Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Computer Science and Engineering

DIPLOMA THESIS ASSIGNMENT

Student: **Bc. Petr Váňa**

Study programme: Open Informatics
Specialisation: Artificial Intelligence

Title of Diploma Thesis: **Path Planning for Non-holonomic Vehicle in Surveillance Missions**

Guidelines:

* Familiarize yourself with the surveillance mission planning that is formulated as a variant of the traveling salesman problem with neighborhoods (TSPN) [1].
* Consider Dubins curves as paths in the mission planning formulated as the Dubins TSP [2].
* Study current approaches for the DTSP and its extension called the DTSPN and describe selected approaches, e.g., [3,4,5].
* Implement selected approaches and compare their performance regarding the computational requirements and quality of solution.
* Design an extension of the existing approaches for the DTSPN with obstacles.

Bibliography/Sources:

[1] J. Faigl, V. Vonásek, L. Preucil: Visiting convex regions in a polygonal map. Robotics and Autonomous Systems 61(10): 1070-1083 (2013)
[2] K. Savla, E. Frazzoli, F. Bullo: Traveling Salesperson Problems for the Dubins Vehicle. IEEE Trans. Automat. Contr. 53(6): 1378-1391 (2008)
[3] J. T. Isaacs, J. P. Hespanha: Dubins Traveling Salesman Problem with Neighborhoods: A Graph-Based Approach. Algorithms 6(1): 84-99 (2013)
[4] D. G. Macharet, A. A. Neto, V. F.-C. Neto, M. F. M. Campos: An evolutionary approach for the dubins' traveling salesman problem with neighborhoods. GECCO 2012: 377-384
[5] K. J. Obermeyer, P. Oberlin, and S. Darbha, Sampling-based path planning for a visual reconnaissance unmanned air vehicle, Journal of Guidance, Control, and Dynamics, 35(2): 619-631, (2012)

Diploma Thesis Supervisor: doc. Jan Faigl Ing., Ph.D.

Valid until the end of the summer semester of academic year 2015/2016

[illegible]

doc. Ing. Filip Železný, Ph.D.
Head of Department

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta elektrotechnická
L.S.

[illegible]

prof. Ing. Pavel Ripka, CSc.
Dean

Prague, April 1, 2015

Czech
Technical
University
in Prague

Faculty of Electrical Engineering
Department of Computer Science

Master's thesis

Path Planning for Non-holonomic Vehicle in Surveillance Missions

Bc. Petr Váňa

May 2015

Thesis supervisor: doc. Ing. Jan Faigl, Ph.D.

Prohlášení

Acknowledgement

I would like to thank to my supervisor doc. Ing. Jan Faigl, Ph.D. for the idea of this theme, for his valuable recommendations, and finally for his patience with me.

Abstract

In this thesis, we address the problem of optimal path planning for a non-holonomic vehicle in surveillance missions. We consider car-like vehicles with a limited turning radius, modeled as the Dubins vehicle. We formulate the surveillance mission as a multi-goal path planning problem to visit a set of regions by Dubins vehicle, which is also known as the Dubins Traveling Salesman Problem with Neighborhoods (DTSPN). Several approaches for solving this infinite combinatorial optimization problem can be found in literature, including genetic algorithms or reduction to other variants of the TSP. We study the properties of the optimal path for the DTSPN and we provide detailed analysis of the optimal solution of the restricted problem where regions are in a distance longer than four times of the turning radius. Based on this analysis, we propose a new local iterative optimization procedure to find Dubins path visiting the regions. Experimental results indicate that the procedure provides better solutions with lower computational requirements than existing approximation approaches.

Abstrakt

Diplomová práce se zabývá plánováním cesty pro neholonomní vozidla v úlohách dohledu. V práci uvažujeme vozidlo s omezených poloměrem zatáčení, které modelujeme jako Dubinsovo auto. V práci definujeme problém dohledu jako úlohu plánování cesty přes více cílů (oblastí) pro Dubinsovo auto. Tento problém je znám jako Dubinsův problém obchodního cestujícího s okolím (DTSPN). V literatuře můžeme nalézt několik přístupů k řešení tohoto nekonečného kombinatorického problému. Například řešení založené na genetických algoritmech nebo přístup využívající transformace problému na varianty úlohy obchodního cestujícího. V této práci studujeme vlastnosti optimálního řešení DTSPN, které jsou následně využity v návrhu nového způsobu řešení založeného na dvoufázovém přístupu. Nejprve je určeno pořadí cílových regionů řešení úlohy eukleidovského obchodního cestujícího a následně je použita nově navržená iterativní optimalizační procedura, která hledá nejkratší cestu pro Dubinsovo auto procházející všemi cílovými oblastmi. Navržený algoritmus je porovnán s ostatními existujícími přístupy. Z výsledků vyplývá, že námi navržený algoritmus poskytuje řešení srovnatelné kvality s ostatními přístupy, přičemž je výrazně méně výpočetně náročný.

Contents

List of Figures

List of Tables

List of Algorithms

CHAPTER 1

Introduction

During the last decade, the number of applications with unmanned vehicles is growing rapidly. These vehicles can be used in many different missions by both civilian and defense organizations and can operate in a semi or fully autonomous mode. It allows to accomplish missions more effectively and potentially reduce the mission cost. One of the common type of tasks for unmanned vehicles are surveillance missions in which a vehicle is utilized to gather required information about areas of interest by providing online (usually visual) information of the selected locations. In surveillance missions, the vehicles can be equipped with various types of sensors (e.g., camera, elector-magnetic, sonic, etc.). The sensors have particular limitations, but all of them are limited in the field of view and maximal sensing range from which collected measurements provide information of sufficient quality.

Having areas of interest, a vehicle has to follow a path (trajectory) such that it is able to collect the required information about the areas by performing particular measurements along the path, i.e., at the sensing locations. Thus, the surveillance mission is a problem to find a cost efficient solution of the information gathering problem, where the cost can be the time needed to collect the required information. Such a cost can consist of the time to perform the particular measurements and the time needed to travel to the sensing locations. Regarding the sensing locations, we can identify two types of surveillance missions.

The first type of surveillance missions are inspection tasks where we need to determine the sensing locations to provide the requested information. Here, we further distinguish situations where the cost of a sensor measurement is high, e.g., because of a long duration of measurement or a limited storage capacity. In such a case, the total cost mostly depends on the number of sensing locations and thus the inspection task can be formulated as the problem to minimize the number of sensing locations, where the measurements provide the required information. This problem can be formulated as a variant of the *art gallery problem*. This visibility problem originates in computational geometry and it is inspired by a problem of guarding an art gallery with the minimal possible number of stationary guards who observe particular parts of the gallery [1] and thus all together provide continuous surveillance of the whole gallery. The stationary surveillance problem is a limiting case where we have the same number of sensing vehicles as the minimal number of sensing locations. The opposite limiting case is for a single vehicle, where it is required to plan a path to visit the sensing locations once they are determined.

Nowadays, cameras are already digitized and also memory storage has enough capacity to save continuous streams from visual sensors. Hence, the sensing system is able to provide continuous streams of data almost at zero cost. Therefore, the cost function of the surveillance mission consists mainly of the length of the path to collect information about the requested areas of interest. In this case, the problem to collect visual information about the areas of interest can be formulated as the *coverage path problem*. Beside surveillance missions with visual sensing, other robotic tasks as cleaning, painting or plowing can be considered as this problem. The fundamental approach for coverage was published in [2], where the author proposed the boustrophedon cellular decomposition. Another variant of the coverage path problem is the *coverage sampling problem* (CSP) [3], where particular sensing locations are determined. In the CSP, each sample corresponds to a single measurement providing a partial information about the object or area of interest similarly to the art gallery problem. The main difference from the art gallery problem is that the CSP uses a more than the minimal number of sensing locations and thus the problem to collect the required information at minimal cost also includes optimization of the path cost together with the selection of the most suitable subset of the sensing locations.

The second type of surveillance missions are problems where we assume the particular sensing locations, where performed measurements provide the requested information, are already given and the problem can be formulated as the *multi-goal path planning problem* (MPP) [4, 5]. The problem is defined by a set of goals (targets) that are represented by points or geometrical regions, e.g., discs, convex polygonal areas, etc. In the MPP, the goals can be generally visited in an arbitrary order and the problem is to determine an order of the visits to the goals such that the final path has the minimal cost. In a case the goals are represented by points and the vehicle used to visit the goals satisfies holonomic constraints, the MPP can be considered as the combinatorial *traveling salesman problem* (TSP). The TSP stands to find a shortest path to visit the given set of goals (cities), such that each goal is visited exactly once and the vehicle returns to the starting goal (city). It is known the TSP is NP-hard, and it can be considered as one of the well studied problem in operational research [6].

In practical scenarios of surveillance missions addressed as the MPP, the task is to sense a given set of objects of interest by the used visual sensor or it can also be the case to retrieve data from deployed sensors using wireless communication. Thus, the measurement can be performed from a limited distance and it is not necessary to precisely visit the goal location. Then, we can extend the goal location represented as a point to a goal region. This extension enables a vehicle to fulfill the mission by visiting any point of each given region. Hence, the traveling salesman problem is transformed to the *traveling salesman problem with neighborhoods* (TSPN).

For the purpose of this thesis, we formulate the surveillance mission as a multi-goal planning problem for a nonholonomic vehicle. An example of nonholonomic vehicle is a fix-wing unmanned aerial vehicle (UAV), for which the nonholonomic constraint is the limited turning radius. There can be several approaches how to address challenges arising from the non-holonomic constraint. Since we focus the thesis to aerial vehicles, we consider the vehicle as the Dubins vehicle [7], which is a widely used model for path planning for fix-wing aerial vehicles. Regarding the aforementioned overview of the related problems, a surveillance mission for Dubins vehicle can be considered as the *Dubins traveling salesman problem with neighbourhoods* (DTSPN) [8]. In the DTSPN, a set of goal regions to be visited by the vehicle is given and the problem is to find a cost efficient tour path for the Dubins vehicle to visits all the goal regions.

The main challenge of the DTSPN is related to the nonholonomic constraint of the vehicle and the combinatorial nature of the problem. It is necessary to determine the sequence to visit the regions, the points at which the vehicle visits the regions, and finally also the orientation of the vehicle at these points since Dubins vehicle cannot turn arbitrarily at a single point. One can image that we can find a feasible solution of the DTSPN by a subsequent solution of these particular sub-problems. However, if we like to find an optimal solution, we cannot do that separately, because these problems are mutually dependent. We need particular points to visit the regions and their orientations to determine the optimal sequence, e.g., as a solution of the TSP. However, such a position of the entry point can be selected as any point on the boundary of the regions. Moreover, the orientation of the vehicle at this entry point can be also selected arbitrarily. Thus, we call the problem infinite combinatorial optimization problem and we cannot simply use existing discrete combinatorial approaches for the TSP. Therefore, it is desirable to address the problem differently. In the thesis, we study this challenging problem and we propose a new, fast, and efficient any-time algorithm, which provides competitive solutions to existing approaches while its computational requirements are significantly lower.

The thesis is organized as follows. In Chapter 2 we familiarize the reader with the basic terms and ideas, which are used further in the text. The DTSPN is formally defined in Chapter 3 and analyzed in Chapter 4. Existing approaches are divided into three classes and described in Chapter 5. In Chapter 6, we propose a new decoupled approach to address the DTSPN. It has been designed on the top of the analyzed properties of the optimal solution for the DTSPN. We tested our new local optimization algorithm, compared it with existing approaches, and the achieved results are presented in Chapter 7. We also designed extensions of existing approaches to deal with obstacles in Chapter 8.

CHAPTER 2

Background

In this chapter, we familiarize the reader with the basic terms and ideas used in the further chapters of the thesis. More experienced readers can simple jump over this part. The chapter is divided into two sections. In the first section, we formulate path planning problem and further focus on path planning for vehicles with the limited turning radius. Subsequently, in the second section, we define the traveling salesman problem and introduce its the most related variants.

2.1 Path planning

A fundamental task in robotics it is to find a collision-free path for a robot from an initial configuration to the desired goal configuration among a collection of static obstacles. This *Geometric Path Planning Problem* is also known as the *Piano Mover's Problem* [9]. We use the definition of the planning problem from [10] and modify it according to the addressed surveillance mission. In the planning problem, the robot operates in a *word* $\mathcal{W}$ which is either a two-dimensional or a three-dimensional space ($\mathcal{W} = \mathbb{R}^N$). The robot $\mathcal{A}$ is represented by a rigid body defined in the world. A position and orientation of the robot is specified by a *configuration* q. The space of all possible configurations of the robot in the world is the *configuration space* $\mathcal{C}$, or C-space. The dimension of the C-space is the number of degrees of freedom of the robot. An *obstacle region* is represented by the closed set $\mathcal{O} \subset \mathcal{W}$, which usually contains polyhedrons, spheres, or piecewise-algebraic surfaces. From this, the *C-space obstacle region* $\mathcal{C}_{obs}$ can be defined as follows:

$$\mathcal{C}_{obs} = \{q \in \mathcal{C} | \mathcal{A}(q) \cap \mathcal{O} \neq \emptyset\}. \tag{2.1}$$

The *free space* $\mathcal{C}_{free}$ is the set of all robot configurations that are not in collision with the obstacle region:

$$\mathcal{C}_{free} = \mathcal{C} \setminus \mathcal{C}_{obs}. \tag{2.2}$$

Using the definitions stated above, we formulate the *Geometric Path Planning Problem* as follows:

Problem 1 (Geometric Path Planning Problem)
Given:

1. *A world $\mathcal{W}$ in which either $\mathcal{W} = \mathbb{R}^2$ or $\mathcal{W} = \mathbb{R}^3$.*
2. *A set of obstacle regions $\mathcal{O} \subset \mathcal{W}$.*
3. *A robot $\mathcal{A}$ defined in $\mathcal{W}$.*
4. *A configuration space $\mathcal{C}$ determined by the set of all possible transformations that may be applied to the robot. From this, $\mathcal{C}_{obs}$ and $\mathcal{C}_{free}$ are derived.*
5. *An initial configuration $q_I \in \mathcal{C}_{free}$.*
6. *A goal configuration $q_G \in \mathcal{C}_{free}$.*

The task is to find a continuous path τ: $\langle 0, 1\rangle \rightarrow \mathcal{C}_{free}$, which starts at the initial configuration $\tau(0) = q_I$ and ends at the goal configuration $\tau(1) = q_G$.

A special variant of path planning problem is planning with nonholonomic vehicles. It is complicated by an additional nonholonomic constraints, which restrict the search space.

2.1.1 Nonholonomic vehicle

We can divide vehicles into two main groups holonomic and non-holonomic. A holonomic vehicle is able to control all degrees of freedom independently. An example of a holonomic vehicle is a robot with three omnidirectional wheels. It can fully control its movement in all three degrees of freedom and execute any collision-free path. By contrast, a vehicle is non-holonomic if the controllable degrees of freedom are less than the total degrees of freedom. An alternative definitions of the non-holonomic vehicle can be found in [10]. An example of a non-holonomic vehicle is a standard car used every day by billions of people all over the world. It has limited turning radius and that is the reason why it is so complicated to learn parallel parking.

The nonholonomic constraint restrict the movement of the vehicle. Therefore, the possible transformations inside the configuration space are also limited. This forces the path planners to consider the whole configuration of the robot, instead of the position only.

2.1.2 Dubins vehicle

A commonly used model for both aerial and ground non-holonomic vehicles is the model of Dubins vehicle. It models a vehicle which goes only forward at a constant speed and has a limited minimum turning radius ρ. Dubins vehicle is suitable mainly for car-like vehicles or fix-wings aerial vehicles. An example of the Dubins vehicle represented by a car-like vehicle is depicted in Fig. 2.1.

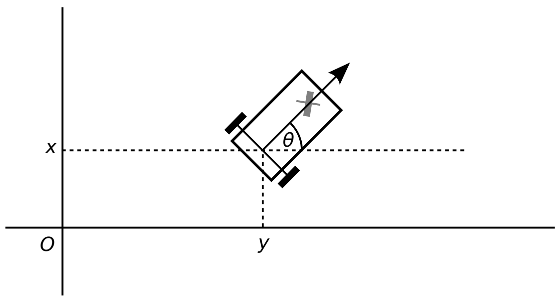


Figure 2.1: Dubins vehicle represented by a car-like vehicle

A configuration of the Dubins vehicle is specified by the position (x, y) and the orientation θ. Mathematical model of the Dubins vehicle can by formulated as follows:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = v \begin{bmatrix} \cos\theta \\ \sin\theta \\ \frac{u}{\rho} \end{bmatrix}, \quad |u| \leq 1, \tag{2.3}$$

where v is the constant forward velocity, ρ is the minimum turning radius, and u is the bounded control input. It is possible to normalize both vehicle velocity and minimum turning radius by scaling the space ($x' = \frac{x}{\rho}$, $y' = \frac{y}{\rho}$, $t' = \frac{t}{v}$) and thus simplify the mathematical model of the Dubins vehicle into the following alternative form:

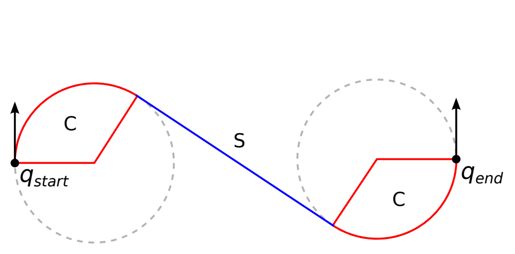
$$\begin{bmatrix} \dot{x}' \\ \dot{y}' \\ \dot{\theta}' \end{bmatrix} = \begin{bmatrix} \cos\theta' \\ \sin\theta' \\ u \end{bmatrix}, \quad |u| \leq 1. \tag{2.4}$$

2.1.3 Dubins maneuver

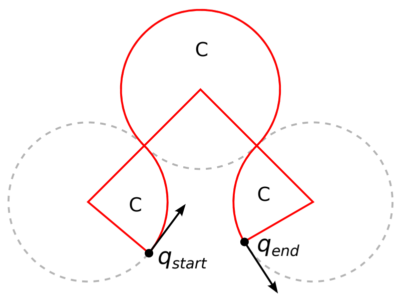
Properties of paths with a minimum turning radius were first studied in 1957 by Dubins [7]. He proved by geometric arguments that the path with a minimum possible length between two configurations, each defined by position and direction, can contain only line segments and arcs with exactly the minimum turning radius. Therefore we call the minimum length path Dubins maneuver. It was further shown that only 6 maneuvers of 2 types can be optimal:

- **CCC type:** LRL, RLR;
- **CSC type:** LSL, LSR, RSL, RSR.

Here, C denotes the arc with the radius equal to the minimum turning radius ρ and S denotes a straight line segment. Two orientations of C segments are possible: L for the left turn and analogously R for the right turn. Examples of CSC and CCC maneuver are depicted in Fig. 2.2. We can notice that for the situation in Fig. 2.2b it is not possible to construct RSR maneuver due to the limited turning radius and thus the optimal maneuver is the CCC.



(a) CSC maneuver



(b) CCC maneuver

Figure 2.2: Two types of Dubins maneuver

A problem of finding Dubins maneuver was also addressed in [11] where authors formulated conditions that determine which of the six possible maneuvers is the shortest one. Dubins maneuver is the shortest path between two vehicle configuration, but two configurations have 6 degrees of freedom in total. Luckily, we can choose coordinate system in such a way that the start configuration lies on the origin and the direction angle is zero (parallel to x axis), i.e., $q_{start} = \vec{0}$. This transformation reduces the input space into 3 degrees of freedom. Now we can simply plot it for some fixed turning angle θ_{end}. An example for $\theta_{end} = \pi/3$ is plotted in Fig. 2.3. Here, each type of the Dubins maneuver is highlighted by a different color. Curves in the figure represent areas with the same length of the Dubins maneuver. Hence, we can introduce new function $\mathcal{L}$ to denote the length of the Dubins maneuver for any configuration q_{end} in the normalized coordination system.

$$\mathcal{L}: \quad \mathbb{R}^2 \times \mathbb{S}^1 \to \mathbb{R} \tag{2.5}$$

Even though the function $\mathcal{L}$ is strictly continuous inside regions with the same type, it can have discontinuity in the places where CSC maneuver is changing to CCC maneuver. Since the CCC maneuver exists only if the Euclidean distance between the two configurations is shorter than 4ρ, the function $\mathcal{L}$ is strictly continuous for all configurations which are at least 4ρ apart [11].

2.2 Traveling salesman problem

The traveling salesman problem (TSP) is one of the most studied combinatorial problem in operational research. The TSP stands to find the shortest tour visiting each city exactly ones. The set of cities (locations to be visited) is given and it is further assumed that the particular cost to travel from one city to another city is also given, e.g., in a form of a cost (distance) matrix. The TSP is a combinatorial optimization problem, which is proved to be NP-hard [12]. It is unclear when the TSP was first formulated, but first attempts to address this problem was in mid-20th century in [13] and [14]. The TSP can be defined by the cost matrix C which contains distances c_{ij} between

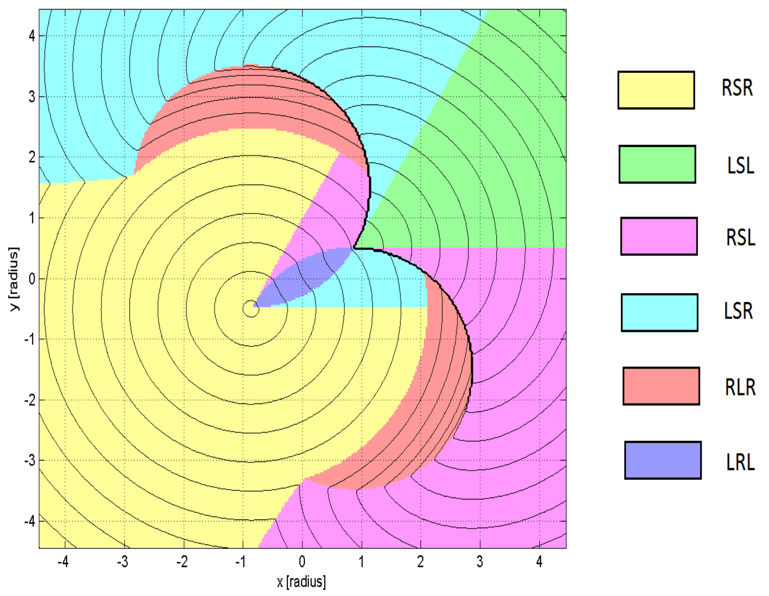


Figure 2.3: Distribution of Dubins maneuver types where $\theta_{end} = \pi/3$

each pair of cities. This matrix can be both symmetric and asymmetric. The matrix is symmetric if Equation 2.6 holds, otherwise the matrix is asymmetric. According to this property the problem is called symmetric traveling salesman problem (TSP) or asymmetric traveling salesman problem (ATSP).

$$c_{i\,j} = c_{j\,i}, \quad \forall\, i,\, j \tag{2.6}$$

Another important property of the TSP is the triangle inequality. It basically means that there does not exist a shortcut between two cities which goes through the third one and the shortcut is strictly shorter than the direct connection between the two cities. This condition can be formulated as Equation 2.7. If this condition holds we call the TSP is metric.

$$c_{i\,k} \le c_{i\,j} + c_{j\,k}, \quad \forall\, i,\, j,\, k \tag{2.7}$$

A special case of the cost function for the TSP is Euclidean distance. Then, we call such a problem *Euclidean Traveling Salesman Problem* (ETSP). It is obvious that the ETSP is both symmetric and metric. An important property of the solution of any ETSP instance is that the solution does not contain any self-intersection.

To compare different algorithms it is necessary to have enough representative instances of the TSP. For this purpose, the TSPLIB library [15] can be used. It contains many instances for the TSP from various sources and of various types.

2.2.1 Exact algorithms

The exact algorithms are guaranteed to find the optimal solution in a finite number of steps. On of such algorithms is a brute force exhaustive search. Unfortunately, this approach is not applicable for very large instances because it is too computationally demanding using now days computers. Another method is to formulate the TSP as a problem of the *integer linear programming* (ILP). One way how to formulate the TSP as the ILP problem is to use Boolean variable $x_{i\,j}$ for oriented connection between two cities:

$$x_{i\,j} = \begin{cases} 1 & \text{the path goes from city } i \text{ to city } j \\ 0 & \text{otherwise} \end{cases} .$$

Then, it is necessary to restrict the number of input and output edges connected to a city. These variables are used to ensure that there exist only closed tours; however, it cannot eliminate creating more smaller separated subtours. To effectively eliminate the subtours Miller et al. proposed a new formulation of the TSP as the ILP problem [16]. They suggested to add new variables u_i and formulated the TSP as Problem 2.

Problem 2 (TSP - MTZ formulation)

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{i,j}\, x_{i,j}$$

$$\begin{aligned} \textit{subject to:} \quad \sum_{j=1}^{n} x_{i,j} &= 1, && \forall i \in \{1, \dots, n\} \\ \sum_{i=1}^{n} x_{i,j} &= 1, && \forall j \in \{1, \dots, n\} \\ x_{i,j} &\in \{0,1\}, && \forall i,j \in \{1, \dots, n\} \\ \\ u_1 &= 1 \\ 2 \le u_i &\le n, && \forall i \in \{2, \dots, n\} \\ u_i - u_j + 1 &\le (n-1)(1 - x_{ij}), && \forall i,j \in \{2, \dots, n\} \end{aligned}$$

The MTZ formulation eliminates subtours by adding $\mathcal{O}(n)$ extra variables and $\mathcal{O}(n^2)$ constraints. The ILP problem can be further solve by a branch and bound algorithm [17]. An example of the state-of-the-art exact algorithm is Concorde solver[18], which is publicly available. It can solve problems with hundreds of nodes in few seconds to optimum using standard desktop computer.

2.2.2 Approximate algorithms

Approximate algorithm solves the problem while it can guarantee the approximation factor of the found solution for every instance of the given problem. An example of an approximate algorithm for the TSP is the well known Christofides' algorithm for which the approximate factor is equal to $\frac{3}{2}$.

2.2.3 Heuristic algorithms

For larger instances of the TSP, or in a lack of time, heuristic algorithms are suitable choice. Although they do not guarantee the optimal solution would be found, they can find solution of high quality in much shorter time. Some of them can even find the optimal solution with a high probability. According to [19], there are three classes of heuristic algorithms:

- Tour construction algorithms;
- Tour improvement algorithms;
- Composite algorithms.

The tour construction algorithms incrementally build a tour by adding a new city in each iteration. A simple example is the nearest-neighbour algorithm which in each step adds a nearest unused city. This greedy way is fast and generates only feasible solutions but not optimal in average.

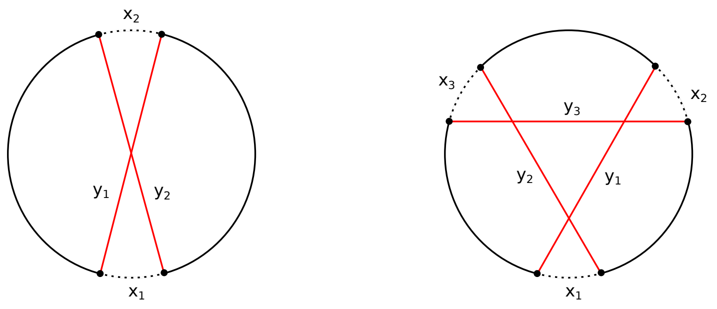
The tour improvement algorithms try to repeatedly modify a tour in order to shorten the length of the tour until there is no such modification that will improve the tour. Probably the simplest tour improvement algorithm is the 2-opt algorithm. The algorithm starts with a given tour and iteratively evaluates possible exchanges of the connections between two edges in the tour.

The composite algorithms combine both previous approaches. An initial tour is found by a construction algorithm and improvement algorithms continue in trying to find better solution. This method shares all benefits from both previous approaches. The initial tour is quickly found and the algorithm continues in finding a better solution in a case of enough computational time.

2.2.4 λ-opt approximate algorithm

An example of approximate algorithm is the 2-opt algorithm. It iteratively improves the tour by 2-opt move where the initial tour is given. If there is no improving 2-opt move the algorithms terminates. The 2-opt move exchanges a pair of edges by another pair which creates a new tour. An example of the 2-opt move is depicted in Fig. 2.4a. The black circle represents the original tour, where the doted edges x_1 and x_2 are removed and replaced by the red edges y_1 and y_2. It is necessary to make sure that disconnected sub-tours are not generated.

The 2-opt algorithm can be generalized by changing the number of exchanged edges λ and the exchange procedure is then called the λ-opt algorithm. An example of the 3-opt move in the 3-opt algorithm is depicted in Fig. 2.4b. The λ-opt algorithm returns the λ-optimal tour. A tour is said to be λ-optimal if it is impossible to obtain a shorter tour by replacing any λ of its links by any other set of λ links [20]. From this definition, it is obvious that a tour containing n cities is optimal if and only if the tour is n-opt. It is also easy to see that any λ-opt tour is also λ'-opt for each λ' satisfying $1 \leq \lambda' \leq \lambda$.



(a) 2-opt move (b) 3-opt move

Figure 2.4: k-opt moves for TSP

A higher the λ is, a higher is the probability of finding the optimal solution by the λ-opt algorithm. Further, the cases when $\lambda = 4, 5$ were studied in [21]. Although, increasing of λ improves the probability of finding a good solution, the computational time grows rapidly as the number of cities increases. A naive implementation of the λ-opt algorithm has a time complexity of $\mathcal{O}(n^{\lambda})$. That is why 2-opt and 3-opt are the most commonly used as a suitable balance between the solution quality and the required computational time.

2.2.5 Lin-Kernighan heuristic

Another heuristic algorithm for the TSP is the Lin-Kernighan heuristic algorithm which was first published in [19]. It is based on the λ-opt algorithm. The key idea is to use a variable λ for the λ-opt move. The algorithm tries all possible moves in each step and recursively all possible moves from improved tours. Any exchange can be represented by the two sets $X = \{x_1, \dots, x_\lambda\}$ and $Y = \{y_1, \dots, y_\lambda\}$ where X and Y stands for a set of removed edges and added edged, respectively. To reduce the number of possible moves sequentially edge by edge, four following properties are checked. If any property does not hold the move is not further considered.

1. *The sequential exchange criterion:* In a sequence of move $(x_1, y_1, x_2, y_2, \dots, x_\lambda, y_\lambda)$, all adjoining edges must share the same city. A necessary condition is that the sequence of edges makes a closed chain.

2. *The feasibility criterion:* It is required that if a smaller number of m edges from the move are used the last edge y_m closes the edges into a feasible tour. This criterion significantly reduce a running time.

3. *The positive gain criterion:* We define the gain G_i by weight of edges $G_i = (g_1 + g_2 + \cdots + g_i)$, where g is the change in the total cost by exchange of one pair of the edges $g_i = c(x_i) - c(y_i)$. To improve the tour length it is necessary the gain G_i is positive. The Lin-Kernighan algorithm also requires all partial sums to be positive. This criterion makes the algorithm effective but is not restrictive. There always exist any permutation which meets this criterion.

4. *The disjunctivity criterion:* Once the edge is removed it cannot be added back again. Hence, X and Y are disjoint sets.

These criteria are described in more detail in [20]. The author significantly improved effectiveness of the Lin-Kernighan and estimated the time complexity of the procedure to be approximately $\mathcal{O}(n^{2.2})$. He implemented this algorithm and make it publicly available as the LKH solver [22].

2.2.6 Generalized traveling salesman problem

The *Generalized traveling salesman problem* [23] (GTSP) is a variant of the TSP, in which the cities are partitioned into mutually exclusive sets. The task is to find a tour which visit exactly one city in each set. The GTSP can be both symmetric and asymmetric. An example instance of the GTSP is depicted in Fig. 2.5, where p denotes a city and S is a set of cities. Several approaches were proposed to address this problem. It can be transformed by the Noon-Bean transformation [24] into the ATSP, see Section 5.3. Another approaches are based on memetic algorithm [25] and modified Lin-Kernighan heuristic for the GTSP [26].

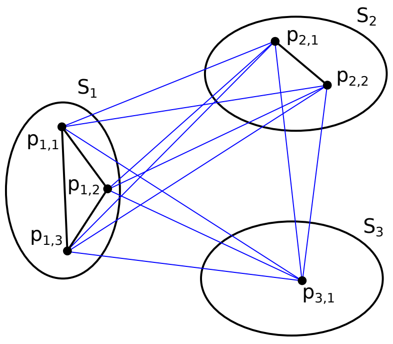


Figure 2.5: An instance of the GTSP

CHAPTER 3

Problem statement

There exist many types of surveillance missions, but in this thesis, we focus on the *multi-goal path planning problem* (MPP) for the Dubins vehicle. In this problem, the Dubins vehicle is requested to visit a given set of goals. The goals can be represented by points or regions. In the case points are used, the MPP problem can be considered as the *Dubins traveling salesman problem* (DTSP). A more generous variant of the DTSP is the *Dubins traveling salesman problem with neighborhoods* (DTSPN) in which the set of point goals is replaced by a set of goal regions. These two problems share similar properties because the DTSP can be considered as a sub-problem of the DTSPN.

The Dubins vehicle is considered in a two-dimensional world that is represented as a plane $\mathcal{W} \subseteq \mathbb{R}^2$. The Dubins vehicle is a non-holonomic vehicle which can go only forward at a constant speed and it has a limited turning radius ρ. For simplicity and without loss of generality, if it is not explicitly stated, we consider ρ=1 in the rest of this text. The state of the vehicle is defined by the position $p \in \mathcal{W}$ and the heading $\theta \in \mathbb{S}^1$. Hence, the configuration space of the vehicle $\mathcal{C} = SE(2)$ has three dimensions. The shortest path between two configurations $q_1, q_2 \in \mathcal{C}$ in an environment without obstacles is called Dubins maneuver, viz Section 2.1.3. This maneuver can be found very quickly because there exists an analytic solution. The length of the Dubins maneuver as a fundamental part of our further work, and therefore, we denote $\mathcal{L}(q_1, q_2)$ to represent the length of the shortest path for the Dubins vehicle connecting two configurations q_1 and q_2.

The vehicle is requested to visit all the given goals, and therefore, for each goal p_i, there must be a configuration q_i of the path that satisfy visitation of p_i. For an instance with n goals, there is a set of configurations $q = \{q_1, \ldots, q_n\}$. However, in the DTSP(N) the order of visits as a sequence $\Sigma = (\sigma_1, \ldots, \sigma_n)$ is not given. This causes that a part of the problem is also a combinatorial optimization similarly to the standard TSP.

To further simplify the notation, the length of the shortest tour through the set of configurations q in the sequence Σ is denoted as $\mathcal{L}(q, \Sigma)$, which is the sum of the lengths of Dubins maneuvers between adjoining configurations in the sequence. Formally, it is defined as:

$$\mathcal{L}(q, \Sigma) = \mathcal{L}(q_{\sigma_n}, q_{\sigma_1}) + \sum_{i=1}^{n-1} \mathcal{L}_\rho(q_{\sigma_i}, q_{\sigma_{i+1}}). \tag{3.1}$$

In the MPP, the objective is to minimize the total cost to visit the given goals. Particularly in the DTSP(N), the cost function is represented directly as $\mathcal{L}(q, \Sigma)$. Thus, we need to find both the configurations to visit the goals and also the sequence in order to evaluation the total cost of the solution.

3.1 Dubins TSP

The *Dubins traveling salesman problem* (DTSP) [27] is specified by a given set of point goals to be visited. The problem is to find the shortest path for Dubins vehicle visiting all the given points. In this problem, it is necessary to estimate the sequence of the goal points from the given set and an orientation of the Dubins vehicle at each point goal. The resulting path is then created by connecting every two consecutive points in the sequence using the Dubins maneuver. An example solution of the DTSP is depicted in Fig. 3.1. The path is continuous and fulfills the minimum turning radius constraint ρ. Although, there is not self-intersection of the path, it is possible the optimal path can have self-intersections for specific instances of the DTSP. This is different from the Euclidean TSP for which the optimal path is always without any self-intersection.

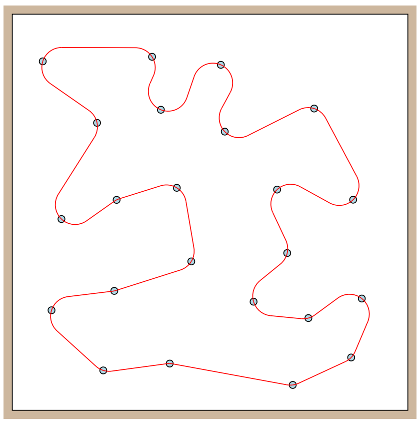


Figure 3.1: Example of the Dubins TSP

More formally, we formulate the DTSP as an optimization problem with the cost function $\mathcal{L}(q, \Sigma)$ defined in Equation 3.1. The set of target positions $p = \{p_1, \ldots, p_n\}$ is given, where $p_i \in \mathcal{W}$ and $\mathcal{W} \subset \mathbb{R}^2$. The problem is to find both the permutation $\Sigma = (\sigma_1, \ldots, \sigma_n)$ of targets points and vehicle headings $\Theta = \{\theta_1, \ldots, \theta_n\}$ at each point p_i. Having these preliminaries, the DTSP can be written down as follows.

Problem 3 (DTSP)

$$\begin{aligned} \mathit{minimize}_{\Sigma,\Theta} \quad & \mathcal{L}(q, \Sigma) \\ \mathit{subject\ to} \quad & q_i = (p_i, \theta_i),\ i = 1, \ldots, n, \end{aligned}$$

3.1.1 Targets distances constraint

The DTSP can be significantly simplified if the distances between points in p have similar length as the minimum turning radius or longer. To improve readability of the text, we introduce the minimum distance constraint D_K on the set of given points p as follows:

$$||p_i - p_j|| > K\rho \qquad \forall i, j = 1, \dots, n; \quad i \neq j. \tag{3.2}$$

3.2 Dubins TSP with Neighborhoods

The *Dubins traveling salesman problem with neighborhoods* (DTSPN) [8] is a straightforward generalization of the DTSP in which the set of point goals to be visited is generalized into a set of regions and thus the problem is to determine the shortest path visiting all the given goal regions by the Dubins vehicle. A region is considered as visited if there exist an intersection between the region and the path. The DTSPN is motivated by real scenarios in which an UAV is requested to provide snapshots of some objects or communicate with stationary stations at some communication radius. The DTSPN is complicated by the fact that the vehicle can visit any part of each region in any direction. It often enables to shorten the length of the solution in comparison to the original DTSP, where the point goals are prescribed, instead of the goal regions. An example solution of the DTSPN is depicted in Fig. 3.2.

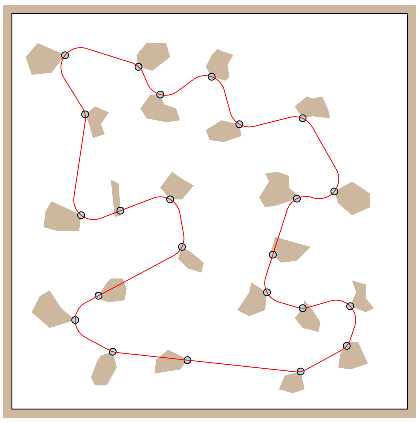


Figure 3.2: Example of Dubins TSP with Neighborhoods

Similarly to the DTSP, we formulate the DTSPN as an optimization problem with the cost function $\mathcal{L}(q, \Sigma)$ defined by Equation 3.1. Unlike in the DTSP, positions of the configuration q_i is not fixed and can be placed anywhere inside the particular goal region. The DTSPN is defined for a given set of regions $\mathcal{R} = \{\mathcal{R}_1, \dots, \mathcal{R}_n\}$ and a projection of the configurations $P(q_i)$ to its position p_i in the world coordinates $P(q_i) = p_i$, where $p_i \in \mathcal{W}$. Having these preliminaries, the DTSPN can be written down as follows.

Problem 4 (DTSPN)

$$\begin{aligned} \mathit{minimize}_{q,\Sigma} \quad & \mathcal{L}(q,\Sigma) \\ \mathit{subject\ to} \quad & \mathcal{P}(q_i) \in \mathcal{R}_i;\ i = 1,\ldots,n \\ & q = \{q_1,\ldots,q_n\} \end{aligned}$$

3.2.1 Targets distances constraint

Two main variants of the DTSPN can be considered depending on whether the regions are overlapping or not. In the case of non-overlapping regions, we can extend the minimum distance constraint D_K for the DTSPN. We define the minimum distance constraint D_K for the given set of regions $\mathcal{R}$ as a minimum distance between any pair of regions:

$$||p_i - p_j|| > K\rho \qquad \forall p_i \in R_i; \quad \forall p_j \in R_j; \quad i,j \in 1,\ldots,n; \quad i \neq j \tag{3.3}$$

CHAPTER 4

Analysis

In this chapter, we study the DTSPN and analyze the basic properties of the optimal solution. In the first section, we focus on the DTSP which can be considered as a sub-problem of the DTSPN. Therefore, all properties of the optimal path for the DTSP hold also for the optimal solution of the DTSP with Neighborhoods.

4.1 On the Dubins TSP

The DTSP is specified by the set of point goals p. For each goal point there exists a corresponding configuration of the Dubins vehicle on the path. The final path is constructed from the set of configurations $q = \{q_1, \ldots, q_n\}$ by applying the Dubins maneuver between two consecutive configurations in the sequence of configurations defined by the sequence Σ. The total length $\mathcal{L}(q, \Sigma)$ is computed according to Equation 3.1. Since the optimal path is constructed from Dubins maneuvers, the path is continuous and also its first derivatives are continuous. Hence, the optimal path is a closed curve from the C^1 class.

As Dubins maneuvers are used, the optimal path consists of straight line segments and curve segments with the minimum turning radius of the Dubins vehicle. No curves with a different curvature are possible to be in the optimal path.

Properties of the optimal solution of the DTSP are influenced strongly by the value of the minimum distance constraint D_K measured in multiples of the minimum turning radius. A special case is when the value $K \geq 4$. In this case, we can formulate the following lemma.

Lemma 1 *For an optimal solution of the DTSP with the D_4 constraint, all Dubins maneuvers connecting two consecutive configurations q_i and q_{i+1} are always of the CSC type.*

Proof 1 *Euclidean distance between q_i and q_{i+1} is always longer than 4ρ, and therefore, it is not possible to construct the CCC maneuver [7].*

Lemma 1 holds independently on the permutation Σ of the given point goals. As the lemma holds, all parts of the optimal path connecting two given points p_i and p_j, corresponding to configurations q_i and q_j, are made only by Dubins CSC maneuvers. If the D_4 constraint holds, the length of the Dubins maneuver $\mathcal{L}(p_i, p_j)$ is a continuous function [28].

For the DTSP with D_4, only CSC maneuvers are possible to be optimal. Therefore, we introduce new notation where C_i^+ states for the C part of Dubins maneuver leading to configuration q_i and C_i^- for the C part going from the configuration. An example of this notation is depicted in Fig. 4.1. Generally, these $C^\pm$ parts can have zero length and any orientations of the curvature. If the DTSPN satisfies the D_4 constraint, the $C^\pm$ parts connected with one point have the same length and orientation. This property is formulated in the following lemma.

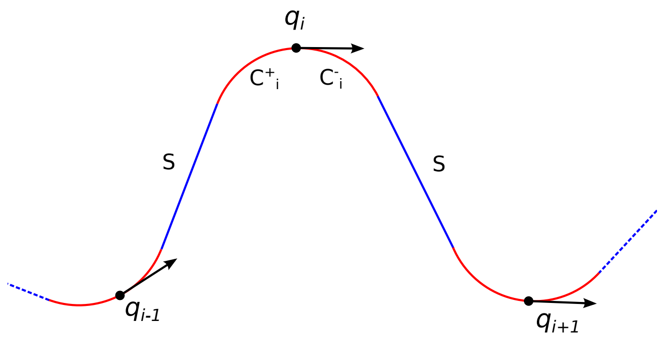


Figure 4.1: Example of a path for the DTSP

Lemma 2 *For an optimal solution of the DTSP with the D_4 constraint, for all configurations q_i it holds that both corresponding curve segments C_i^+ and C_i^- are equally long and they have the identical orientation.*

Proof 2 *The Equality of lengths and orientations of C_i^+ and C_i^- was proven in [29] and [28] for an arbitrary permutation Σ. Since, the lemma holds for any permutation, it also certainly holds for the permutation of the optimal solution of the DTSP.*

4.1.1 Dubins touring problem

In the case the sequence Σ is given, the DTSP can be transformed into a problem of finding the orientations Θ for the given set of points $p = \{p_1, \ldots, p_n\}$. We call the problem *Dubins touring problem*[1] (DTP). The transformation of the original sequence of points

[1]We call the problem *Dubins touring problem* due to it's similarity with the *Touring polygon problem* (TPP) [30]. For the both problems the permutation of the goals to be visited is given.

p in the DTSP into the ordered sequence of points p' of the DTP is straightforward. The point set is rearranged by the given permutation $\Sigma = (\sigma_1, \ldots, \sigma_n)$:

$$p'_i = p_{\sigma_i}, \quad i = 1, \ldots, n. \tag{4.1}$$

Now, the ordered set of point is given and the DTP can be formulated as an optimization problem. The objective function is the length of the path, defined by Equation 3.1. Variables are the headings Θ for all the given points.

Problem 5 (DTP)

$$\begin{aligned} \textit{minimize}_{\theta} \quad & \mathcal{L}(q, \Sigma) \\ \textit{subject to} \quad & q_i = (p_i, \theta_i),\ i = 1, \ldots, n, \end{aligned}$$

In the case the D_4 constraint holds, the objective function is continues due to properties of the Dubins maneuver. The DTP was studied in [28], where the authors showed that the DTP with the D_4 constraint can be reduced to n-dimensional convex optimization sub-problems, each over a convex polyhedral domain defined by at most $4n$ inequalities. The maximum number of the sub-problems is 2^{n-2}. Even though, the number of sub-problems grows exponentially, in a usual case the real number of sub-problems is considerably smaller. The authors used the term *sharp turn* and derived relationship to the number of sub-problems which are needed to be considered.

4.1.2 Existing approaches for the DTSP

Many approaches for solving the DTSP were proposed in the past. Most of them are based on a decoupled approach. The sequence of waypoints is determined first by using the Euclidean TSP. Subsequently, the DTSP is transformed into the DTP.

One of first attempts to address the DTP was in [27] where authors proposed *Alternating Algorithm* (AA). The AA algorithm applies straight line segment to even segments in the sequence which defines particular orientation of the configurations of the segment endpoints. For odd segments, the Dubins maneuver is applied. An example of such a solution is depicted in Fig. 4.2 where straight lines segments are denoted by the black color and the Dubins maneuvers are in red and blue.

The DTP was further studied in [29], where a receding horizon planning approach was suggested. Instead of using two consecutive points, the authors optimized the path regarding three following waypoints. This approach is also called *k-step look-ahead algorithm*, where k stands for the number of the goal points considered in the local optimization.

In [31], authors propose an approach based on a discretization of possible values of headings at the points to be visited. Having the heading samples, particular configurations are created and they are connected by Dubins maneuver to form a roadmap of Dubins maneuvers connecting particular configurations. This roadmap is then used to formulate the problem as the *generalized asymmetric traveling salesman problem* (GATSP). The GATSP can be then transformed to the ATSP by the Noon-Bean transformation [24] and solved by any existing solver, e.g., the Lin-Kernighan algorithm.

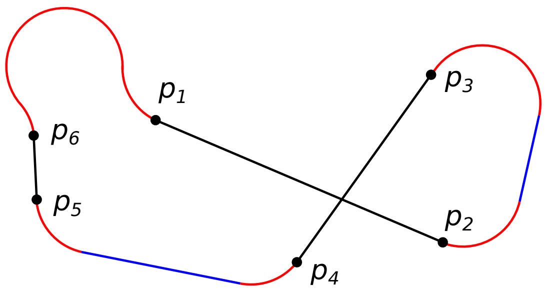
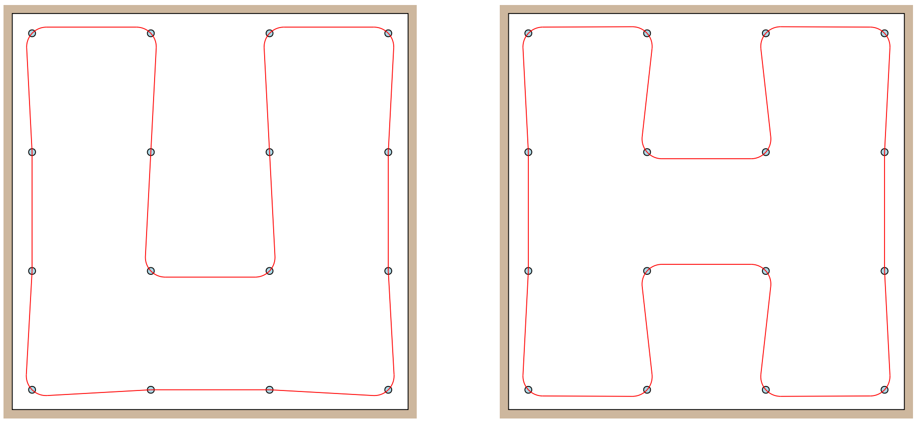


Figure 4.2: Alternating algorithm

4.1.3 On the sequence in the DTSP

The DTSP includes the combinatorial problem of finding the optimal sequence Σ_{opt} to visit the given goals. If the Σ_{opt} is known the problem is transformed to the DTP where the problem is to determine particular orientations Θ. The advantage of the transformed problem is that the ATSP can be efficiently addressed by existing available algorithms, which significantly simplifies the solution of the original problem. It would be helpful to have a method which can independently find the optimal sequence. Unfortunately, it is not possible due to the complexity of the DTSP and mutual dependencies of the optimal sequence and optimal headings at the point for the optimal solution of the original problem.



(a) Optimal solution (Dubins distance 65.38 m) (b) Non-optimal solution (Dubins distance 66.14 m)

Figure 4.3: Two different solution of the same instance of the DTSP

Finding an optimal sequence Σ_{opt} is a challenging combinatorial problem. Hence, heuristic methods are often used. One of the most straightforward method is estimation of the sequence by simplifying the DTSP to the TSP using Euclidean distance. The *Euclidean Traveling Salesman Problem* (ETSP) is the well studied problem [6] and many solvers are available. Despite the ETSP is a suitable heuristic for estimating the sequence, it cannot guarantee the optimal solution of the DTSP. An example is shown in Fig. 4.3. There two solutions for the same instance of the DTSP visualized in the figure. The set of given points is generated on a grid with the step 4ρ where $\rho = 1$ m. There are 16 points and the optimal solution for the corresponding ETSP is 64 m long. There exist two different optimal solution of the ETSP forming 'U' and 'H' patterns. However, corresponding solutions for the DTSP have different lengths (Dubins distances). It is caused by a different number of the sharp turns. This simple example clearly shows that a sequence of visits determined as an optimal solution of the ETSP does not guarantee an optimal solution of the DTSP would be found.

4.2 On the Dubins TSP with neighbourhoods

In this section, we further extend the discussed properties of the optimal solution of the DTSP with the D_4 constraint in the context of the optimal solution of the DTSPN with the D_4 constraint.

First, we extend Lemma 1 for the DTSPN, while the idea remains the same, i.e., only CSC maneuvers can be used in the optimal solution of the DTSPN with the D_4 constraint.

Lemma 3 *For an optimal solution of the DTSPN with the D_4 constraint, all Dubins maneuvers connecting two consecutive configurations q_i and q_{i+1} are always of the CSC type.*

Proof 3 *Euclidean distance between q_i and q_{i+1} is always longer than 4ρ, and therefore, it is not possible to construct the CCC maneuver [7].*

A similar extension can be applied also to Lemma 2. Here, we need to modify the proof and assume the truthfulness of the original lemma. An example illustrating the lemma is depicted in Fig. 4.4a.

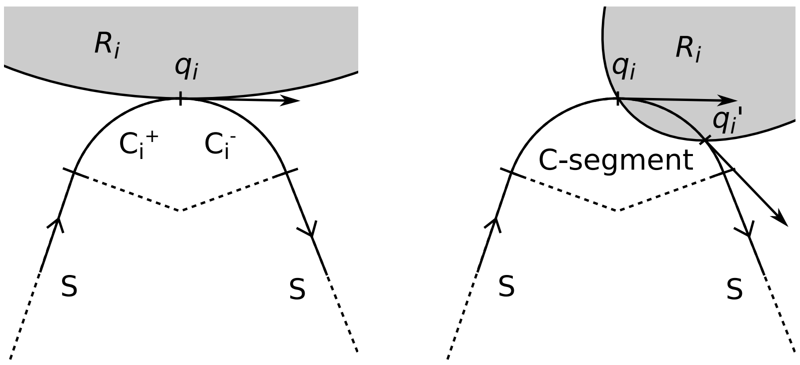
Lemma 4 *For an optimal solution of the DTSPN with the D_4 constraint, for all configurations q_i it holds that both corresponding curve segments C_i^+ and C_i^- are equally long and have the identical orientation.*

Proof 4 *Let assume we have the optimal solution (q, Σ). We can fix the position p_i and let the orientation θ_i be free for each configuration q_i in the optimal solution of the DTSPN with the D_4 constraint. Then, the problem becomes the DTSP with the optimal solution (q, Σ). Since Lemma 2 holds for the DTSP, this property must also hold for the DTSPN.*

The previous lemmas were extended from the DTSP. Now we study properties of the vehicle configurations q_i which visits the goal region $\mathcal{R}_i$. As it is depicted in Fig. 4.4b, we assume the optimal path has only a single point intersecting the goal region. This is an appropriate consideration, but it is necessary to consider an intersection of the region with straight lines segments of the path. In such a situation, there exist more configurations intersecting the same goal region with the optimal path. This observation can be formulated in the following lemma.

Lemma 5 *In the optimal solution of the DTSPN with the D_4 constraint, for each region $\mathcal{R}_i$ there is only one configurations $q_i \in \mathcal{R}_i$ of the optimal path that is intersecting with the region, or there is not a turn part of the optimal maneuver corresponding to q_i, i.e., $C_i^+ = C_i^- = 0$.*

Proof 5 *This can be shown by a contradiction. Let an optimal solution of the DTSPN pass a region $\mathcal{R}_i$ by a turn part at the configuration q_i. Assume there are several intersections of the optimal path with $\mathcal{R}_i$. Since q_i is a part of the optimal path and Lemma 4 holds, the both curve segments C_i^+ and C_i^- are equally long. Now, consider a different configuration q_i', which is also a configuration of the optimal path intersecting the region $\mathcal{R}_i$. An example is depicted in Fig. 4.4b. The optimal path consists of CSC maneuvers (Lemma 3) and thus q_i' must be on the same C–segment as q_i. However, its corresponding C^+ and C^- parts are not equally long, which is in contradiction with Lemma 4, unless there is not a turn segment corresponding to q_i, i.e., $C^+ = C^- = 0$.*



(a) C_i^+ and C_i^- have the same length and orientation

(b) C-segment does not satisfy property of the optimal solution

Figure 4.4: Properties of the optimal solution of the DTSPN

Notice, Lemma 5 does not forbid more intersections of the optimal path with the particular region if there is no turn segment corresponding to the visiting configuration.

The results of the analysis can be used as a guideline, how to restrict possible solutions, which are surely not optimal. Therefore, we use the results in further chapters to design new approach to address the DTSPN.

CHAPTER 5

Existing approaches for the DTSPN

In literature, we can find several different approaches to address the DTSPN. The DTSPN is a challenging problem, where it is necessary to determine both the target visiting sequence and the visiting configurations q for each given region. An intuitive approach is to find the permutation separately and then transform the DTSPN into a different problem in which particular configurations q are determined. Another approaches are based on sampling possible configuration to solve the DTSPN and then the samples are utilized to transform the DTSPN into the ATSP. Genetic algorithms can also be used.

According to the key principle how a solution of the DTSPN is found, we can divide existing approaches into three classes. The first class are the decoupled approaches. They find the sequence of visits separately and then transform the DTSPN into a different problem in which configurations q are determined. The second class are sampling-based methods, where the goal regions are sampled and then the DTSPN is transformed into the ATSP. Finally, the third class contains methods based on genetic algorithms that can be considered as a general optimization technique.

5.1 Decoupled methods

Decoupled methods solve the DTSPN by separating the problem into sub-problems which are then solved separately. They were first used for the DTSP for which the ETSP is commonly used as a suitable heuristic for estimating the sequence.

In [32], authors proposed to simplify the DTSPN by a solution of the related the *Traveling Salesman Problem with Neighborhoods* (TSPN) where Euclidean metric is used. The solution of the TSPN contains both the permutation and the visited points of the goal regions. The points and the determined sequence of their visits are then used for transformation of the DTSPN into the DTP. The authors also showed that this approach provides a suitable heuristic for Dubins-like vehicle. An example solution of the TSPN is depicted in Fig. 5.1a.

The solution of the TSPN contains permutation and positions which are further used in Fig. 5.1b, where the transformed DTP is depicted together with an example solution. The TSPN is a well known problem which was addressed by several approaches [33, 34, 35, 36, 37] where different algorithms were proposed. Notice, there are also approaches to deal with the TSPN with obstacles in the polygonal domain, e.g., [38] where authors consider an algorithm based on self-organizing maps.

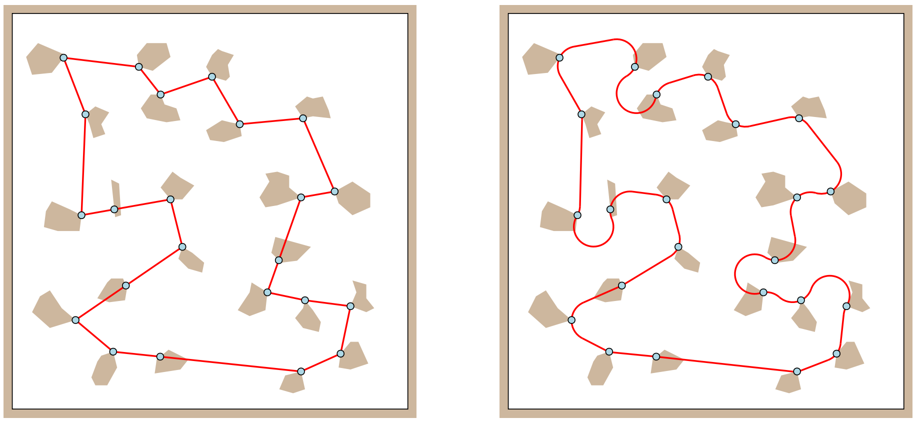

(a) An example solution of the TSPN (b) An example solution of the DTP

Figure 5.1: Decoupled approach for the DTSPN

Another decoupled method was presented in [39]. The authors proposed a three-stage process based on an evolutionary algorithm to solve the DTSPN. These stages are:

1. Position update: Move the position of the configuration q_i inside the region R_i in order to minimize the cost function.
2. Orientation update: Change the orientation θ_i of the configuration q_i in order to minimize the cost function.
3. Visiting sequence update: Check if there is a better sequence Σ for configurations q.

The algorithm starts from an arbitrary solution. This solution is locally optimized in the first two stages. The key idea is in the third stage, where the current configurations q are connected into a fully connected roadmap constructed from the Dubins maneuvers. This roadmap is subsequently considered as the ATSP which is solved by existing solvers. In the article, the authors used the Concorde solver [18] to solve the ATSP to optimum, but it is possible to use any other solver. The three stages are repeated until the solution is improving.

5.2 Sampling-based methods

Another class of the approaches for solving the DTSPN are based on sampling-based methods [40, 41]. They make samples of the target regions. The samples can be created in the whole regions or only on the boundaries of the regions. In the next step, the samples are connected together between different regions by the Dubins maneuver, the shortest possible path, to form a roadmap. Such a created roadmap is fully connected except configuration in the same region and can be considered as an instance of the asymmetric GTSP. Notice, the sampling based method is resolution complete [41] if the GTSP is solved to the optimum. If a solution exists, a *resolution complete* [10] algorithm will find it in finite time; however, if a solution does not exist, the algorithm may run forever.

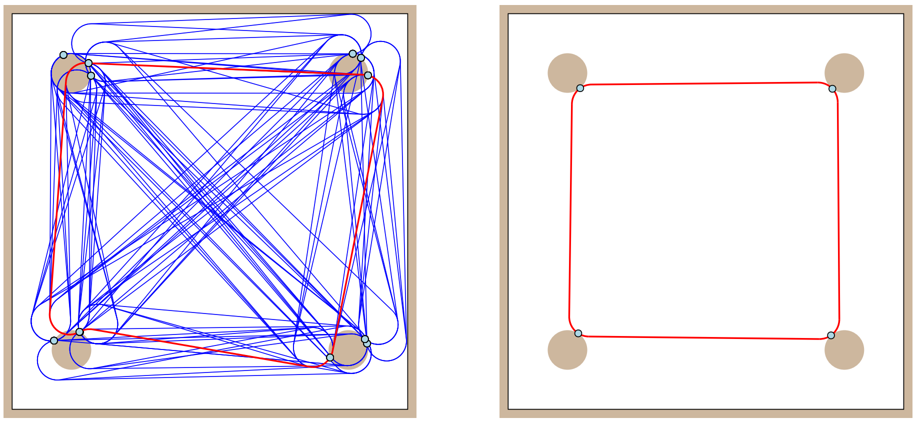

(a) 3 samples (b) 100 samples

Figure 5.2: An example of sampled road map for DTSPN

The samples can be created in whole regions or only on the boundaries of the regions, but using only samples on the boundaries have statistically better results for the same number of samples. An example of a created roadmap is depicted in Fig. 5.2a, where 3 random sampled configurations are used for each given region to be visited. The roadmap is blue and the best solution of the generated GTSP is in red. In Fig. 5.2b, a shorter solution for the same instance of the DTSPN is depicted. Here, 100 samples were used for each region, and therefore, a bit shorter solution is found.

The generated GATSP can be transformed to the ATSP by the Noon-Bean transformation [24]. An example of the Noon-Bean transformation is depicted in Fig. 5.3. Here, vertices $p_{i,j}$ stand for the sampled configurations of the Dubins vehicle sampled in the goal region $\mathcal{R}_i$ which is denoted as the set S_i in the GATSP. An oriented edge in the graph stands for the Dubins maneuver and the associated cost of the edge is equal to the length of the maneuver. The transformation creates a zero cost circuit of vertices $p_{i,j}$. An oriented edge is transformed such that it starts at the same vertex but it leads to the vertex following in the zero cost circuit. The associated cost of the edge is transformed into the sum of the original cost and a big constant M. This big constant forces the optimal solution of the generated ATSP to visit each set of vertices S_i only once and use the zero cost circuit. It ensures that all used edges create a tour and each region is visited only once.

A backward transformation is straightforward. The edges still represent original Dubins maneuvers which directly create a tour in the GATSP. The transformed ATSP

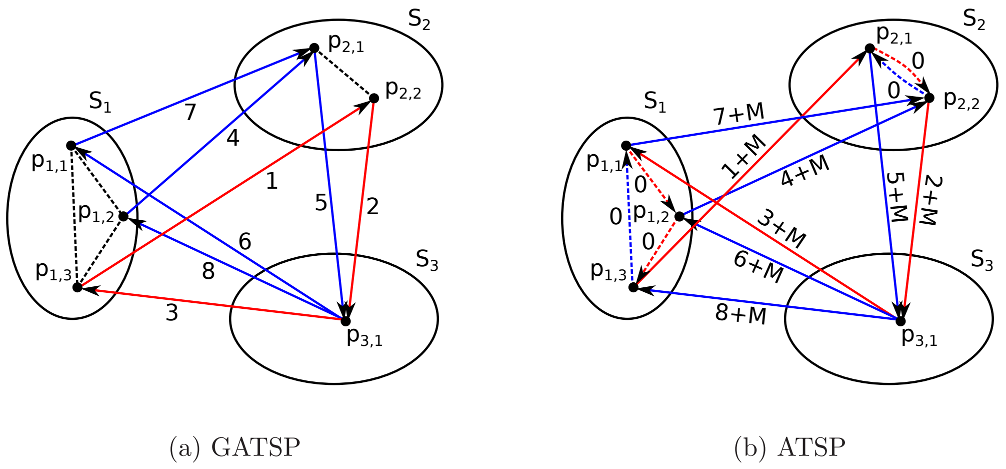


Figure 5.3: Example of the Noon-Bean transformation

can be solved by many existing algorithms. The authors of [40, 41] used Lin-Kernighan heuristic algorithm.

5.3 Genetic algorithms

The last class of the existing approaches, are methods based on genetic or more specifically evolutionary techniques [8, 42]. The general idea is to encode the solution by a chromosome in which permutation $\Sigma = (\sigma_1, \ldots, \sigma_n)$ and configurations $q = \{q_1, \ldots, q_n\}$ are stored:

$$\text{Chromosome:} \quad \begin{pmatrix} \sigma_1 \\ q_{\sigma_1} \end{pmatrix} \cdot \begin{pmatrix} \sigma_2 \\ q_{\sigma_2} \end{pmatrix} \cdot \begin{pmatrix} \sigma_3 \\ q_{\sigma_3} \end{pmatrix} \cdot \begin{pmatrix} \sigma_4 \\ q_{\sigma_4} \end{pmatrix} \cdots \begin{pmatrix} \sigma_n \\ q_{\sigma_n} \end{pmatrix}$$

The population of individuals is evolved by applying the mutation and crossover operators. Different types of selection method can be used, e.g., *roulette wheel selection* or *tournament selection*. Three kinds of mutation operators were used by the authors in [8]. First mutation operator is *orientation shift*, which randomly changes the orientation of the configuration q_i in the interval $\langle 0, 2\pi)$. The second operator is *position shift*, which randomly changes the positions of the configuration q_i inside the corresponding target region $\mathcal{R}_i$, i.e., $q_i \in \mathcal{R}_i$. The last operator is the *partial reverse* operator, which reverses the sequence Σ between two uniform randomly chosen indexes. The configurations in the reverse part of the sequence are rotated to the reversed orientation, i.e., rotated by π.

The authors adapted the *Ordered Crossover* operator (OX) [43] for the DTSPN in which random parts from two parents are selected. These parts are directly used in the descendants. Remaining parts are completed from the second parent. For example, let two parent chromosomes be:

- Parent 1: $\binom{2}{q_2^1} \cdot \binom{4}{q_4^1} \cdot \binom{6}{q_6^1} \cdot \binom{5}{q_5^1} \cdot \binom{3}{q_3^1} \cdot \binom{1}{q_1^1}$
- Parent 2: $\binom{5}{q_5^2} \cdot \binom{4}{q_4^2} \cdot \binom{1}{q_1^2} \cdot \binom{3}{q_3^2} \cdot \binom{2}{q_2^2} \cdot \binom{6}{q_6^2}$.

Then, their derived children are chosen. Randomly choose parts go directly from Parent 1 to Child 1 and from Parent 2 to Child 2, respectively:

- Child 1: $\binom{?}{?} \cdot \binom{?}{?} \cdot \binom{6}{q_6^1} \cdot \binom{5}{q_5^1} \cdot \binom{3}{q_3^1} \cdot \binom{?}{?}$
- Child 2: $\binom{?}{?} \cdot \binom{?}{?} \cdot \binom{1}{q_1^2} \cdot \binom{3}{q_3^2} \cdot \binom{2}{q_2^2} \cdot \binom{?}{?}$.

Finally, the descendant chromosomes are completed by remaining part from the second parent. The children are completed in the following way:

- Child 1: $\binom{4}{q_4^2} \cdot \binom{1}{q_1^2} \cdot \binom{6}{q_6^1} \cdot \binom{5}{q_5^1} \cdot \binom{3}{q_3^1} \cdot \binom{2}{q_2^2}$
- Child 2: $\binom{4}{q_4^1} \cdot \binom{6}{q_6^1} \cdot \binom{1}{q_1^2} \cdot \binom{3}{q_3^2} \cdot \binom{2}{q_2^2} \cdot \binom{5}{q_5^1}$.

Recently, the genetic approach was modified into a memetic algorithm in [42] to address the DTSPN. The authors used similar operators for their memetic algorithm and add local improvement of the chromosome by optimizing the position of configurations. They also used only configurations on the boundaries of the regions; so, they can code configurations only by one real variable. They estimated orientations by the AA, which is a heuristic algorithm.

CHAPTER 6

Proposed approach for the DTSPN

In this chapter, we propose new decoupled approach based on local optimization methods. The approach has been primarily designed to solve instances of the DTSPN for which D_4 constraint holds and all regions are convex. Although the D_4 constraint was originally assumed, the proposed method can be used even for more general instances of the DTSPN, which is further support by the performed evaluation and comparison with existing solutions in Chapter 7.

6.1 Proposed method

The proposed decoupled method to address the DTSPN consists of two main parts. The first part estimates the sequence of the given regions while the second part is dedicated to find the path visiting all the given goal regions in the order defined by the estimated sequence. The proposed method works as follows.

Similarly to the previous approaches, we use the ETSP solver to find the sequence based on a point distance between centers of the regions. The center of region is a point inside the region which is, in some sense, the most interesting part of the region. The DTSPN is often inspired by real problems in which the vehicle can sense the object from a neighborhood. In such a case, we define the center as a real position of the object of interest. Alternatively, the center can be determined as a center of gravity of the regions.

After the sequence is determined, the DTSPN is reduced to the *Dubins touring regions problem* (DTRP)[1] which is a sub-problem of the DTSPN. The DTRP is an extended version of the DTP where instead of point goals, the set of regions are given. Hence, the DTP is a sub-problem of the DTRP. We propose a new method to address the DTRP called *Local Iterative Optimization* (LIO) which is based on local optimization procedure to find the most suitable position and orientation of each configuration to visit all the regions.

[1]We call the problem *Dubins touring regions problem* due to it's similarity with the *Touring polygon problem* (TPP) [30], similarly as the DTP.

6.1.1 Reduction of the DTSPN to the DTRP

In the first part of the proposed method, the sequence Σ of visits to the regions $\mathcal{R}$ is determined. Using the sequence, we reduce the DTSPN to the DTRP. The problem to determine the optimal solution of the DTRP is to find configurations $q = \{q_1, \dots, q_n\}$ that minimize the total tour length $\mathcal{L}(q, \Sigma)$ for the given sequence Σ. Each configuration q_i, from the sequence of configurations defined by Σ, has to lie in the corresponding region, i.e., $q_i \in \mathcal{R}_i$. For every solution of the DTRP, the path intersects each region at least in one configuration. Hence, we can consider only configurations on the boundary $\delta\mathcal{R}_i$ of the corresponding region $\mathcal{R}$:

$$q_i \in \delta\mathcal{R}_i, \quad i = 1, \dots, n. \tag{6.1}$$

We introduce new set of variables $\alpha = \{\alpha_1, \dots, \alpha_n\}$ which determines positions on the edge for each region, where $\alpha_i \in \langle 0, 1)$. The α is defined by a projection $\mathcal{B}$ which projects the position variable α on the boundary of the region $\mathcal{R}_i \in \mathcal{R}$ by the equation:

$$p_i = \mathcal{B}(R_i, \alpha_i), \quad i = 1, \dots, n. \tag{6.2}$$

A straightforward way to define $\mathcal{B}$ is to define an initial point on the boundary and measure a relative length of the boundary from the initial point. An example of such a projection $\mathcal{B}$ is shown in Fig. 6.1.

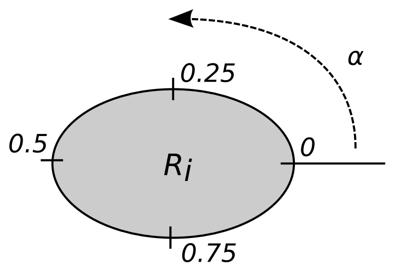


Figure 6.1: Position of the point of visit to the region $\mathcal{R}_i$ defined by the projection $\mathcal{B}$

We can define the DTRP as a minimization optimization problem where the target regions $\mathcal{R}$ and its sequence of visits are given. The problem is to determine values of the positions α with the associated orientations Θ under the objective function. The objective function is the total length of the solution which is constructed from Dubins maneuvers. Formally, the problem can be stated as follows.

Problem 6 (DTRP)

$$\begin{aligned} \textit{minimize}_{\alpha,\Theta} \quad & \mathcal{L}(q, \Sigma) \\ \textit{subject to} \quad & q_i = (p_i, \theta_i),\ p_i = \mathcal{B}(R_i, \alpha_i),\ i = 1, \dots, n \end{aligned}$$

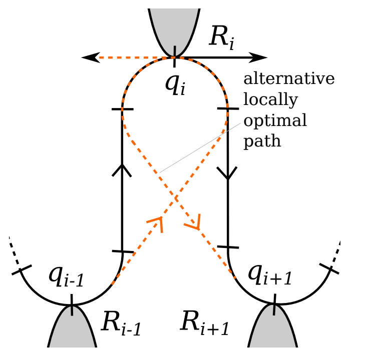
6.1.2 Local Iterative Optimization for the DTRP

In the second part of proposed method, the DTRP is solved by the newly proposed *Local Iterative Optimization* (LIO) algorithm. The LIO algorithm starts with some initial configurations which can be generated randomly or by a heuristic algorithm, like the AA. Then, the algorithm optimizes the total tour length by adjusting values of directions Θ and positions α of candidate configurations for all regions. The process is repeated until the path is improving or a termination condition is not meet, e.g., after a given number of iterations. The algorithm is expressed in Algorithm 1.

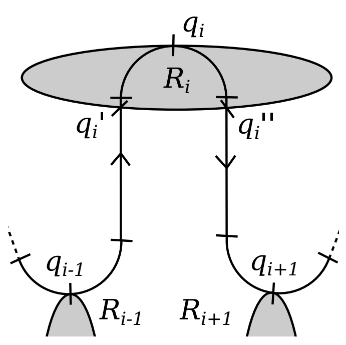
Algorithm 1: Local Iterative Optimization (LIO) for the DTRP

```
   Data: Input regions $\mathcal{R}$, candidate sequence $\Sigma$
   Result: Configurations $q = (q_1, \ldots, q_n)$, $q_i \in \delta\mathcal{R}_i$
1  initialization()                    // random assignment of $q_i \in \delta\mathcal{R}_i$;
2  while global solution is improving do
3      for every $\mathcal{R}_i \in \mathcal{R}$ do
4          $(\theta_i, q_i)$ := optimizeOrientationLocally($\theta_i$);
5          $(\alpha_i, q_i)$ := optimizePositionLocally($\alpha_i$);
6      end
7  end
```

Since the proposed iterative procedure performs only local optimization, it can stuck in a local minima. However, for the DTRP with the D_4 constraint, we can identify two types of possible local minima. The first type occurs during the optimization of the orientation of a single configuration. This local minima, depicted in Fig. 6.2a, has been studied in [28]. The second type occurs during optimization of the configuration position. In the example depicted in Fig. 6.2b, the locally shortest paths intersects the boundary of the target region R_i in two other configurations q_i' and q_i''. From Lemma 5, we know that this path is not globally optimal. Thus, such a situation can be detected to avoid the local optima.



(a) Two locally optimal solutions



(b) Local minimum of position α_i

Figure 6.2: Local extremes evaluated during the proposed local optimization

The LIO algorithm tries to avoid local shortest paths by examination of alternative orientations and positions. The first subroutine called *optimizeOrientationLocally()* optimizes the orientation of one chosen configuration while all other configurations are fixed. Subsequent configurations are connected by Dubins maneuver to form the shortest possible path. The task is to minimize the length of two adjacent Dubins maneuvers. For each configuration q_i, the subroutine locally optimizes the cost function $cost(q_i)$ defined as:

$$cost(q_i) = \mathcal{L}(q_{i-1}, q_i) + \mathcal{L}(q_i, q_{i+1}). \tag{6.3}$$

The subroutine locally optimizes the orientation of the current configuration using variable step which is expressed in Algorithm 2. At the beginning, the step is set to π. By using this value, the subroutine examines a "reverse" configuration to avoid a local minimum. Although it works in many situations, it is not sufficient to guarantee the global optimal solution [28]. In subsequent, the variable step is adjusted depending on whether the path was shortened in the current iteration. If the shortening was successful, the step is increased. Otherwise, the step is reduced and its sign is changed. The algorithm terminates when the absolute value is reduced and it is shorter than the required tolerance. In the case the DTRP satisfies the D_4 constraint, the path converges to local minimum where the configuration cuts the corresponding turn into half. This meets Lemma 4, therefore such a path is a good candidate to be the optimal solution of the DTRP.

Algorithm 2: *optimizeOrientationLocally()* – Optimize orientation locally (subroutine of LIO)

```
Data: Actual region $R_i$ and corresponding heading $\theta_i$
Result: Improved heading $\theta_i$ and corresponding configuration $q_i$
1  step := $\pi$;
2  while $abs(step) > TOLERANCE$ do
3      $\theta'_i$ := $\theta_i$ + step;
4      $q'_i$ := onEdge($R_i$, $\alpha_i$, $\theta'_i$);
5      if $cost(q'_i) < cost(q_i)$ then
6          $\theta_i$ := $\theta'_i$;
7          $q_i$ := $q'_i$;
8          step := 2 · step;
9      else
10         step := -0.5 · step;
11     end
12 end
13 return ($\theta_i$, $q_i$)
```

The second subroutine called *optimizePositionLocally()* is based on the similar idea and is expressed in Algorithm 3. It locally optimizes the position of the given configuration q_i. It uses the same cost function defined by Equation 6.3. The main difference from the previous subroutine is that the position of the configuration q_i is optimized and is encoded by the variable α_i. The function $\mathcal{B}$ transforms it into the position at the border of the corresponding region. The variable step starts with the value 0.5, which represents the position on the opposite side of the region. If the position variable α_i leaves the range $\langle 0, 1)$, it is normalized back into to the interval $\langle 0, 1)$.

Algorithm 3: *optimizePositionLocally()* – Optimize position locally (subroutine of LIO)

```
Data: Actual region $R_i$ and corresponding position $\alpha_i$
Result: Improved position $\alpha_i$ and corresponding configuration $q_i$
step := 0.5;
while abs(step) > TOLERANCE do
    $\alpha'_i$ := $\alpha_i$ + step;
    $q'_i$ := onEdge($R_i$, $\alpha'_i$, $\theta_i$);
    if $cost(q'_i) < cost(q_i)$ then
        $\alpha_i$ := $\alpha'_i$;
        $q_i$ := $q'_i$;
        step := 2 · step;
    else
        step := -0.5 · step;
    end
end
return ($\alpha_i$, $q_i$)
```

For the DTRP with the D_4 constraint, the LIO algorithm ends with the path that satisfies Lemma 4. But satisfaction of Lemma 5 is not guaranteed. The returned path can intersect the boundary of the regions in more configurations while there is a turn. Such a path is proven to be non-optimal. These intersections can be further examined to improve the quality of the solution. Notice, the LIO algorithm can still stuck at a local minima even after applying all these additional tests. However, based on the performed evaluation and comparison with existing approaches, LOI provides competitive solutions with significantly less computations requirements then evolutionary approaches, see the results presented in Section 7.3.

6.1.3 Determination of the sequence to visit the regions

In most of decoupled approaches, the sequence to visit the regions is determined at first. For this purpose the ETSP is used. It is clear that using ETSP provides a feasible solution. However, it may not be necessarily the optimal sequence for the DTSPN. An example of the optimal solution of the ETSP which definitely does not provide the optimal sequence to visit the regions in the DTSP is shown in Fig. 4.3. Since the DTSP is a sub-problem of the DTSPN, this problem also occurs for the DTSPN. Therefore, we propose the following framework to address this issue.

The framework is based on considering several sequences that can be for example found as a solution of the k-best TSP [44]. Alternatively, the candidate sequences can be generated by solving k-best ETSPN, which can utilize information about the shapes of the goal regions. The main idea of the proposed framework is depicted in Fig. 6.3.

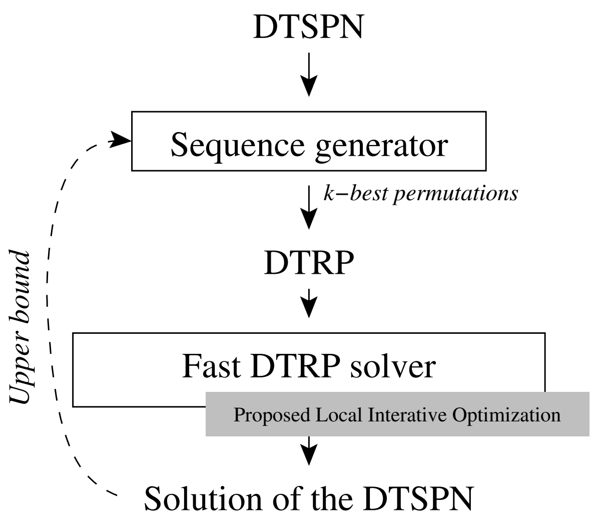


Figure 6.3: Proposed framework to solve the DTSPN

CHAPTER 7

Results

We have implemented several algorithms for the DTSPN without obstacles. To compare them, we created a random problem generator which can generate random instances of the DTSPN with various shapes of regions. All algorithms have been implemented in C++ and tested on a single core of the Intel Core i5-M480 CPU running at 2.67 GHz. The processor was accompanied with 4 GB RAM.

7.1 Problem generator

To evaluate the performance of the studied approaches for the DTSPN, we decided to create a random problem generator and provide an exhaustive comparison on several random instances of hte DTSPN with particular parameters of the problem. The developed generator places regions in the workplace. The input is the number of generated regions n and the minimum distance d between the regions. Since the minimum distance is given, the generated DTSPN instances fulfills the D_d constraint. Centers of regions are generated inside a bounding box with the side $6\sqrt{n}\rho$ which produces instances with a similar density of regions regardless the number of goal regions. A newly generated region is added only if it meets the D_d constraint with all already generated regions. This process continues until all regions is generated. The instances with up to n=500 regions were used. Besides, we considered several types of regions:

- single point;
- circle with the radius ρ;
- ellipse with the semi-axis 2ρ and 0.5ρ;
- polygon with up to 6 vertices created from a circle with the radius ρ.

Examples of randomly generated instances with 50 regions accompanied with a sample solution are depicted in the Fig. 7.1. The main difference between these instances is in the minimum distance constraint. A number regions are the same in the both instances, which is caused by using the same random seed for the utilized random generator. It can also illustrate the higher computational demands for creating instances with higher minimum distance d. The same method can be used for generating instances of the DTSP simply by generation only single points instead of more complex regions.

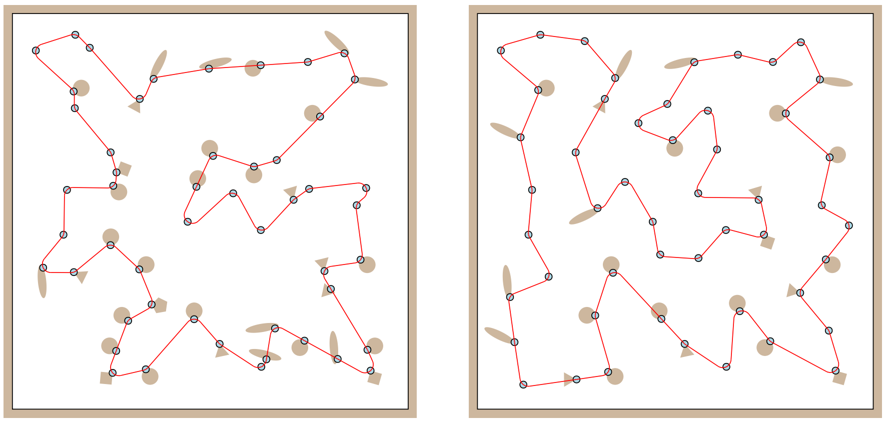

(a) D_1 convex regions (b) D_4 convex regions

Figure 7.1: Examples of randomly generated DTSPN instances ($n = 50$)

7.2 Implementations details of the used algorithms

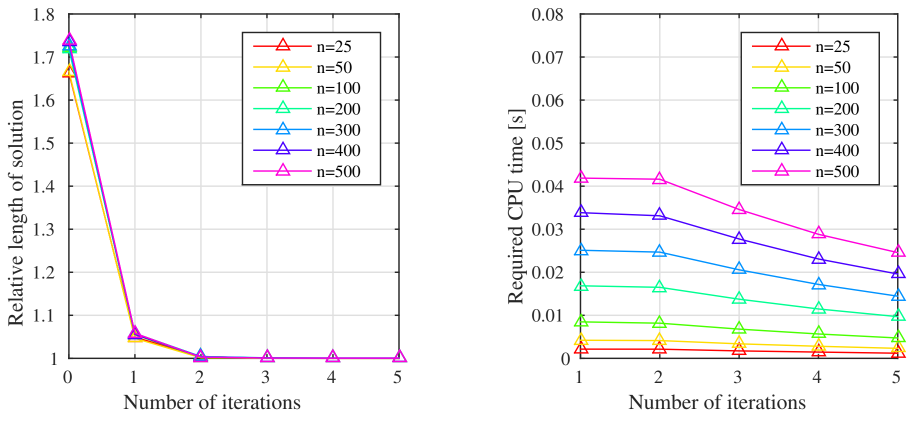
All algorithms have been implemented in the C++ language and two external libraries have been utilized. The first is the Concorde solver [18] which was used for solving the ETSP problems to optimal. The solver is based on linear programming solver and it is designed especially for solving the symmetric traveling salesman problem. The second library was LKH solver [22] which is an implementation of the Lin-Kernighan approximate algorithm. More details about the algorithm can be found in [20]. The LKH solver is faster than Concorde, but it does not guarantees the optimal solution. We use the LKH in our experiments to solve large ATSP instances created by the sampling-based approach for the DTSPN. Both libraries are free for an academic research.

7.2.1 Proposed decoupled algorithm

The first algorithm to compare is our new Local Iterative Optimization (LIO) algorithm. It is used together with the Concorde solver to provide a sequence of the given regions by solving the ETSP. Hence, in presented plots with the results, the combined algorithm is denoted as the ETSP+LIO. The LIO algorithm was implemented according to Algorithm 1. This algorithm call iteratively two subroutines defined in Algorithm 2 and Algorithm 3 until the solution converges to a local optima. The minimum step of the gradient descent used in the both subroutines is 10^{-5}.

The speed of the convergence of the LIO algorithm has been studied for an instances of the DTSPN with the D_4 constrain and a given sequence. The instance was randomly generated for various numbers of regions defined by n. The results are depicted in

Fig. 7.2. It shows a quick convergence of the optimization in the first few iterations for the main while-loop of Algorithm 1. The computational time of a single iteration decreases with the number of iterations.



(a) Average tour length (b) Average computational time

Figure 7.2: Performance of ETSP+LIO in particular iteration of the main loop

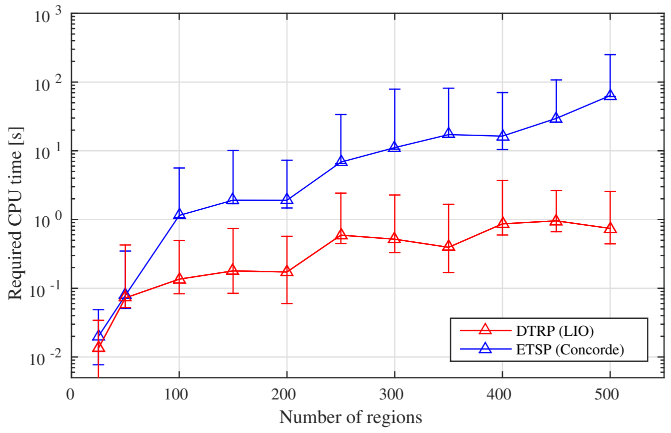


Figure 7.3: Average required computational time (from 10 trials) for the DTSPN algorithms with increasing number of regions n

Secondly, we have measured the required computational time separately for the ETSP and LIO parts of the decoupled algorithm. We have used instances of the DTSPN with D_4 and up to 500 regions. The results are depicted in Fig. 7.3. The presented results are the average values over 10 trials and the minimum and the maximum required computational time are shown as the endpoints of the error bars. The plots show the LIO part is executed much faster than find an optimal solution of the ETSP by the Concorde solver. The proposed algorithm (ETSP+LIO) can be accelerated by using heuristic algorithm for solving ETSP, e.g., the Lin-Kernighan algorithm. But, for the further evaluation of the LIO algorithm, we decided to use the Concorde solver and determine the optima solution of the underlying ETSP.

7.2.2 Other decoupled algorithms

We have implemented two other decoupled algorithms based on a solution of the ETSPN prior determination of Dubins maneuvers. We use a heuristic method for solving the ETSPN. A solution of the ETSPN is found in two phases. First, a sequence of the visits is determined from the solution of the ETSP by the Concorde solver, similarly to the ETSP+LIO approach. Then, the particular points of visits on the boundary of the goal regions are determined by local optimization method. Since the sequence of visits to the regions and particular entry point are determined, the problem is transformed to the DTP. We have implemented two reference methods for the DTP. The first method is the Alternating Algorithm (AA) [27] and the corresponding decoupled algorithm is denoted as the ETSPN+AA. The second implemented method for solving the DTP is a simplified version of the proposed LIO algorithm, where only headings are optimized but positions are taken from a solution of the ETSPN. We denote this algorithm as the ETSPN+HoLIO.

7.2.3 Genetic based algorithms

The next studied approach for the DTSPN found in literature are genetic based algorithms [8] and [42]. We have implemented two variants of the algorithm: genetic algorithm [8] and memetic algorithm [42]. Both of the algorithms used the same framework expressed in Algorithm 4. We use a tournament selection and the *Partial Reverse* operator is utilized as the sequence mutation operator. It randomly choose the part of the sequence to be reverted. Finally, the adapted version of the *Ordered Crossover* is used.

Table 7.1: Settings of the genetic based algorithms

Property	**Genetic**	**Memetic**
Population size	60	25
Tournament size	8	5
Probability K_1	0.2	0.1
Probability K_2	0.3	0.05

The difference between these two implemented genetic based algorithms is in the procedure `updateConfigurations()`. In the genetic algorithm, the procedure uses position and orientation mutation of the configurations. In contrast to that, a local optimization of the configurations is used in the memetic algorithm. The proposed LIO algorithm is utilized to perform the local optimization. The used settings of the both algorithms are expressed in Table 7.1. The coefficient K_1 stands for a probability of the sequence mutation and K_2 stands for a probability of changing configurations.

7.2.4 Sampling based algorithms

The last implemented algorithm is the sampling based transformation of the DTSPN to the aTSP, which is denoted in plots as Sample+ATSP. The given regions are sampled

Algorithm 4: Genetic algorithm for the DTSPN

```
Data: Target regions $\mathcal{R}$
i := 0;
P(0) := initializePopulation();                 // The initial population
while not termination do
    for each in population do
        parents := selection(P(i));
        if random() < $K_1$ then
            newSolution := mutateSequence(parents);
        else
            newSolution := crossover(parents);
        end
        if random() < $K_2$ then
            newSolution := updateConfigurations(newSolution);
        end
        P(i + 1) ← newSolution;
    end
    i := i + 1;
end
return the best solution
```

on the boundary by m random sample configurations. Hence there are $n \cdot m$ samples at total. The samples are then connected by Dubins maneuvers into the so-called Dubins roadmap. The roadmap can be considered as an instance of the GTSP which is subsequently transformed into the ATSP by the Noon-Bean transformation [24]. The ATSP is solved by the LKH solver [22] which is an implementation of the Lin-Kernighan heuristic algorithm. To study basic properties of the sampling algorithm, we have used an instance of the DTSPN with only 4 circle regions, which is depicted in Fig. 7.4. The sequence is obvious in this instance; hence, we utilized the proposed LIO algorithm as a reference solution of this problem. We have measured a quality of the solution and the required computational time. The results are depicted in Fig. 7.5 and Fig. 7.6, respectively.

Building the roadmap takes $\mathcal{O}((n \cdot m)^2)$ where n is number of regions and m is number or samples in each region. It is because a computation of the Dubins maneuver takes a constant time and there are $(n \cdot m)^2$ edges in the complete graph. The Noon-Been transformation does not change then number of vertices. For solving the generated ATSP, we use the LKH solver. According the author [22], the time complexity of the LKH is approximately $\mathcal{O}(n_{ATSP}^{2.2})$. Hence, the expected total time complexity is $\mathcal{O}((n \cdot m)^{2.2})$.

To compare the performance of the sampling based algorithm with other algorithm, we need to modify this algorithm. We want to have an any-time algorithm. The sampling-based algorithm works with a constant number of samples which are given a priory. Therefore, we run the sampling based algorithm repeatedly with an increasing

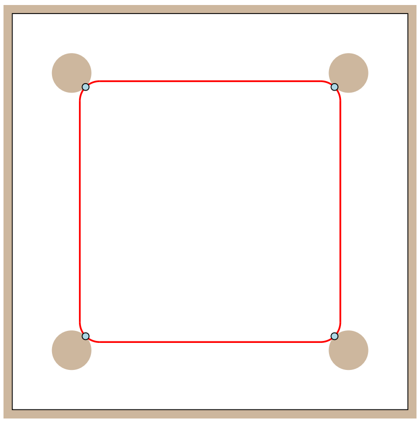


Figure 7.4: An instance of the DTSPN with 4 regions

number of samples. Since the time complexity of the algorithm is nearly quadratic, we want to use inverse function for the number of samples m_k according to the number k of the actual iteration:

$$m_k \approx \sqrt{2^k}. \tag{7.1}$$

After the rounding the m_k to integer value, it gives us the following series of numbers:

$$M = \{1, 2, 3, 4, 6, 8, 11, 16, 23, 32, 45, 64, 91, 128, 181, 256, \ldots\}.$$

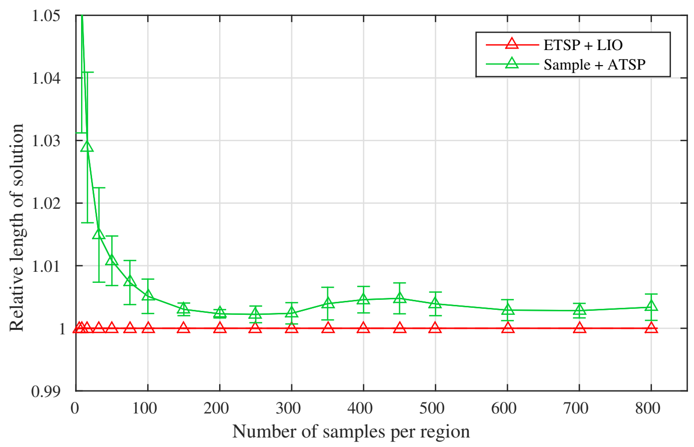


Figure 7.5: Average ratio of the tour length (from 20 trials) according to the ETSP+LIO solution for the instance of the DTSPN with 4 circle regions and D_4 constraint.

7.3 Results for the DTSPN

The performance of the proposed algorithm (ETSP+LIO) have been evaluated in a series of scenarios. The quality of the solutions provided by the proposed algorithm is

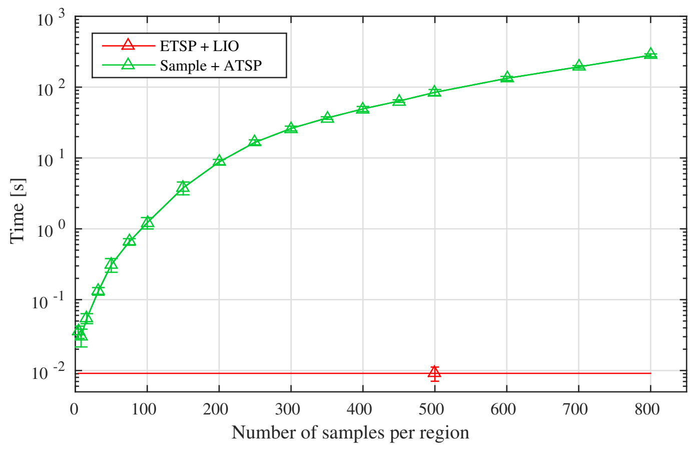


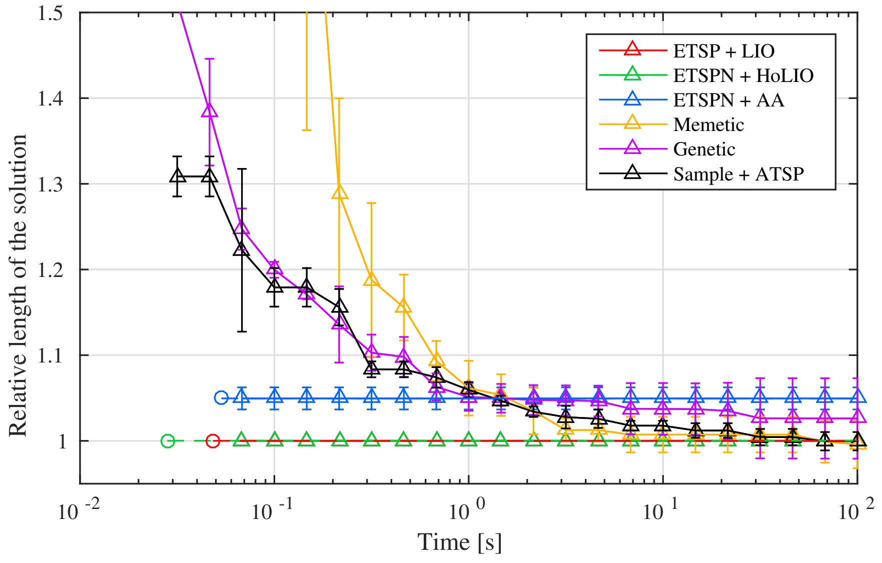
Figure 7.6: Average required computational time (from 20 trials) for the instance of the DTSPN with 4 circle regions and D_4 constraint.

compared with all other implemented algorithms described above. For each scenario several random problem instances have been created with the minimum turning radius ρ set to $\rho = 1$. For each trial a new instance is generated, so the number of generated instances is equal to the number of trials. The scenario is defined by the number of regions n and the minimum allowed mutual distance between the goal regions. All the tour length are expressed as the ratio to the tour length generated by the proposed algorithm. Hence, the proposed algorithm has always the ratio equal to one. All the presented results are averaged from several instances and the standard deviation is computed.

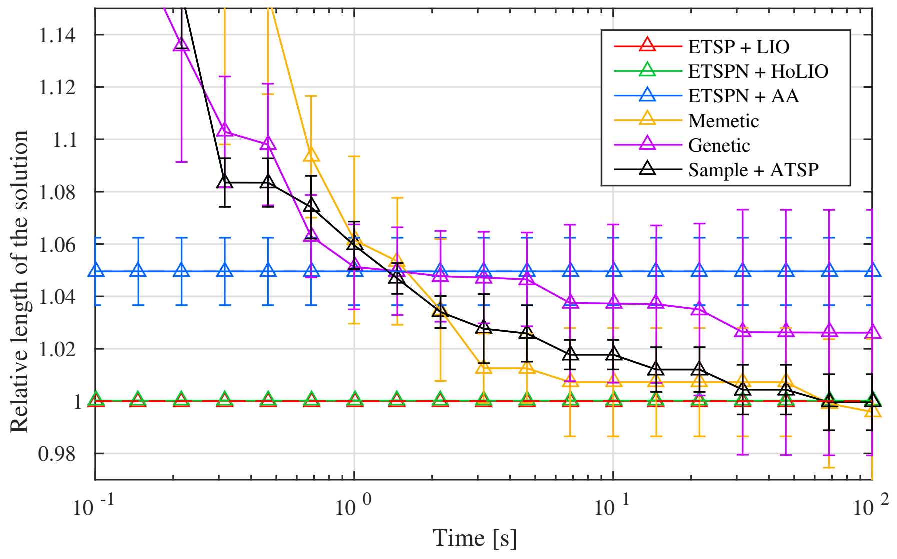
In the first scenario, we compared the quality of solutions regarding the dedicated computational resources for the instances of the DTSPN in which the D_4 constraint holds. The results for 20 and 40 convex regions are shown in Fig. 7.7 and Fig. 7.8, respectively. From the results, it can be seen that all three decoupled algorithms find an initial solution and it is not improving any more. For 20 regions the solution is found in less than 100 milliseconds. In contrast, the other algorithms (genetic, memetic and sampling-based) continue until the time limit of 100 seconds is exceeded.

We further evaluated the scalability of the algorithms with the increasing number of the goal regions n from 5 to 50. Instances with the D_4 constraint were used and the computation time has been limited to 10 seconds. The results are shown in Fig. 7.9.

So far, random instances with D_4 and convex regions were used. Here, we relax the D_4 constraint in order to evaluate the algorithms on instances according to the minimum mutual distance d of the goal regions. Hence, the instances meets the D_d constraint. The results are shown in Fig. 7.10. Although, the LIO algorithm has been designed on top of the found properties of the optimal solution of the DTSPN with the D_4 constraint, the results for the relaxed constraint indicate suitability of the proposed algorithm also for a more general problems with closer regions.



(a) General view



(b) Focused view

Figure 7.7: Average ratio of the tour length (from 50 trials) according to the ETSP+LIO solution for the DTSPN with n=20 convex regions. Plots start from the time when the first solution is available.

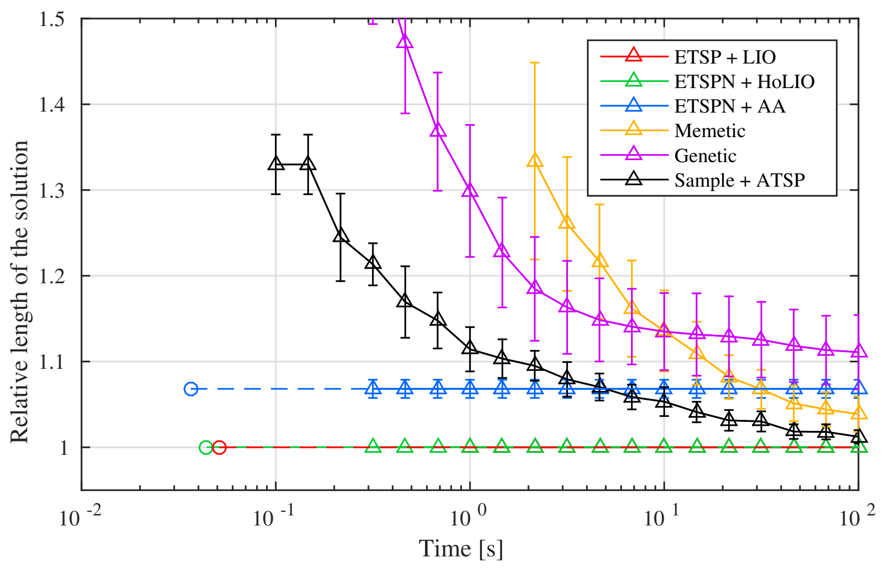


Figure 7.8: Average ratio of the tour length (from 50 trials) according to the ETSP+LIO solution for the DTSPN with n=40 convex regions. Plots start from the time when the first solution is available.

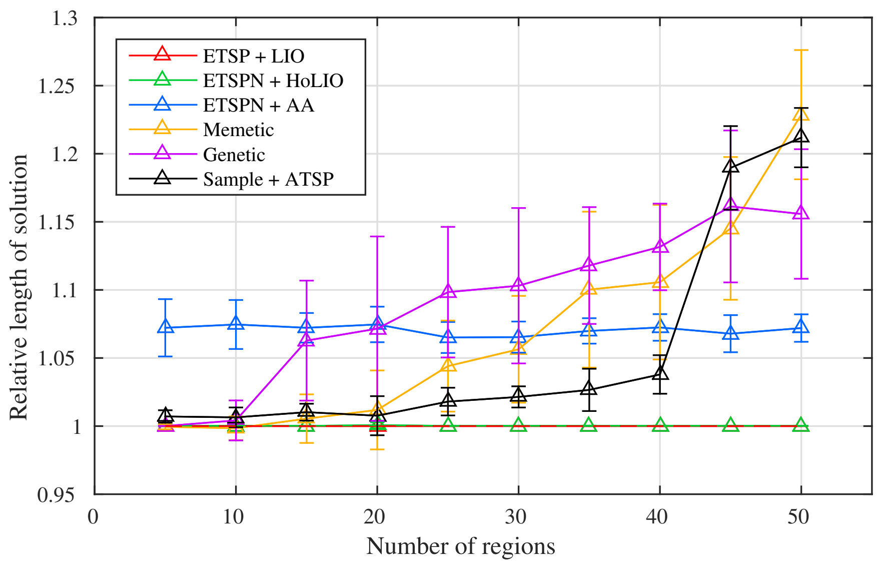


Figure 7.9: Average ratio of the tour length (from 20 trials) according to the ETSP+LIO solution for problems with increasing number of regions. The time limit is 10 seconds.

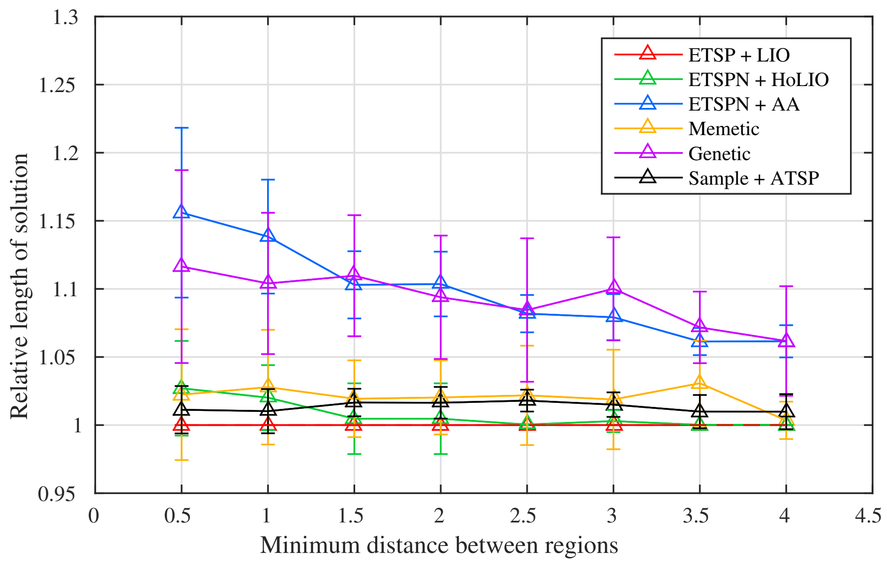


Figure 7.10: Average ratio of the tour length (from 20 trials) according to the ETSP+LIO solution for instances of the DTSPN with 20 regions increasing minimal mutual distance of the regions. The time limit is 10 seconds.

7.4 Discussion

The presented results support feasibility of the proposed decoupled algorithm, called ETSPN+LIO, for the DTSPN. It transforms the DTSPN into the DTRP using a sequence estimated by solving the ETSP for centers of the goal regions. The DTRP is then solved by the proposed LIO algorithm which adjusts the vehicle heading and entry point to the goal regions. The LIO algorithm provides solution of competitive quality with significantly lower computation requirements (about three orders of magnitude lower) then the evolutionary and sampling based approaches.

Although the LIO has been designed on top of the found properties of the optimal solution of the DTSPN with the D_4 constraint, the results for the relaxed constraint indicate suitability of the proposed approach also for general problems. It can be seen in Fig. 7.10 that the LIO provides better results than other approaches even for $D_{0.5}$ and D_1 constraints if the time is limited.

The most computationally demanding part of the proposed LIO approach is a solution of the ETSP, which increases with the number of the goal regions. The Concorde solver was used in experiments, which solves the ETSP to optimum. It can be replaced by approximation algorithms to speed up the computations, while the quality of the solution is still satisfactory. For example the LKH solver [22] can be utilized.

CHAPTER 8

Dealing with obstacles

In this chapter, we study the DTSPN in the environment with obstacles. The general multi-goal path planning problem with obstacles was considered in [4, 5]. But, to the best of our knowledge, there is only one approach [41] that explicitly considers the DTSPN with obstacles. The author proposed to use sampling based method, similarly to the case without obstacles, and create samples even outside the given regions to be visited. All the samples form a roadmap and that is then used to solve to problem as the ATSP. Although this approach provides a feasible solution and in particular cases also solutions of good quality, the DTSPN with obstacles is a challenging problem. The final path is constructed as a sequence of Dubins maneuvers passing not only particular entry configurations in the goal regions, but also arbitrary configurations in the free configuration space to avoid collision with the obstacles.

A simple naive approach to address Dubins planning with obstacles is proposed in the next section to further show this difficulty and how it can be addressed by the sampling and genetic based algorithms. Then, the sampling based method [41] is described.

8.1 Naive based approach

We propose naive based approach to address the DTSPN with obstacles. The idea of the approach is based on the optimization problem defined in Problem 4 where all paths that are intersecting with obstacles are discarded. The obstacles are not considered in the original problem, and therefore, the Dubins maneuver is used between two configurations, which is the shortest possible path in an environment without obstacles. For the DTSPN with obstacles, the same concept is used, but only collision-free Dubins maneuvers considered. Since the Dubins maneuvers can pass through obstacles and have to be discarded, unnecessary long paths are preserved and thus this naive approach is only a heuristic method. Moreover, it can be simple shown that an existence of a feasible solution is not guaranteed. The example instance containing this issue is shown in Fig. 8.1. In this example, there is not collision-free Dubins maneuver between two disc goal regions (denoted by brown color), because it necessarily goes throw the obstacle denoted by green color. However, there exist a collision-free path for the Dubins vehecle, which is not provided by the simple naive based approach. In the following section, we discuss two extensions of this very simple idea that are able to provide feasible solution with great probability. The first approach utilized sampling-based method while the secont approach combines idea of genetic algorithms.

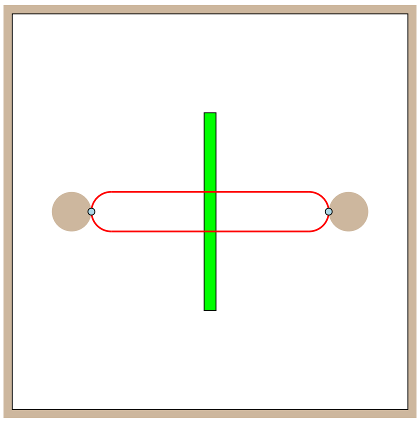


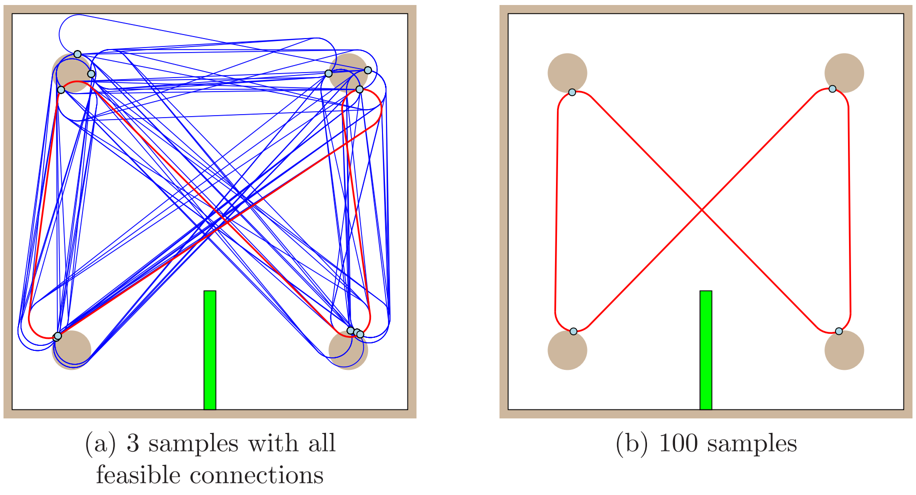
Figure 8.1: Example of an unfeasible solution of the DTSPN with obstacles

8.1.1 Sampling-based methods

The first method, which follows the idea of the the naive approach, is adapted version of the sampling-based method described in Section 5.2. As well as in the original method, the goal regions are randomly sampled. They can be sampled anywhere in the regions or only at the boundaries. Based on evaluation of these two sampling strategies, both variants produced solutions of competitive quality. The sampled configuration are connected by Dubins maneuvers into a roadmap which is considered as an asymmetric version of the GTSP. The GTSP is transformed into ATSP and solved by available solvers, e.g., LKH [22].

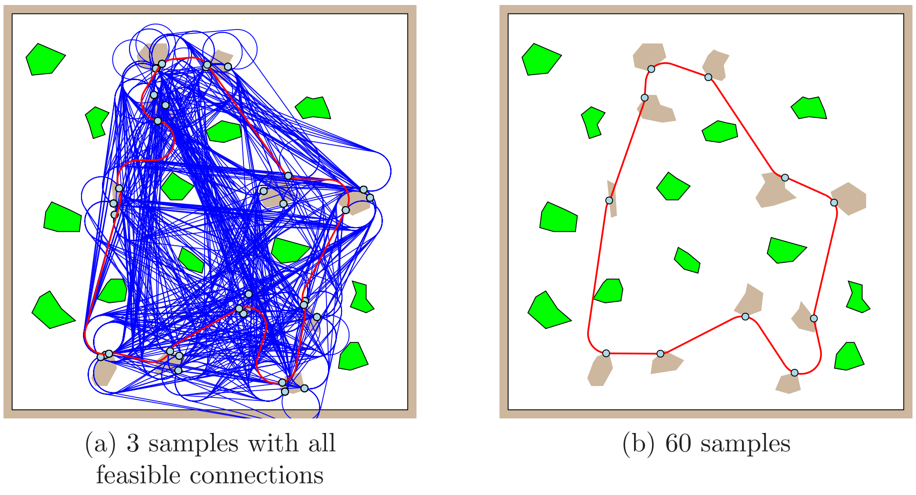
Although, the original sampling-based method for the DTSPN without obstacles is resolution complete [41], the adapted version considering the obstacles does not necessarily converges into an optimal solution. Such a situation is shown in Fig. 8.2. On the left, there is a full roadmap in blue, which was created from 3 samples for each goal regions. The shortest path in the roadmap visiting the goal regions is shown red. On the right, the shortest path for the Dubins vehicle found for the roadmap with 100 samples per goal region is depicted.

The naive sampled-based method also does not guarantee that a feasible solution will be found, if such a solution exists. Nevertheless, the method can be used for instances of the DTSPN with obstacle. Examples of problems with 11 goal regions and 12 obstacle regions are shown in Fig. 8.3.


(a) 3 samples with all feasible connections

(b) 100 samples

Figure 8.2: Example instances of the DTSPN with obstacles solved by the naive sampling-based method with different number of samples per goal regions.


(a) 3 samples with all feasible connections

(b) 60 samples

Figure 8.3: Large instances of the DTSPN with obstacles solved by the naive sampling-based method with different number of samples per goal regions.

8.1.2 Genetic methods

We also consider the idea of the naive approach for the DTSPN with obstacles in combination of the genetic approach described in Section 5.3. The original version of the algorithm have been proposed for hte DTSPN without obstacles, and therefore, we need to modify it appropriately to discard solution that pass through the obstacles. However, in genetic algorithm, we cannot directly forbid unfeasible paths colliding with obstacles. Therefore, we adjust the fitness function used for the evaluation of individuals. Originally, the the fitness function was equal to the length of the Dubins path $\mathcal{L}(q, \Sigma)$ going through one configuration for each goal region, defined in Equation 3.1. We modify the fitness function by doubling it for each intersection with the obstacle. Hence, the algorithm prefers solutions with less number of collisions. The proposed fitness function f can be express by the following equation, where K denotes the number of collisions:

$$f = \mathcal{L}(q, \Sigma) \cdot 2^K. \tag{8.1}$$

An example instance of the DTSPN with obstacles is shown in Fig. 8.4 together with solution generated by the adapted genetic algorithm.

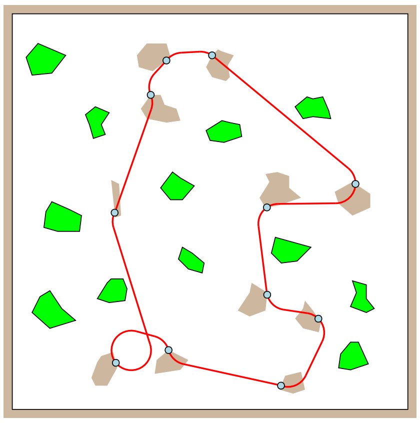


Figure 8.4: A larger instance of the DTSPN with obstacles solved by the modified genetic algorithm. The timelimit was 10 s.

8.2 Sampling-based approach with obstacles

The main disadvantage of the naive based approach is the fact that it is capable to find any feasible solution for some instances of the DTSPN with obstacles. In [41], the author proposed new method to address the problem. The sampling-based method was extended to make two types of vehicle configuration samples. The first type of the samples are configurations inside the goal regions which was also used in the original sampling-based approach without the obstacles. Let denote such a configuration $r_{i,j}$ which represents the j-th sample inside the i-th goal region. The second type of samples are configuration outside the goal regions. Let denote them s_i which represents the i-th sample of this type. In the next step, the complete roadmap is created by applying the Dubins maneuver. All maneuvers that are intersecting with obstacles are discarded. Further, a path between any pair of r samples is determined in the roadmap using graph path planning algorithm, e.g., Dijkstra or A*. Once the paths are found, the problem can be considered as an asymmetric version of the GTSP. From this point, the process is the same as for the original sampling-based approach without the obstacles [40]. Finally, the GTSP is transformed into the ATSP and it can be solved by existing solvers. This approach is similar to the classical Probabilistic Roadmaps (PRM) introduced in [45]. Ideas of the PRM can be further used to speed up the process of construction the roadmap between r samples.

CHAPTER 9

Conclusion

The diploma thesis deals with the problem of optimal path planning for a non-holonomic vehicle in surveillance missions. We consider the surveillance mission as the multi-goal planning problem for the Dubins vehicle which we formulate as the Dubins traveling salesman problem with neighbourhoods (DTSPN). The main challenge of the DTSPN is related to the nonholonomic constraint of the vehicle and the combinatorial nature of the problem. It is necessary to determine both the order of visits the given goal regions and also the particular configurations of the vehicle that guarantee visitation of each region. The configurations consist of entry points accompanied by the vehicle orientation, where both values may significantly influence the final cost of the solution. Because these values can be selected from infinite sets, existing discrete combinatorial approaches for the TSP cannot be directly applied to solve this challenging infinite combinatorial optimization problem.

The main contribution of the thesis is our approach to address the DTSPN which is based on the analysis of the optimal solution of the DTSPN. More specifically, we focus on the restricted variant of the problem with the D_4 constraint. The found properties of the optimal solution are then used in the developed algorithms and for detection of non-optimal solutions in existing methods.

We have proposed a decoupled approach to address the DTSPN which uses the newly designed Local Iterative Optimization (LIO) algorithm. The LIO algorithm has been designed on top of the found properties of the optimal solution for the DTSPN with the D_4 constraint. The performance of the LIO based approach have been evaluated in a series of scenarios and the presented results support a feasibility of the proposed approach. Although, the LIO was originally designed for problem instances with the D_4 constraint, the results indicate it is also suitable for general instances of the DTSPN with non-overlapping goal regions.

We have compared the LIO based approaches with several state-of-the-art approaches found in literature. In particular, we considered sampling-based, genetic-based, and decoupled methods. According to the results of the comparison, the LIO based approach provides competitive solutions to the existing approaches while its computational requirements are significantly lower.

Finally, the DTSPN with obstacles was also considered in the thesis. We have designed a naive based approach to address the problem. Although the propose naive based approach does not guarantee a solution is found if such a solution exists, to the best of our knowledge, there is not an algorithm based on computation of the Dubins maneuvers that provides such a guarantee. We have investigated the sampling-based approach with two types of samples which can solve this issue. Unfortunately, this approach have not been appropriately studied so far and it is still an open problem.

9.1 Suggestion for future work

A multi-goal path planning problem for a nonholonomic vehicle suggests variety of research directions that need to be pursued to make it suitable for a wider range of possible applications. One such direction would be to focus on developing a fast algorithm for the DTSPN with obstacles. To the best of our knowledge, only one article [41] has been published to address this problem. The author proposed a sampling-based method with two types of samples, but the algorithm has not been evaluated exhaustively. We assume that there exist interesting possibilities to speed up the algorithm and evaluate this approach by experiments using real aerial vehicles in an environment with obstacles.

Bibliography

[1] Der-Tsai Lee and Arhurk Lin. Computational complexity of art gallery problems. *Information Theory, IEEE Transactions on*, 32(2):276–282, 1986.

[2] Howie Choset and Philippe Pignon. Coverage path planning: The boustrophedon cellular decomposition. In *Field and Service Robotics*, pages 203–209. Springer, 1998.

[3] Brendan Englot and Franz S Hover. Sampling-based coverage path planning for inspection of complex structures. In *ICAPS*, 2012.

[4] Steven N Spitz and Aristides AG Requicha. Multiple-goals path planning for coordinate measuring machines. In *ICRA*, volume 3, pages 2322–2327. IEEE, 2000.

[5] Mitul Saha, Tim Roughgarden, Jean-Claude Latombe, and Gildardo Sánchez-Ante. Planning tours of robotic arms among partitioned goals. *The International Journal of Robotics Research*, 25(3):207–223, 2006.

[6] David S Johnson and Lyle A McGeoch. The traveling salesman problem: A case study in local optimization. *Local search in combinatorial optimization*, 1:215–310, 1997.

[7] Lester 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*, pages 497–516, 1957.

[8] Karl J Obermeyer. Path planning for a uav performing reconnaissance of static ground targets in terrain. In *AIAA Guidance, Navigation, and Control Conference*, pages 10–13, 2009.

[9] Jacob T Schwartz and Micha Sharir. On the "piano movers" problem. ii. general techniques for computing topological properties of real algebraic manifolds. *Advances in applied Mathematics*, 4(3):298–351, 1983.

[10] Steven M LaValle. *Planning algorithms*. Cambridge university press, 2006.

[11] X.N. Bui, P. Souères, J-D. Boissonnat, and J-P. Laumond. The shortest paths synthesis for non-holonomic robots moving forwards. *ICRA*, 1994.

[12] Eugene L Lawler, Jan Karel Lenstra, AHG Rinnooy Kan, and David B Shmoys. *The traveling salesman problem: a guided tour of combinatorial optimization*, volume 3. Wiley New York, 1985.

[13] George Dantzig, Ray Fulkerson, and Selmer Johnson. Solution of a large-scale traveling-salesman problem. *Journal of the operations research society of America*, 2(4):393–410, 1954.

[14] Shen Lin. Computer solutions of the traveling salesman problem. *Bell System Technical Journal, The*, 44(10):2245–2269, 1965.

[15] Gerhard Reinelt. Tsplib—a traveling salesman problem library. *ORSA journal on computing*, 3(4):376–384, 1991.

[16] Clair E Miller, Albert W Tucker, and Richard A Zemlin. Integer programming formulation of traveling salesman problems. *Journal of the ACM (JACM)*, 7(4):326–329, 1960.

[17] Eugene L Lawler and David E Wood. Branch-and-bound methods: A survey. *Operations research*, 14(4):699–719, 1966.

[18] D. Applegate, R. Bixby, V. Chvátal, and W. Cook. CONCORDE TSP Solver. `http://www.math.uwaterloo.ca/tsp/concorde.html`. [cited 31 Mar 2015].

[19] Shen Lin and Brian W Kernighan. An effective heuristic algorithm for the traveling-salesman problem. *Operations research*, 21(2):498–516, 1973.

[20] Keld Helsgaun. An effective implementation of the lin–kernighan traveling salesman heuristic. *European Journal of Operational Research*, 126(1):106–130, 2000.

[21] Nicos Christofides and Samuel Eilon. Algorithms for large-scale travelling salesman problems. *Operational Research Quarterly*, pages 511–518, 1972.

[22] Keld Helsgaun. LKH solver 2.0.7. `http://www.akira.ruc.dk/~keld/research/LKH`. [cited 31 Mar 2015].

[23] Gilbert Laporte, Hélène Mercure, and Yves Nobert. Generalized travelling salesman problem through n sets of nodes: the asymmetrical case. *Discrete Applied Mathematics*, 18(2):185–197, 1987.

[24] Charles E Noon and James C Bean. A lagrangian based approach for the asymmetric generalized traveling salesman problem. *Operations Research*, 39(4):623–632, 1991.

[25] Gregory Gutin and Daniel Karapetyan. A memetic algorithm for the generalized traveling salesman problem. *Natural Computing*, 9(1):47–60, 2010.

[26] Daniel Karapetyan and Gregory Gutin. Lin–kernighan heuristic adaptations for the generalized traveling salesman problem. *European Journal of Operational Research*, 208(3):221–232, 2011.

[27] Ketan Savla, Emilio Frazzoli, and Francesco Bullo. On the point-to-point and traveling salesperson problems for dubins' vehicle. In *Proceedings of the American Control Conference*, pages 786–791. IEEE, 2005.

[28] Xavier Goaoc, Hyo-Sil Kim, and Sylvain Lazard. Bounded-curvature shortest paths through a sequence of points using convex optimization. *SIAM Journal on Computing*, 42(2):662–684, 2013.

[29] Xiang Ma and David A Castanon. Receding horizon planning for dubins traveling salesman problems. In *45th IEEE Conference on Decision and Control*, pages 5453–5458, 2006.

[30] Moshe Dror, Alon Efrat, Anna Lubiw, and Joseph SB Mitchell. Touring a sequence of polygons. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 473–482. ACM, 2003.

[31] Jerome Le Ny, Eric Feron, and Emilio Frazzoli. On the dubins traveling salesman problem. *IEEE Trans. Automat. Contr.*, 57(1):265–270, 2012.

[32] Xin Yu and JY Hung. Optimal path planning for an autonomous robot-trailer system. In *38th Annual Conference on IEEE Industrial Electronics Society (IECON)*, pages 2762–2767, 2012.

[33] Esther M Arkin and Refael Hassin. Approximation algorithms for the geometric covering salesman problem. *Discrete Applied Mathematics*, 55(3):197–218, 1994.

[34] Adrian Dumitrescu and Joseph SB Mitchell. Approximation algorithms for tsp with neighborhoods in the plane. In *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, pages 38–46. Society for Industrial and Applied Mathematics, 2001.

[35] Khaled Elbassioni, Aleksei V Fishkin, Nabil H Mustafa, and René Sitters. Approximation algorithms for euclidean group tsp. In *Automata, Languages and Programming*, pages 1115–1126. Springer, 2005.

[36] Joseph SB Mitchell. A ptas for tsp with neighborhoods among fat regions in the plane. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 11–18. Society for Industrial and Applied Mathematics, 2007.

[37] Bo Yuan, Maria Orlowska, and Shazia Sadiq. On the optimal robot routing problem in wireless sensor networks. *Knowledge and Data Engineering, IEEE Transactions on*, 19(9):1252–1261, 2007.

[38] Jan Faigl, Vojtěch Vonásek, and Libor Přeučil. Visiting convex regions in a polygonal map. *Robotics and Autonomous Systems*, 61(10):1070–1083, 2013.

[39] Douglas Guimaraes Macharet, Armando Alves Neto, Vilar Fiuza da Camara Neto, and Mario 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*, pages 377–384. ACM, 2012.

[40] Jason T Isaacs, Daniel J Klein, and Joao P Hespanha. Algorithms for the traveling salesman problem with neighborhoods involving a dubins vehicle. In *American Control Conference (ACC), 2011*, pages 1704–1709. IEEE, 2011.

[41] Karl J Obermeyer, Paul Oberlin, and Swaroop Darbha. Sampling-based path planning for a visual reconnaissance unmanned air vehicle. *Journal of Guidance, Control, and Dynamics*, 35(2):619–631, 2012.

[42] Xing Zhang, Jie Chen, Bin Xin, and Zhihong Peng. A memetic algorithm for path planning of curvature-constrained uavs performing surveillance of multiple ground targets. *Chinese Journal of Aeronautics*, 27(3):622–633, 2014.

[43] Zbigniew Michalewicz and David B Fogel. *How to solve it: modern heuristics*. Springer Science & Business Media, 2004.

[44] Edo S Van der Poort, Marek Libura, Gerard Sierksma, and Jack AA van der Veen. Solving the k-best traveling salesman problem. *Computers & operations research*, 26(4):409–425, 1999.

[45] Lydia E Kavraki, Petr Svestka, J-C Latombe, and Mark H Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *Robotics and Automation, IEEE Transactions on*, 12(4):566–580, 1996.

APPENDIX A

Used symbols and abbreviations

Table A.1: Used symbols

Symbol	Description
$\mathcal{W}$	World in which the vehicle operates ($\mathbb{R}^2$)
$\mathcal{O}$	Finite set of obstacle regions
$\mathcal{A}$	Robot represented by a rigid body
$\mathcal{C}$	Configuration space
ρ	Minimal turning radius
$\mathcal{R}$	Set of regions to visit
R_i	Region i to be visited by Dubins' vehicle, $R_i \in \mathcal{R}$
δR_i	Boundary of the region $R_i \in \mathcal{R}$
n	Number of the regions (points)
p_i	Point where R_i is visited, $p_i \in \mathbb{R}^2$
θ_i	Heading of the Dubins' vehicle at p_i, $\theta_i \in \mathbb{S}^1$
q_i	Configuration of the vehicle $q_i = (p_i, \theta_i)$, $q_i \in SE(2)$
α_i	Position of q_i at the border of R_i, $\alpha \in \langle 0, 1)$
Σ	Sequence of the regions – $\Sigma = (\sigma_1, \ldots, \sigma_n)$, where $\sigma_i \neq \sigma_j$, $i \neq j$, and $1 \leq \sigma_i \leq n$, $1 \leq \sigma_j \leq n$
$\|\|p_i - p_j\|\|$	Euclidean distance between points p_i and p_j
$\mathcal{D}(q_i, q_j)$	Dubins maneuver connecting q_i and q_j
$\mathcal{L}(q_i, q_j)$	Length of the shortest path ($\mathcal{D}(q_i, q_j)$) connecting q_i and q_j for the Dubins vehicle
$\mathcal{L}(q, \Sigma)$	Length of the shortest path connecting configurations q in sequence Σ for the Dubins vehicle, defined by Equation 3.1

Table A.2: Used abbreviations

Abbreviations	**Description**
UAV	Unmanned Aerial Vehicle
MPP	Multi-goal path planning problem
CSP	Coverate sampling problem
TSP	Traveling salesman problem
ATSP	Asymmetric traveling salesman problem
ETSP	Euclidean traveling salesman problem
ETSPN	Euclidean traveling salesman problem with neighborhoods
DTSP	Dubins traveling salesman problem
DTSPN	Dubins traveling salesman problem with neighborhoods
TPP	Touring polygon problem
DTP	Dubins touring problem
DTRP	Dubins touring regions problem
AA	Alternating algorithm
LIO	Local iterative optimization

APPENDIX B

Content of the enclosed CD

```
CD
├─ ReadMe.txt
├─ thesis
│   ├─ thesis.pdf
│   └─ sources
├─ program
│   ├─ src
│   │   └─ ...source files of my program
│   ├─ DTSTP ...compiled program for linux-64bit
│   ├─ ruh.sh ...script to run the program
│   └─ dtspn.pdf ...the result path in the pdf format
└─ experiments
    ├─ plot
    │   └─ generateAllPlots.m
    ├─ test
    │   └─ ...scripts used for results generation
    └─ results
        └─ ...raw results in mat-files
```