is RSLSR or LSRSL; and

(iv) $\zeta = \frac{11\pi}{36}$ if P^* is RSRSL, LSRSR, RSLSL, or LSLSR.

Note that the ζ values in Proposition 5.3.1 are the worst-case bounds, thus in Section 5.6, we provide the mean deviation of the approximated heading from the optimal on 50000 instances with various distance constraints. Also, note that the worst-case bounds improve as the distance between the points increase. Given this approximation of the optimal heading at the mid-point, we can initialize an iterative method to converge to the optimal.

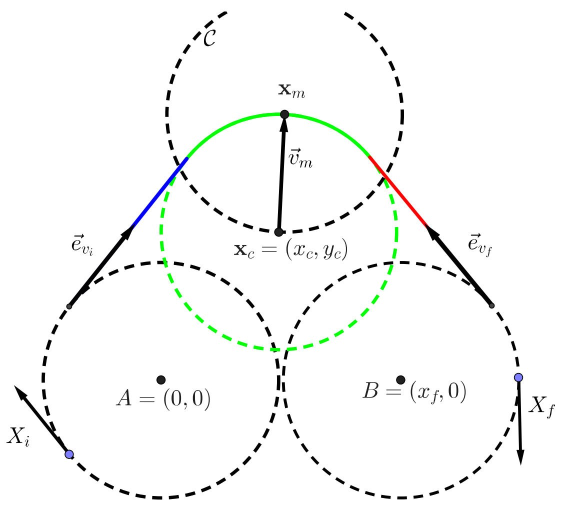


Figure 5.5: Illustration of the vectors $\vec{e}_{v_i}, \vec{e}_{v_f}$ and $\vec{v_m}$ for a path of type $L_1S_2L_3S_4R_5$.

5.3.2 Iterative Method

Starting from the heading given in Section 5.2 as the initial heading, we propose the following method for iteratively improving the heading. The method converges to the optimal heading by iteratively correcting the angle between the bisector of the two line segments of the path $C_1S_2C_3S_4C_5$ and the vector between the mid-point and the center of the circle associated with the heading (see Figure 5.5). Without loss of generality, assume that the center of the first curve is located at the origin and the center of the final curve is located at $(x_f, 0)$ and let $\mathbf{x}_m = (x_m, y_m)$ be the mid-point. We define vectors $\vec{v_i}$ and $\vec{v_f}$ parallel to the first and second line segments the path $C_1S_2C_3S_4C_5$. Let $\mathbf{x}_c = (x_c, y_c)$ be the center of the circle associated with a heading at the mid-point. Let Rot_θ be the rotation matrix with angle θ and $\vec{e}_v$ be the unit length vector in the direction of vector $\vec{v}$. We have,

$$\vec{v}_i = \text{Rot}_{\theta_i}[x_c, y_c], \quad \vec{v}_f = \text{Rot}_{\theta_f}[x_c - x_f, y_c],$$
$$\vec{v}_m = [x_m - x_c, y_m - y_c], \quad \vec{v} = \vec{e}_v + \vec{e}_{v_f}.$$

The angle θ_i is the angle of a common tangent of two circles circle$(A, 1)$ and circle$(\mathbf{x}_c, 1)$ from the line connecting the centers A and $\mathbf{x}_c$. The angle θ_i equals zero if the line segment is an outer-common tangent and $\sin^{-1}(2/|v_i|)$, otherwise. The algorithm for each path of type $C_1S_2C_3S_4C_5$ is as follows:

(i) Find the approximated heading $\bar{\alpha}$ (Equation (5.3.1)),

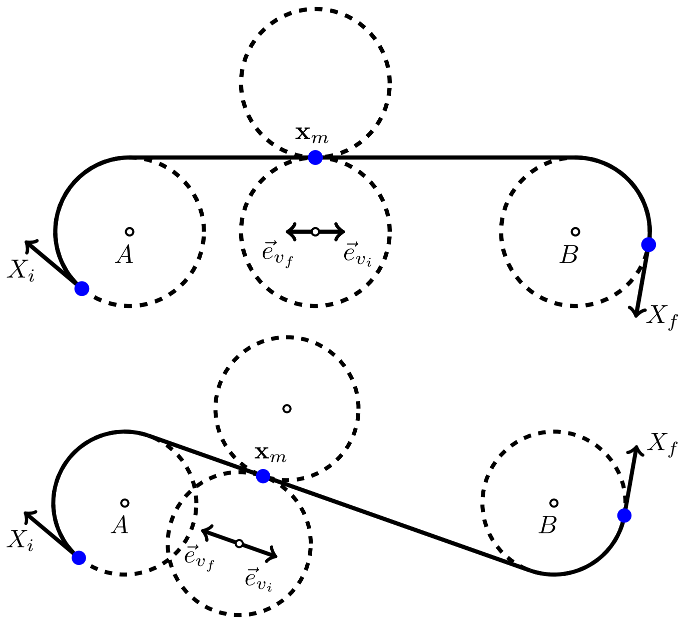


Figure 5.6: Two examples of the case that $\vec{e}_{v_i}$ and $\vec{e}_{v_f}$ sum to zero.

(ii) Compute vectors $\vec{v_i}$, $\vec{v_f}$ and $\vec{v_m}$,

(iii) Compute the vector $\vec{v}$ bisecting the angle between $\vec{v_i}$ and $\vec{v_f}$,

(iv) Return if vectors $\vec{v}$ and $\vec{v_m}$ are aligned,

(v) compute the angle γ between $\vec{v_i}$ and $\vec{v_m}$,

(vi) Rotate (x_c, y_c) about $\mathbf{x}_m$ by γ,

(vii) continue from step (ii).

The problem of finding the optimal heading at the mid-point is defined as the following:

$$\min_{x_c, y_c} \cos^{-1}(\vec{e}_v \cdot \vec{v}_m) \tag{5.6}$$

The minimum of (5.6) occurs when the vectors $\vec{v}$ and $\vec{v_m}$ are parallel. Note that the derivative of the right hand side of objective function (5.6) is not defined where the vectors $\vec{e}_v$ and $\vec{v}_m$ are parallel. However, minimizing (5.6) is equivalent to the following maximization:

$$\max_{x_c, y_c} \vec{e}_v \cdot \vec{v}_m \tag{5.7}$$

To prove correctness of the iterative method, it suffices to show that all local maxima (x_c, y_c) of (5.7) are globally maximal. The following lemma validates the iterative method.

Lemma 5.3.2. *The optimal heading is the unique maximim of* (5.7)*.*

Proof. Note that the length of the vectors $\vec{v}_m$, $\vec{e}_{v_i}$ and $\vec{e}_{v_f}$ are independent of α, and the derivative of either of the vectors is orthogonal to the vector itself. The derivative of $\vec{e}_v \cdot \vec{v}_m$ with respect to change of the heading α at the mid-point is as follows:

$$\begin{aligned}\frac{d(\vec{e}_v \cdot \vec{v}_m)}{d\alpha} = &-\frac{\vec{e}_{v_f} \cdot \frac{d\vec{e}_{v_i}}{d\alpha} + \vec{e}_{v_i} \cdot \frac{d\vec{e}_{v_f}}{d\alpha}}{|\vec{e}_{v_i} + \vec{e}_{v_f}|^3}(\vec{e}_{v_i} + \vec{e}_{v_f}) \cdot \vec{v}_m + \\ &\frac{\frac{d\vec{e}_{v_i}}{d\alpha} + \frac{d\vec{e}_{v_f}}{d\alpha}}{|\vec{e}_{v_i} + \vec{e}_{v_f}|} \cdot v_m + \frac{\vec{e}_{v_i} + \vec{e}_{v_f}}{|\vec{e}_{v_i} + \vec{e}_{v_f}|} \cdot \frac{dv_m}{d\alpha}.\end{aligned} \tag{5.8}$$

Case (i): The vectors $\vec{v_i}$ and $\vec{v_f}$ are dependent, Equation (5.8) simplifies to

$$\frac{d(\vec{e}_v \cdot \vec{v}_m)}{d\alpha} = \frac{\frac{d\vec{e}_{v_i}}{d\alpha} + \frac{d\vec{e}_{v_f}}{d\alpha}}{|\vec{e}_{v_i} + \vec{e}_{v_f}|} \cdot v_m + \frac{\vec{e}_{v_i} + \vec{e}_{v_f}}{|\vec{e}_{v_i} + \vec{e}_{v_f}|} \cdot \frac{dv_m}{d\alpha}. \tag{5.9}$$

Note that $|\vec{e}_{v_i} + \vec{e}_{v_f}|$ equals zeros only if the vectors $\vec{e}_{v_i}$ and $\vec{e}_{v_f}$ are collinear with different directions. Therefore, the line segments are tangent to the circle at $\mathbf{x}_m$ and the heading corresponding to this case is the optimal heading (see Figure 5.6). If $\vec{e}_{v_i}$ and $\vec{e}_{v_f}$ do not sum up to zero, then the optimal heading is the root of the following equation:

$$\begin{aligned}(\frac{d\vec{e}_{v_i}}{d\alpha} + \frac{d\vec{e}_{v_f}}{d\alpha}) \cdot v_m + (\vec{e}_{v_i} + \vec{e}_{v_f}) \cdot \frac{dv_m}{d\alpha} \\ = \frac{d}{d\alpha}((\vec{e}_{v_i} + \vec{e}_{v_f}) \cdot \vec{v}_m) = 0.\end{aligned}$$

Trivially, the roots of this equation occur where $\vec{v_m}$ is parallel to $\vec{e}_{v_i} + \vec{e}_{v_f}$.

Case (ii): The vectors $\vec{v_i}$ and $\vec{v_f}$ are linearly independent. Therefore, we can write $\vec{v_m}$ and the derivative as a linear combinations of $\vec{v_i}$ and $\vec{v_f}$, i.e.

$$\vec{v}_m = \frac{1}{|\vec{e}_{v_i} + \vec{e}_{v_f}|}\vec{e}_{v_i} + \frac{c}{|\vec{e}_{v_i} + \vec{e}_{v_f}|}\vec{e}_{v_f}, \tag{5.10}$$

$$\frac{d\vec{v}_m}{d\alpha} = \frac{1}{|\vec{e}_{v_i} + \vec{e}_{v_f}|}\mathrm{Rot}_{\frac{\pi}{2}}\vec{e}_{v_i} + \frac{c}{|\vec{e}_{v_i} + \vec{e}_{v_f}|}\mathrm{Rot}_{\frac{\pi}{2}}\vec{e}_{v_f}. \tag{5.11}$$

In order to prove that the extremums occur where the bisector of the angle between $\vec{v_i}$ and $\vec{v_f}$ aligns with $\vec{v_m}$, we need to show that $c = 1$ is the only solution to Equation (5.8). Substituting Equations (5.10) and (5.11) into Equation (5.8) and simplify the equation, we have

$$\begin{aligned}(1-c)(\vec{e}_{v_f} \cdot \mathrm{Rot}_{\frac{\pi}{2}} \vec{e}_{v_i} + c\vec{e}_{v_i} \cdot \mathrm{Rot}_{\frac{\pi}{2}} \vec{e}_{v_f}) = \\ (1-c)^2(\vec{e}_{v_i} \cdot \mathrm{Rot}_{\frac{\pi}{2}} \vec{e}_{v_f}) = 0.\end{aligned}$$

With the assumption that $\vec{v_i}$ and $\vec{v_f}$ are linearly independent, the equation equals to zero if and only if $c = 1$. □

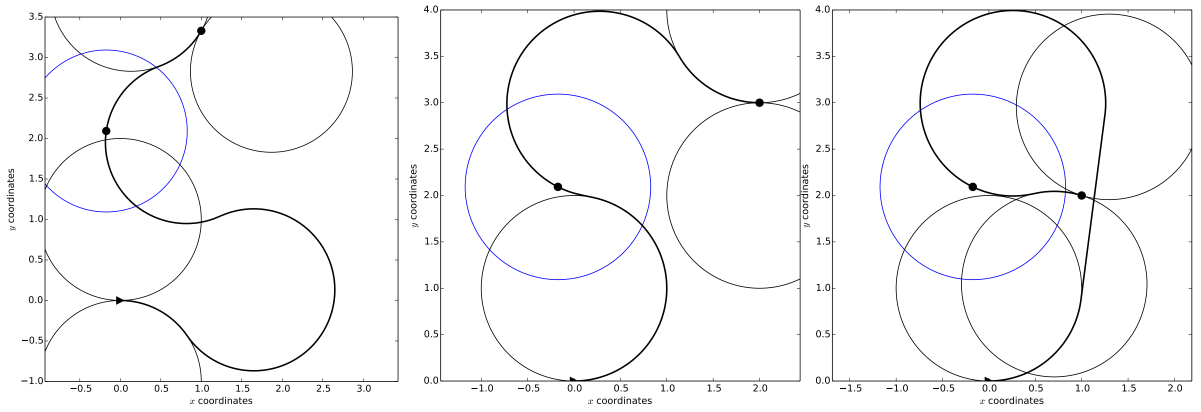
An immediate consequence of Lemma 5.3.2 is the convergence of the iterative method.

Corollary 5.3.3. *The iterative method converges to the optimal heading at the mid-point.*

5.4 Relaxing The Distance Constraint

The $4R_{\min}$ distance constraint in Problem 5.1.1 ensures that the path types are of type $C_1S_2C_3S_4C_5$. Eliminating the distance constraint introduces additional path types, i.e., paths including CC and CCC segments. The proof of optimality for Lemma 5.3.2 does not consider any distance constraint between the points. Therefore, the iterative method is applicable to any $C_1S_2C_3S_4C_5$ path type even when $4R_{\min}$ is not satisfied.

Note that by eliminating the $4R_{\min}$ distance constraint, the optimal path may contain CC path types. The Dubins path type CC is considered as a CSC path with a zero-length line segment. However, the results in Lemma 5.1.2 does not hold for this case. Recall that the lemma requires a non-zero line segment. Figure 5.7 shows three optimal paths consisting of a CC segment. A constant time method is presented in [32] for computing CC paths. Implementing the method for computing CC paths alongside our iterative method for $C_1S_2C_3S_4C_5$ paths, we obtain a method to optimally find the heading at the mid point for all path types between three consecutive points with exception of $\{C_1C_2C_3S_4C_5, C_1S_2C_3C_4C_5, C_1C_2C_3C_4C_5\}$. Although these path types are not considered in our method, the extensive simulation results in Section 5.6 shows that under the relaxed distance condition the paths generated by our method are in 0.1 percent of the optimal path.



(a) $(C_1C_2C_3)_1(C_3C_4)_2$ (b) $(C_1C_2)_1(C_2C_3)_2$ (c) $(C_1S_2C_3)_1(C_3C_4)_2$

Figure 5.7: Optimal three point Dubins paths – each consisting of a CC segment.

5.5 Locally Optimizing a Dubins TSP Tour

The solution to Problem 5.1.1 provides a method for locally optimizing a Dubins TSP tour in a post-processing phase. Given a set of n points in the Euclidean plane, a solution to the Dubins TSP is an ordering of the n points, along with a heading at each point that minimizes the total path length. Let T be a Dubins tour such that T_i is the ith configuration $(\mathbf{x}_i, \alpha_i)$. Now we define our post-processing method as follows:

(i) For every T_i, solve the problem $(T_{i-1}, \mathbf{x}_i, T_{i+1})$ and update α_i,

(ii) Randomly delete a configuration T_i in T and re-insert to a position in the tour with minimum additional cost.

Note that every segment of three consecutive vertices on the tour is a $(X_i, \mathbf{x}_m, X_f)$ problem instance. Therefore, in a tour of length n, finding the position to insert a point with minimum additional cost requires solving $n-1$ problem instances of type $(X_i, \mathbf{x}_m, X_f)$. The steps (i) and (ii) of refinements terminates if there is no improvement in the path.

5.6 Simulation Results

We evaluate the performance of the proposed approach on both randomly generated $(X_i, \mathbf{x}_m, X_f)$ instances and in post-processing Dubins TSP tours as in Section 5.5. The

	$2R_{\min}$	$3R_{\min}$	$4R_{\min}$
Approx. heading	65.3	67.1	74.2
Iterative method	5.2	6.8	13.6

Table 5.1: The factor of improvement in runtime of the iterative and approximation method over 360 discrete headings.

point-to-point Dubins path [103] and the three-point Dubins method are implemented in Python and the experiments are conducted on an Intel Corei5 @2.5Ghz processor. The experiments in this section consider a Dubins vehicle with $R_{\min} = 1$.

5.6.1 Three-Point Dubins

In this section, we compare the performance of the initial approximation and the iterative method to discretizing the heading at $\mathbf{x}_m$ with 360 equally-spaced headings. Let α_d be a heading among the discretized headings. The discretization method creates the configuration $X_m = (\mathbf{x}_m, \alpha_d)$, and solves two Dubins path problems, namely (X_i, X_m) and (X_m, X_f). The discretization method returns the minimum path among the headings.

Figure 5.8 (top) shows the percentage deviation in path length for the approximate heading $\bar{\alpha}$ in (5.5), relative to the path length computed using 360 heading discretizations. The bottom figure shows the deviation of the path length produced by the iterative method to that of the discretized heading. The experiments are conducted on 50000 random $(X_i, \mathbf{x}_m, X_f)$ instances, where the points are uniformly randomly selected in a 10×10 environment. The x-axis in Figure 5.8 is the rounded minimum distance of the three points. The negative values represent instances in which the proposed methods outperform the discretization method. The distribution shows that even in the cases where points are less than $4R_{\min}$ apart from each other, the iterative method generates shorter paths.

The average computation time may vary based on the distances of the points due to considering additional path types mentioned in Section 5.4. The iterative method improves the runtime of computing a three-point Dubins path, under $4R_{\min}$ distance constraint, compared to 360 discretization by a factor of 13.65. However, this factor of improvement is 5.21 for the instances with points less than $2R_{\min}$ apart. Table 5.1 shows the factor of improvement in runtime of the iterative and approximation method when compared to discretization with 360 headings.

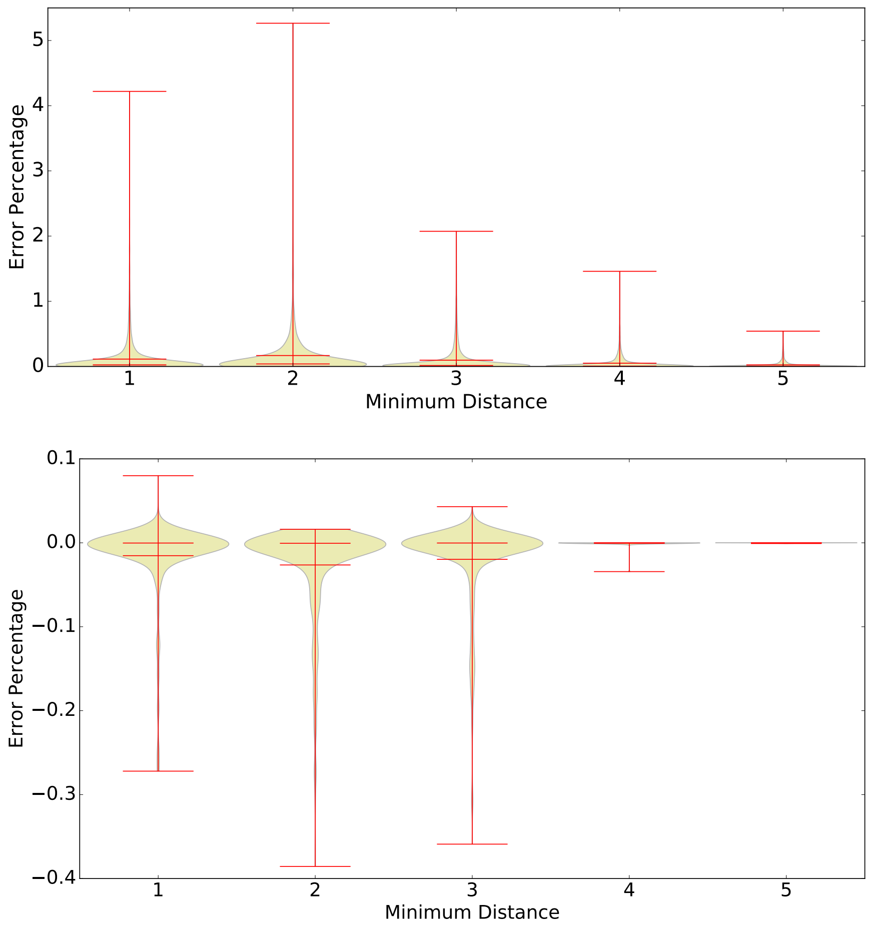


Figure 5.8: The percentage deviation of the length of paths generated by the approximation method (top) and iterative method (bottom) from the discretization method with 360 equally spaced headings. The width of the distributions represent the probability of occurring an instance in the corresponding difference percentile.

5.6.2 Post-processing on Dubins Tour

In this experiment, we implement the GTSP method [9] on random instances with various discretization levels followed by our post-processing method in Section 5.5. Given a Dubins TSP on n points, and a discretization level of d at each point, the GTSP instance will have nd vertices. The results show the advantages of the local optimization on GTSP solutions with coarse discretization over solving GTSP with fine discretization.

To characterize the performance of our algorithm, we conduct experiments on low and high-density Dubins TSP instances. Tables 5.2 and 5.3 show the results on uniformly randomly generated instances. Each row of the table is a class of 20 random instances with the same problem parameters: that is, the environment size $W \times W$, the number of points N, and the minimum pair-wise distance D.

The GTSP instances are solved using the state-of-the-art GTSP solver, GLKH [100] which is implemented in `C`. In Table 5.2 the abbreviations G. Len and G. Time represent the average tour length and solver time, in seconds, for the GTSP solver. Similarly, P. Len represents the average tour length after post-processing and P. Time represents the time required for the post-processing of the GTSP tour. The total time of the GTSP approach and the post-processing is denoted by Time. Table 5.2 shows the performance of the post-processing technique on the GTSP tours with a discretization level of 1 and 10 in low-density Dubins TSP instances. The time and the tour length of the GTSP solution with discretization level 20 is the reference, denoted by ref, for evaluating the performance of the post-processing method. The table includes the ratios of the total time and post-processed tour length to the reference. In the class of instances N30W20D2.0, the deviation of the post-processed tour length from the reference is 3.7% and the total time of solving the GTSP with 1-discretization followed by the post-processing technique is just 1.6% of the solver-time of the GTSP approach with discretization level 20.

In an environment with high density of points, the discretization level has larger impact on the ordering of the points in a GTSP solution. Table 5.3 shows the results of the GTSP tour with post-processing on high-density instances. For example, the results on the class of instances N50W20D0.0 show that the tour length of the post-processed GTSP tour with discretization level 5 is 5.3% longer than the GTSP tour with discretization level 20. However, the runtime is improved by a factor of 13.26.

		Low Density					
		P. Time	G. Time	G. Len	P. Len	P. Len/ref	Time/ref
disc. 1	N10W15D4.0	0.1	0.0	75.5	54.6	1.000	0.018
	N10W10D3.0	0.1	0.0	66.0	38.2	1.003	0.029
	N20W20D3.0	0.3	0.0	142.5	94.5	1.002	0.013
	N30W20D2.0	0.5	0.4	187.3	110.3	1.037	0.016
	N30W30D3.0	0.4	0.1	229.0	157.6	1.006	0.030
	N40W30D4.0	0.9	0.3	289.7	205.3	1.022	0.040
disc. 10	N10W15D4.0	0.1	0.8	54.8	54.6	1.000	0.167
	N10W10D3.0	0.1	0.6	38.4	38.1	1.000	0.178
	N20W20D3.0	0.3	9.5	94.7	94.5	1.002	0.255
	N30W20D2.0	0.5	18.4	107.1	106.4	1.000	0.182
	N30W30D3.0	0.4	20.5	157.1	156.7	1.000	0.152
	N40W30D4.0	0.9	45.8	201.3	200.7	1.000	0.162

Table 5.2: Average tour length and time of the GTSP approach compared to the post-processing method on random instances with low-density of points. The instance names consist of the environment size $W \times W$, the number of points N, and the minimum pair-wise distance D.

	P. Time	GTSP 3-discretization			GTSP 5-discretization		
		G. Time	G. Len	P. Len	G. Time	G. Len	P. Len
N10W5D0.0	0.3	0.6	30.8	27.8	1.5	28.0	25.1
N20W5D0.0	0.6	4.48	50.37	45.9	7.5	44.6	41.8
N30W5D0.0	2.3	13.9	68.5	59.4	24.5	58.2	54.5
N30W20D0.0	0.7	1.4	112.9	103.7	3.6	103.0	97.5
N50W20D0.0	0.8	13.3	170.5	160.2	23.2	153.5	145.2
	P. Time	GTSP 10-discretization			GTSP 20-discretization		
		G. Time	G. Len	P. Len	G. Time	G. Len	P. Len
N10W5D0.0	0.3	2.4	24.6	22.9	6.5	21.4	21.2
N20W5D0.0	0.6	16.5	38.6	36.4	46.2	35.1	34.8
N30W5D0.0	2.3	39.2	52.5	50.9	121.4	46.6	46.0
N30W20D0.0	0.7	16.0	97.0	93.2	67.3	95.4	92.3
N50W20D0.0	0.8	63.3	141.7	137.9	318.0	137.7	136.9

Table 5.3: Average tour length and time of the GTSP approach compared to the post-processing method on random instances with high-density of points. The instance names consist of the environment size $W \times W$, the number of points N, and the minimum pair-wise distance D.

Chapter 6

Conclusions and Future Work

This thesis presented two routing and scheduling problems. The first problem considered task allocation and sequencing for multiple robots with differential motion constraints. Our approach was based on transforming the problem to a multi-vehicle GTSP. We provided bounds on the performance of several new insertion methods for the GTSP. We also proposed an auction-based distributed implementation of the allocation and sequencing algorithm that utilizes the core ideas of large neighborhood search.

In the second part of the thesis, we considered the problem of three-point Dubins path between three consecutive points. The presented approximation method followed by the iterative method show improvement in run-time compared to the discretization of the headings. In addition to the experimental results, the application of inversive geometry provides a direction for further research.

6.1 Future Directions for Distributed Task Allocation and Sequencing Algorithm for Robots with Differential Constraints

The sampling of the configuration space offers the flexibility to define a different number of tasks types [12] and the approximate configuration space of a large number of vehicle models [9, 11]. The solution quality of the sampling methods relies on the number of the samples and the size of the configuration space. For instance, consider a 14 DOF robotic arms performing a set of tasks. The sampling of the 14 dimensional configuration space to cover the close to optimal solutions requires an enormous number of samples. The discretization level for approximating the motion of the robots directly affects the quality of the result and has an inverse relation to the solving time. In Chapter 5 we proposed the idea of inserting tasks without explicitly constructing samples of the allowable configurations for each task for the Dubins vehicle. Although the implementation shows considerable improvement in the quality, this approach is limited to the Dubins vehicle. For future work, we are pursuing the idea of sampling the continuous space and eliminating the samples that are not present in any optimal solution. A related problem is the elimination of the edges in the symmetric-TSP [104]. The authors provide heuristic methods to eliminate the edges that are not present in any optimal path. In [105], the authors introduce an LP-based approach to eliminate edges in a GTSP instance. Eliminating all the edges incident to a vertex in the GTSP problem removes the vertex from the problem and reduces the problem size. This approach is based on the LP-formulation of the GTSP problem and requires solving the LP several times.

6.2 Future Directions for Optimal Dubins Path Between Three-Consecutive Points

The problem of finding the optimal Dubins path through three consecutive points is originally motivated by the well-known TSP-insertion methods [6]. Implementation of the large neighborhood search (LNS) with the TSP-insertion methods shows considerably good results on TSP instances. Similar to the analogy of the TSP insertion methods, our algorithm for three points Dubins path inserts a point between two configurations with optimal cost. However, the LNS algorithm relies on repeated deletions and insertions in order to achieve a close to optimal solution. In the GTSP implementations of Dubins TSP, the pair to pair distances are precomputed and the computation of an insertion cost requires accessing the fixed location in the memory in constant time. On the one hand, the GTSP method has the discretization error and the run-time is extremely dependent on the discretization level and on the other hand, accessing the pre-computed distances is faster than computing the insertion costs with our algorithm. Therefore, for future research, we will pursue the idea of introducing insertion methods and large neighborhood search algorithm for Dubins TSP based on the proposed method in Chapter 5 and eliminating redundant computations of the insertion costs.

As another direction for the future work, we are interested in extending our method to assign headings to n ordered points. In fact in the optimal path through n points, each subset of three consecutive points is the optimal solution to the problem we addressed

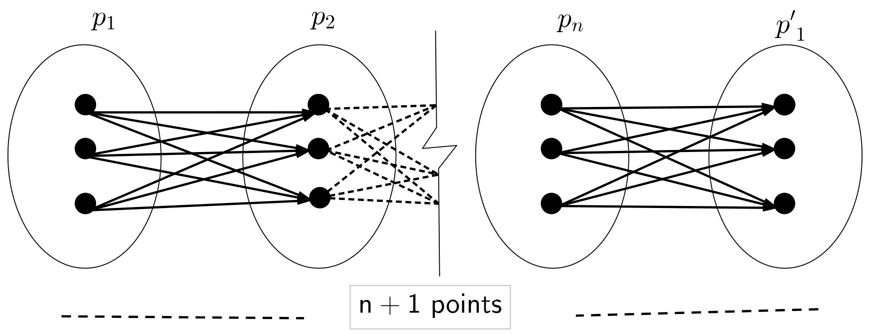


Figure 6.1: In the Dubins tour through n ordered points problem, the headings are discretized at each point and the problem is presented as the shortest path problem in multipartite graphs. Point p'_1, the replica of p_1, is added to the problem in order to close the tour.

in Chapter 5. The problem of Dubins path through n order points with $4R_{\min}$ pair-wise distances is addressed in [36] by solving 2^n convex optimization problems. The problem of optimal Dubins path through n points can be approximated by discretization the headings at the points [1] and implementing the Breadth-first search algorithm (BFS) on the multipartite graph shown in Figure 6.1. Therefore, we are interested in reducing the 2^n path types to one by finding the shortest path in the multipartite graph and converge to the optimal by the iterative method in Section 5.3.2.

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*, vol. 45, no. 3, pp. 378–394, 1997.

[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*, vol. 23, no. 9, pp. 939–954, 2004.

[3] G. A. Korsah, A. Stentz, and M. B. Dias, "A comprehensive taxonomy for multi-robot task allocation," *The International Journal of Robotics Research*, vol. 32, no. 12, pp. 1495–1512, 2013.

[4] K. Helsgaun, "An effective implementation of the Lin–Kernighan traveling salesman heuristic," *European Journal of Operational Research*, vol. 126, no. 1, pp. 106–130, 2000.

[5] B. Korte and J. Vygen, *Combinatorial Optimization: Theory and Algorithms*, 4th ed., ser. Algorithmics and Combinatorics. Springer-Verlag, 2007, vol. 21.

[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*, vol. 6, no. 3, pp. 563–581, 1977.

[7] Z. Tang and U. Ozguner, "Motion planning for multitarget surveillance with mobile sensor agents," *IEEE Transactions on Robotics*, vol. 21, no. 5, pp. 898–908, 2005.

[8] K. Savla, E. Frazzoli, and F. Bullo, "Traveling salesperson problems for the Dubins vehicle," *IEEE Transactions on Automatic Control*, vol. 53, no. 6, pp. 1378–1391, 2008.

[9] J. Le Ny, E. Feron, and E. Frazzoli, "On the Dubins traveling salesman problem," *IEEE Transactions on Automatic Control*, vol. 57, no. 1, pp. 265–270, 2012.

[10] D. Karapetyan and G. Gutin, "Lin–Kernighan heuristic adaptations for the generalized traveling salesman problem," *European Journal of Operational Research*, vol. 208, no. 3, pp. 221–232, 2011.

[11] 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*, vol. 25, no. 3, pp. 207–223, 2006.

[12] J. T. Isaacs, D. J. Klein, and J. P. Hespanha, "Algorithms for the traveling salesman problem with neighborhoods involving a Dubins vehicle," in *American Control Conference*, 2011, pp. 1704–1709.

[13] F. Imeson and S. L. Smith, "Multi-robot task planning and sequencing using the SAT-TSP language," in *IEEE International Conference on Robotics and Automation*, 2015, pp. 5397 – 5402.

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

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

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

[17] H.-L. Choi, L. Brunet, and J. P. How, "Consensus-based decentralized auctions for robust task allocation," *IEEE Transactions on Robotics*, vol. 25, no. 4, pp. 912–926, 2009.

[18] 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*, vol. 2, no. 01, pp. 87–100, 2014.

[19] M. G. Lagoudakis, E. Markakis, D. Kempe, P. Keskinocak, A. J. Kleywegt, S. Koenig, C. A. Tovey, A. Meyerson, and S. Jain, "Auction-based multi-robot routing." in *Robotics: Science and Systems*, vol. 5, 2005.

[20] L. Liu and D. A. Shell, "An anytime assignment algorithm: From local task swapping to global optimality," *Autonomous Robots*, vol. 35, no. 4, pp. 271–286, 2013.

[21] S. L. Smith and F. Bullo, "Monotonic target assignment for robotic networks," *IEEE Trans on Automatic Control*, vol. 54, no. 9, pp. 2042–2057, 2009.

[22] P. Sujit and R. Beard, "Distributed sequential auctions for multiple uav task allocation," in *American Control Conference*, 2007, pp. 3955–3960.

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

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

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

[26] M. H. Rothkopf, A. Pekeč, and R. M. Harstad, "Computationally manageable combinational auctions," *Management science*, vol. 44, no. 8, pp. 1131–1147, 1998.

[27] M. Mito and S. Fujita, "On heuristics for solving winner determination problem in combinatorial auctions," *Journal of Heuristics*, vol. 10, no. 5, pp. 507–523, 2004.

[28] M. Turpin, N. Michael, and V. Kumar, "CAPT: Concurrent assignment and planning of trajectories for multiple robots," *International Journal of Robotics Research*, vol. 33, no. 1, pp. 98–112, 2014.

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

[30] S. Rathinam, R. Sengupta, and S. Darbha, "A resource allocation algorithm for multivehicle systems with nonholonomic constraints," *Automation Science and Engineering, IEEE Transactions on*, vol. 4, no. 1, pp. 98–104, 2007.

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

[32] P. Isaiah and T. Shima, "Motion planning algorithms for the Dubins travelling salesperson problem," *Automatica*, vol. 53, pp. 247–255, 2015.

[33] B. Hérissé and R. Pepy, "Shortest paths for the Dubins' vehicle in heterogeneous environments," in *Annual Conference on Decision and Control*. IEEE, 2013, pp. 4504–4509.

[34] L. E. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *American Journal of mathematics*, pp. 497–516, 1957.

[35] X. Ma, D. Castañón *et al.*, "Receding horizon planning for Dubins traveling salesman problems," in *45th IEEE Conference on Decision and Control*. IEEE, 2006, pp. 5453–5458.

[36] X. Goaoc, H.-S. Kim, and S. Lazard, "Bounded-curvature shortest paths through a sequence of points using convex optimization," *SIAM Journal on Computing*, vol. 42, no. 2, pp. 662–684, 2013.

[37] P. Vana and J. Faigl, "On the Dubins traveling salesman problem with neighborhoods," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2015, pp. 4029–4034.

[38] D. G. Macharet and M. F. Campos, "An orientation assignment heuristic to the Dubins traveling salesman problem," in *Advances in Artificial Intelligence–IBERAMIA 2014*. Springer, 2014, pp. 457–468.

[39] D. G. Macharet, A. Alves Neto, V. F. da Camara Neto, and M. F. Campos, "Efficient target visiting path planning for multiple vehicles with bounded curvature," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 3830–3836.

[40] M. S. Cons, T. Shima, and C. Domshlak, "Integrating task and motion planning for unmanned aerial vehicles," *Unmanned Systems*, vol. 2, no. 01, pp. 19–38, 2014.

[41] S. Manyam and S. Rathinam, "On tightly bounding the Dubins traveling salesmans optimum," *arXiv preprint arXiv:1506.08752*, 2015.

[42] X.-N. Bui, J.-D. Boissonnat, P. Soueres, and J.-P. Laumond, "Shortest path synthesis for Dubins non-holonomic robot," in *Proceedings of the IEEE International Conference on Robotics and Automation*. IEEE, 1994, pp. 2–7.

[43] J. Le Ny and E. Feron, "An approximation algorithm for the curvatureconstrained traveling salesman problem," in *Proceedings of the 43rd Annual Allerton Conference on Communications, Control and Computing*, 2005, pp. 620–9.

[44] L. Techy and C. A. Woolsey, "Minimum-time path planning for unmanned aerial vehicles in steady uniform winds," *Journal of guidance, control, and dynamics*, vol. 32, no. 6, pp. 1736–1746, 2009.

[45] T. G. McGee and J. K. Hedrick, "Path planning and control for multiple point surveillance by an unmanned aircraft in wind," in *American Control Conference, 2006*. IEEE, 2006, pp. 6–pp.

[46] S. Manyam, S. Rathinam, D. Casbeer, and E. Garcia, "Shortest paths of bounded curvature for the Dubins interval problem," *arXiv preprint arXiv:1507.06980*, 2015.

[47] H. Chitsaz and S. M. LaValle, "Time-optimal paths for a Dubins airplane," in *IEEE Conference on Decision and Control*. IEEE, 2007, pp. 2379–2384.

[48] S. Hota and D. Ghose, "Optimal path planning for an aerial vehicle in 3D space," in *IEEE Conference on Decision and Control (CDC)*. IEEE, 2010, pp. 4902–4907.

[49] Y. Wang, S. Wang, M. Tan, C. Zhou, and Q. Wei, "Real-time dynamic Dubins-helix method for 3-d trajectory smoothing," *Control Systems Technology, IEEE Transactions on*, vol. 23, no. 2, pp. 730–736, 2015.

[50] D. G. Macharet and M. F. Campos, "Adaptive path planning for multiple vehicles with bounded curvature," in *Robotics*. Springer, 2015, pp. 153–168.

[51] P. R. Giordano and M. Vendittelli, "Shortest paths to obstacles for a polygonal Dubins car," *IEEE Transactions on Robotics*, vol. 25, no. 5, pp. 1184–1191, 2009.

[52] A. A. Furtuna, D. J. Balkcom, H. Chitsaz, and P. Kavathekar, "Generalizing the Dubins and Reeds-Shepp cars: fastest paths for bounded-velocity mobile robots," in *IEEE International Conference on Robotics and Automation*. IEEE, 2008, pp. 2533–2539.

[53] D. Guimaraes Macharet, A. Alves Neto, V. Fiuza da Camara Neto, and M. Montenegro Campos, "An evolutionary approach for the Dubins' traveling salesman problem with neighborhoods," in *Proceedings of the 14th annual conference on Genetic and evolutionary computation*. ACM, 2012, pp. 377–384.

[54] A. Settimi and L. Pallottino, "A subgradient based algorithm for distributed task assignment for heterogeneous mobile robots," in *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on.* IEEE, 2013, pp. 3665–3670.

[55] S. Rathinam and R. Sengupta, "Lower and upper bounds for a multiple depot uav routing problem," in *Decision and Control, 2006 45th IEEE Conference on.* IEEE, 2006, pp. 5287–5292.

[56] X. Yu and J. Y. Hung, "Optimal path planning for an autonomous robot-trailer system," in *IECON 2012-38th Annual Conference on IEEE Industrial Electronics Society.* IEEE, 2012, pp. 2762–2767.

[57] R. P. Anderson and D. Milutinović, "On the construction of minimum-time tours for a Dubins vehicle in the presence of uncertainties," *Journal of Dynamic Systems, Measurement, and Control*, vol. 137, no. 3, p. 031001, 2015.

[58] S. G. Manyam, S. Rathinam, and S. Darbha, "Computation of lower bounds for a multiple depot, multiple vehicle routing problem with motion constraints," *Journal of Dynamic Systems, Measurement, and Control*, vol. 137, no. 9, p. 094501, 2015.

[59] P. Sujit, B. Hudzietz, and S. Saripalli, "Route planning for angle constrained terrain mapping using an unmanned aerial vehicle," *Journal of Intelligent & Robotic Systems*, vol. 69, no. 1-4, pp. 273–283, 2013.

[60] D. W. Hodo, D. M. Bevly, J. Y. Hung, S. Millhouse, and B. Selfridge, "Optimal path planning with obstacle avoidance for autonomous surveying," in *IECON 2010-36th Annual Conference on IEEE Industrial Electronics Society.* IEEE, 2010, pp. 1577–1583.

[61] R. W. Beard and T. W. McLain, "Implementing Dubins airplane paths on fixed-wing uavs," *Contributed Chapter to the Springer Handbook for Unmanned Aerial Vehicles*, 2013.

[62] G. Reinelt, "TSPLIB – a traveling salesman problem library," *ORSA Journal on Computing*, vol. 3, no. 4, pp. 376–384, 1991.

[63] D. Pisinger and S. Ropke, "A general heuristic for vehicle routing problems," *Computers & operations research*, vol. 34, no. 8, pp. 2403–2435, 2007.

[64] S. Ropke and D. Pisinger, "An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows," *Transportation science*, vol. 40, no. 4, pp. 455–472, 2006.

[65] E. Demir, T. Bektaş, and G. Laporte, "An adaptive large neighborhood search heuristic for the pollution-routing problem," *European Journal of Operational Research*, vol. 223, no. 2, pp. 346–359, 2012.

[66] R. E. Burkard, S. Karisch, and F. Rendl, "Qaplib-a quadratic assignment problem library," *European Journal of Operational Research*, vol. 55, no. 1, pp. 115–119, 1991.

[67] P. Oberlin, S. Rathinam, and S. Darbha, "Today's traveling salesman problem," *Robotics & Automation Magazine, IEEE*, vol. 17, no. 4, pp. 70–77, 2010.

[68] M. Alighanbari, "Task assignment algorithms for teams of uavs in dynamic environments," Ph.D. dissertation, Citeseer, 2004.

[69] J. Bellingham, M. Tillerson, A. Richards, and J. P. How, "Multi-task allocation and path planning for cooperating uavs," in *Cooperative Control: Models, Applications and Algorithms*. Springer, 2003, pp. 23–41.

[70] B. L. Brumitt and A. Stentz, "Dynamic mission planning for multiple mobile robots," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 3. IEEE, 1996, pp. 2396–2401.

[71] ——, "Grammps: A generalized mission planner for multiple mobile robots in unstructured environments," in *Proceedings of the IEEE International Conference on Robotics and Automation.*, vol. 2. IEEE, 1998, pp. 1564–1571.

[72] D. P. Bertsekas, "The auction algorithm: A distributed relaxation method for the assignment problem," *Annals of operations research*, vol. 14, no. 1, pp. 105–123, 1988.

[73] M. B. Dias and A. Stentz, "A free market architecture for distributed control of a multirobot system," in *6th International Conference on Intelligent Autonomous Systems (IAS-6)*, 2000, pp. 115–122.

[74] ——, "Opportunistic optimization for market-based multirobot control," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 3. IEEE, 2002, pp. 2714–2720.

[75] K. Lerman, C. Jones, A. Galstyan, and M. J. Matarić, "Analysis of dynamic task allocation in multi-robot systems," *The International Journal of Robotics Research*, vol. 25, no. 3, pp. 225–241, 2006.

[76] M. M. Zavlanos, L. Spesivtsev, and G. J. Pappas, "A distributed auction algorithm for the assignment problem," in *47th IEEE Conference on Decision and Control, 2008.* IEEE, 2008, pp. 1212–1217.

[77] L. B. Johnson, "Decentralized task allocation for dynamic environments," Ph.D. dissertation, Massachusetts Institute of Technology, 2012.

[78] M. G. Lagoudakis, M. Berhault, S. Koenig, P. Keskinocak, and A. J. Kleywegt, "Simple auctions with performance guarantees for multi-robot task allocation," in *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, vol. 1. IEEE, 2004, pp. 698–705.

[79] R. Zlot, A. T. Stentz, M. B. Dias, and S. Thayer, "Multi-robot exploration controlled by a market economy," in *Proceedings of the IEEE International Conference on Robotics and Automation.*, vol. 3. IEEE, 2002, pp. 3016–3023.

[80] A. Stentz and M. B. Dias, "A free market architecture for coordinating multiple robots," DTIC Document, Tech. Rep., 1999.

[81] N. Kalra, D. Ferguson, and A. Stentz, "Hoplites: A market-based framework for planned tight coordination in multirobot teams," in *Proceedings of the IEEE International Conference on Robotics and Automation.* IEEE, 2005, pp. 1170–1177.

[82] L. B. Johnson, S. S. Ponda, H.-L. Choi, and J. P. How, "Asynchronous decentralized task allocation for dynamic environments," in *Proceedings of the AIAA Infotech at Aerospace Conference*, 2011.

[83] L. Vig and J. A. Adams, "Market-based multi-robot coalition formation," in *Distributed Autonomous Robotic Systems 7.* Springer, 2006, pp. 227–236.

[84] M. B. Dias, R. Zlot, N. Kalra, and A. Stentz, "Market-based multirobot coordination: A survey and analysis," *Proceedings of the IEEE*, vol. 94, no. 7, pp. 1257–1270, 2006.

[85] D. Bhadauria, O. Tekdas, and V. Isler, "Robotic data mules for collecting data over sparse sensor fields," *Journal of Field Robotics*, vol. 28, no. 3, pp. 388–404, 2011.

[86] L. Luo, N. Chakraborty, and K. Sycara, "Provably-good distributed algorithm for constrained multi-robot task assignment for grouped tasks," *IEEE Transactions on Robotics*, vol. 31, no. 1, pp. 19–30, 2015.

[87] A. K. Whitten, H.-L. Choi, L. B. Johnson, and J. P. How, "Decentralized task allocation with coupled constraints in complex missions," in *American Control Conference (ACC), 2011.* IEEE, 2011, pp. 1642–1649.

[88] X. Zheng and S. Koenig, "Generalized reaction functions for solving complex-task allocation problems," in *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, vol. 22, 2011, p. 478.

[89] B. Korte, J. Vygen, B. Korte, and J. Vygen, *Combinatorial optimization.* Springer, 2002.

[90] S. M. LaValle, *Planning algorithms.* Cambridge university press, 2006.

[91] D. Brannan, M. Esplen, and J. Gray, "Geometry.(1999)."

[92] V. V. Vazirani, *Approximation algorithms.* Springer Science & Business Media, 2013.

[93] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[94] 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*, p. 0278364915577958, 2015.

[95] H.-S. Kim and O. Cheong, "The cost of bounded curvature," *Computational Geometry*, vol. 46, no. 6, pp. 648–672, 2013.

[96] C. E. Noon and J. C. Bean, "An efficient transformation of the generalized traveling salesman problem," *INFOR*, vol. 31, no. 1, p. 39, 1993.

[97] R. S. Garfinkel and G. L. Nemhauser, "The set-partitioning problem: set covering with equality constraints," *Operations Research*, vol. 17, no. 5, pp. 848–856, 1969.

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

[99] D. Di Paola, A. Gasparri, D. Naso, and F. L. Lewis, "Decentralized dynamic task planning for heterogeneous robotic networks," *Autonomous Robots*, vol. 38, no. 1, pp. 31–48, 2015.

[100] K. Helsgaun, "Solving the equality generalized traveling salesman problem using the Lin–Kernighan–Helsgaun algorithm," *Mathematical Programming Computation*, vol. 7, no. 3, pp. 269–287, 2015.

[101] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3.2, 2009, p. 5.

[102] R. Bellman, "Dynamic programming and lagrange multipliers," *Proceedings of the National Academy of Sciences*, vol. 42, no. 10, pp. 767–769, 1956.

[103] A. M. Shkel and V. Lumelsky, "Classification of the Dubins set," *Robotics and Autonomous Systems*, vol. 34, no. 4, pp. 179–202, 2001.

[104] S. Hougardy and R. T. Schroeder, "Edge elimination in tsp instances," in *Graph-Theoretic Concepts in Computer Science*. Springer, 2014, pp. 275–286.

[105] C. E. Noon and J. C. Bean, "A lagrangian based approach for the asymmetric generalized traveling salesman problem," *Operations Research*, vol. 39, no. 4, pp. 623–632, 1991.

[106] D. W. Henderson and D. Taimina, *Experiencing geometry*. Prentice Hall, 2000.

[107] A. R. Mosteo and L. Montano, "Simulated annealing for multi-robot hierarchical task allocation with flexible constraints and objective functions," in *Workshop on Network Robot Systems: Toward Intelligent Robotic Systems Integrated with Environments. IROS*. Citeseer, 2006.

[108] A. Aggarwal, D. Coppersmith, S. Khanna, R. Motwani, and B. Schieber, "The angular-metric traveling salesman problem," *SIAM Journal on Computing*, vol. 29, no. 3, pp. 697–711, 2000.

[109] M. Bellmore and S. Hong, "Transformation of multisalesman problem to the standard traveling salesman problem," *Journal of the ACM*, vol. 21, no. 3, pp. 500–504, 1974.

[110] D. Ben-Arieh, G. Gutin, M. Penn, A. Yeo, and A. Zverovitch, "Transformations of generalized atsp into atsp," *Operations Research Letters*, vol. 31, no. 5, pp. 357–365, 2003.

[111] K. Helsgaun, "General k-opt submoves for the Lin–Kernighan TSP heuristic," *Mathematical Programming Computation*, vol. 1, no. 2-3, pp. 119–163, 2009.

[112] M. Athans and P. L. Falb, *Optimal control: an introduction to the theory and its applications.* Courier Corporation, 1966.

[113] C.-H. Ko, Y.-H. Hsieh, Y.-T. Chang, S. K. Agrawal, and K.-Y. Young, "Guidance and obstacle avoidance of passive robot walking helper based on receding horizon control," in *IEEE International Conference on Automation Science and Engineering.* IEEE, 2014, pp. 1032–1037.

[114] B. Kalyanasundaram and K. Pruhs, "Online weighted matching," *Journal of Algorithms*, vol. 14, no. 3, pp. 478–488, 1993.

[115] S. Khuller, S. G. Mitchell, and V. V. Vazirani, "On-line algorithms for weighted bipartite matching and stable marriages," *Theoretical Computer Science*, vol. 127, no. 2, pp. 255–267, 1994.

[116] R. Zlot and A. Stentz, "Market-based multirobot coordination for complex tasks," *The International Journal of Robotics Research*, vol. 25, no. 1, pp. 73–101, 2006.

[117] L. B. Johnson, S. Ponda, H.-L. Choi, and J. P. How, "Improving the efficiency of a decentralized tasking algorithm for uav teams with asynchronous communications," in *AIAA Guidance, Navigation, and Control Conference (GNC)*, vol. 5.2, 2010, pp. 5406–5411.

APPENDICES

Appendix A

Proof of Results

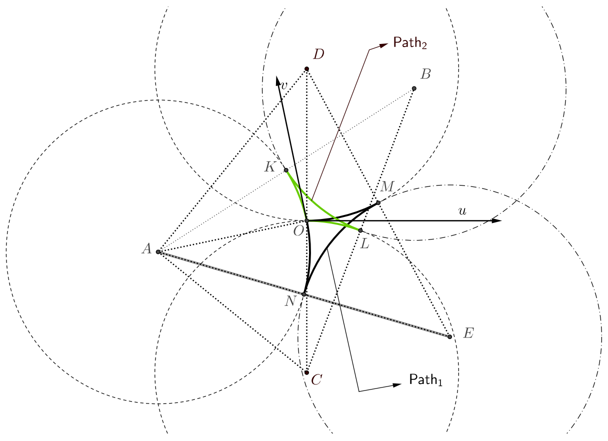
A.1 A Distributed Task Allocation and Sequencing Algorithm for Robots with Differential Constraints

Proof. [Proof of Theorem 4.2.7] 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:

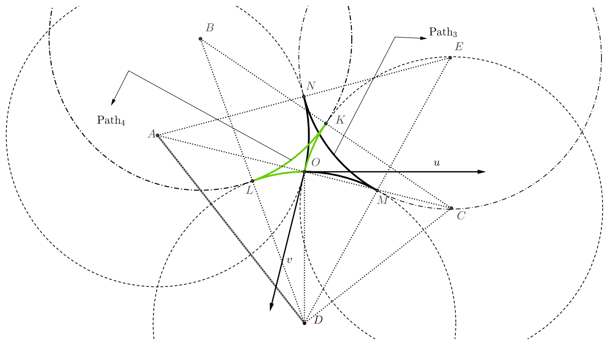
- Rotate by α, translate a distance $\text{Euc}(q_1, q_2)$, and rotate by β; and
- Rotate by $\pi - \alpha$, translate a distance $\text{Euc}(q_1, q_2)$ with reverse gear, and rotate by $\pi - \beta$.

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.1a consists of arcs $\overset{\frown}{OM}^+, \overset{\frown}{MN}^-$ and $\overset{\frown}{NK}^+$ is the feasible path with length $R_{\min}\theta$.



(a) 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.



(b) 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.

Superscripts on the arcs shows the forward and reverse gear motions.

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

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.1a.

With this result, same path construction procedure for DD robot 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 $R_{\min}\pi + \operatorname{Euc}(q_1,q_2)$. □

A.2 On Efficient Computation of Shortest Dubins Paths Through Three Consecutive Points

To prove Proposition 5.3.1 we require the following result.

Lemma A.2.1 (Minimum radius of an inverted circle)**.** *The radius of the inverted circles* circle(C,R) *and* circle(D,R) *in the optimal path are greater than* $\frac{R_{\min}}{4}$.

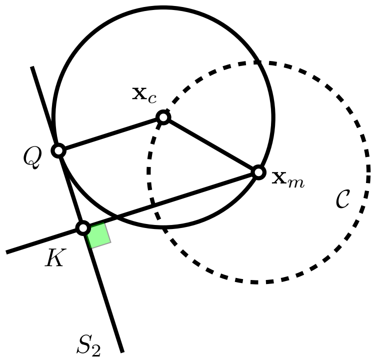


Figure A.2: Line segment S_2 tangent to the circle $\mathbf{x}_c$ at Q. The closest pointon S_2 to the center of inversion $\mathbf{x}_m$ is K.

Proof. There are two cases that we need to consider regarding the relation of either of the line segments (S_2 or S_4) and the circle of inversion as follows: Case 1) The line tangent to circle($\mathbf{x}_c, R_{\min}$) intersects the circle of inversion circle($\mathbf{x}_m, R_{\min}$). Then the inverted circle contains the intersection points and the center of the circle of inversion. The radius of the inverted circle is minimum when the points of intersection are the same, i.e., the line is tangent to the circle of inversion. The corresponding inverted circles have a radius equal to $\frac{1}{2}R_{\min}$.

Case 2) The tangent line does not intersect the circle of inversion. In this case the maximum distance between the center of inversion and one of the tangent lines (blue or red line in Figure 5.3 for instance) is $2R_{\min}$.

Figure A.2 shows that the distance of the center of inversion $\mathbf{x}_m$ from the tangent line to circle($\mathbf{x}_c, R_{\min}$) is not greater than $|\overline{\mathbf{x}_m Q}|$. The maximum distance of the point $\mathbf{x}_m$ from the line tangent to the circle $\mathbf{x}_c$ occurs when the points Q and K are the same, which implies that the maximum distance is bounded by $2R_{\min}$. Thus, the inverse of the point K with respect to the circle $\mathcal{C}$ is within distance $\frac{R_{\min}}{2}$ of $\mathbf{x}_m$. Note that both $\mathbf{x}_m$ and the inverse of Q are in the inverse of S_2 with respect to circle($\mathbf{x}_m, R_{\min}$). The product of the inversion is a circle such that the inverse of K is the farthest point to $\mathbf{x}_m$ (K is the closest point on l to $\mathbf{x}_m$). Therefore, the radius of the inverted circle is greater than $\frac{R_{\min}}{4}$. □

Proof. [Proof of Proposition 5.3.1

For a path contained in $\{RSRSR, LSLSL, RSLSR, LSRSL\}$, Equations (5.2)-(5.4) simplify to the following:

$$|\overline{A\mathbf{x}_m}|\cos(\beta_1 - \theta + \alpha) = |\overline{B\mathbf{x}_m}|\cos(\beta_2 + \theta + \alpha). \tag{A.1}$$

With the assumption of case (i) ($|\overline{A\mathbf{x}_m}| = |\overline{B\mathbf{x}_m}|$) and by Equation (A.1), the optimal heading is equal to the approximated heading, namely $\alpha = \frac{\beta_1-\beta_2}{2}$.

Case (ii): Under the distance constraint of Problem 5.1.1 ($4R_{\min}$), the approximated heading has maximum deviation from the optimal heading when the radius R reaches infinity or its lower bound from Lemma A.2.1, namely $\frac{1}{4}$. For the path types $RSRSR$ and $LSLSL$, substituting the minimum and maximum R values in the Equations (5.2) and (5.3) result the following:

$$-\alpha + \beta_1 + \theta \in \left[\cos^{-1}\left(\frac{1}{3}\right), \cos^{-1}\left(\frac{-1}{3}\right)\right],$$
$$\alpha + \beta_2 + \theta \in \left[\cos^{-1}\left(\frac{1}{3}\right), \cos^{-1}\left(\frac{-1}{3}\right)\right].$$

Therefore, the maximum deviation of the approximated heading from the optimal heading is in the range $[\frac{-\pi}{9}, \frac{\pi}{9}]$.

Case (iii): using similar argument:

$$-\alpha + \beta_1 + \theta \in \left[\cos^{-1}(1), \cos^{-1}\left(\frac{1}{3}\right)\right],$$
$$\alpha + \beta_2 + \theta \in \left[\cos^{-1}(1), \cos^{-1}\left(\frac{1}{3}\right)\right].$$

Therefore, the optimal heading lies within the interval $[-\frac{\pi}{5}, \frac{\pi}{5}]$ centered at the approximated heading.

Case (iv): We prove the bound for path types in $\{RSRSL, LSLSR\}$ and the other path types directly follow same analysis. In these types of paths, substituting the upper and lower bounds of R in Equations (5.2)-(5.4) yields the following:

$$\frac{-1}{3} \leq \cos(\beta_1 + \theta - \alpha) \leq \frac{1}{3},$$
$$\frac{1}{3} \leq \cos(\beta_2 + \theta + \alpha) \leq 1.$$

Thus,

$$\beta_1 + \theta - \alpha \in \left[\frac{\pi}{2} - \frac{\pi}{9}, \frac{\pi}{2} + \frac{\pi}{9}\right], \quad \beta_2 + \theta + \alpha \in \left[0, \frac{\pi}{2} - \frac{\pi}{9}\right].$$

Therefore, the maximum error of $\bar{\alpha}$ is $\frac{11\pi}{36}$. □