Faculty of Electrical Engineering
Department of Computer Science

Bachelor's Thesis

Hybrid algorithm for the Dubins Traveling Salesman Problem with Neighborhoods

Daniel Váchal

January 2020

Supervisor: prof. Ing. Jan Faigl, Ph.D.

BACHELOR'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Váchal Daniel** Personal ID number: **420913**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Computer Science**

Study program: **Open Informatics**

Branch of study: **Software Systems**

II. Bachelor's thesis details

Bachelor's thesis title in English:

Hybrid algorithm for the Dubins Traveling Salesman Problem with Neighborhoods

Bachelor's thesis title in Czech:

Hybridní řešení úlohy obchodního cestujícího s Dubinsovým vozidlem

Guidelines:

1. Familiarize yourself with the Dubins Traveling Salesman Problem (DTSP) [1] and the DTSP with Neighborhoods (DTSPN) [2, 3].
2. Familiarize yourself with the algorithms [4] and [5] and available implementations.
3. Propose a combination of the memetic techniques [4] with the unsupervised learning [5] to improve the quality of solutions provided by [5] and decrease computational requirements of [4].
4. Implement the proposed solution and evaluate its performance in representative benchmarks of the DTSP(N).

Bibliography / sources:

[1] Savla, K., Frazzoli, E., and Bullo, F: On the point-to-point and traveling salesperson problems for Dubins? vehicle. American Control Conference, 2005, 786-791.
[2] Oberlin, P., Rathinam, S., and Darbha, S.: Today's traveling salesman problem. Robotics & Automation Magazine, 2010, 17(4):70-77.
[3] Isaacs, J. T., Klein, D. J., and Hespanha, J. P. Algorithms for the Traveling Salesman Problem with Neighborhoods Involving a Dubins Vehicle. American Control Conference, 2011, 1704-1709.
[4] Zhang, X., Chen, J., Xin, B., and Peng, Z.: A memetic algorithm for path planning of curvature-constrained uavs performing surveillance of multiple ground targets. Chinese Journal of Aeronautics, 2014, 27(3):622-633.
[5] Faigl, J., Váňa, P.: Unsupervised learning for surveillance planning with team of aerial vehicles. IJCNN 2017: 4340-4347.

Name and workplace of bachelor's thesis supervisor:

prof. Ing. Jan Faigl, Ph.D., Artificial Intelligence Center, FEE

Name and workplace of second bachelor's thesis supervisor or consultant:

Date of bachelor's thesis assignment: **31.01.2018** Deadline for bachelor thesis submission: **07.01.2020**

Assignment valid until: **07.02.2020**

prof. Ing. Jan Faigl, Ph.D. Supervisor's signature	Head of department's signature	prof. Ing. Pavel Ripka, CSc. Dean's signature

III. Assignment receipt

The student acknowledges that the bachelor's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor's thesis, the author must state the names of consultants and include a list of references.

. _____
Date of assignment receipt

_____
Student's signature

" Time is the kindest thing of all, it will eventually heal every sorrow. Time is the cruelest thing of all, it will make everything fade away."

-TOKISAKI KURUMI, DATE A LIVE (ENCORE OVA)

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze 7. ledna 2020

Abstract

The thesis presents the Dubins Traveling Problem (DTSP) and its extension, the Dubins Traveling Salesman Problem with Neighborhoods (DTSPN). The thesis focuses on solving the DTSPN and studies two existing algorithms for the problem. One algorithm is the unsupervised learning algorithm based on Self-Organizing Maps (SOM). The second studied algorithm is a Memetic algorithm based on crossover and mutation operators to solve the DTSPN. The advantages and disadvantages of these two approaches can be considered complementary. Based on that, a novel Hybrid algorithm for solving the DTSPN is proposed to combine the two algorithms into a single one, that adopts the advantages of both.

Keywords: dubins traveling salesman problem; dubins traveling salesman problem with neighborhoods; memetic algorithm; self-organizing map; hybrid algorithm; unsupervised learning

Abstrakt

Tato práce pojednává o problematice problému obchodního cestujícího s Dubinsovým Vozidlem (DTSP) a jeho rozšíření na problém obchodního cestujícího s Dubinsovým vozidlem s okolími (DTSPN). Práce se zaměřuje na řešení minimalizační úlohy DTSPN a studuje dva existující algoritmy pro řešení tohoto problému. První algoritmus je založen na neřízeném učení postavený na samo-organizujících se mapách (SOM). Druhý, Memetický algoritmus, využívá operátory křížení a mutací. Výhody a nevýhody algoritmů se navzájem doplňují. V práci je toho využito v navrženém hybridním algoritmu pro řešení DTSPN, který kombinuje výhody obou stávajících algoritmů.

Klíčová slova: problém obchodního cestujícího s Dubinsovým vozidlem; problém obchodního cestujícího s Dubinsovým vozidlem s okolími; memetický algoritmus; samo-organizující se mapy; hybridní algoritmus; neřízené učení

Contents

1 Introduction **1**

2 Problem Statement **3**

3 Related Work **5**

4 Source Algorithms **7**
4.1 Memetic Algorithm 7
4.2 Unsupervised learning algorithm 9
4.2.1 Self Organizing Maps for the TSP and the DTSP 9
4.2.2 SOM for the DTSPN 10

5 Proposed Hybrid Algorithm **13**

6 Results **17**
6.1 SOM initialization count for the Hybrid algorithm 17
6.2 Basic settings 18
6.3 Changing the density of the map 20
6.4 Changing the number of targets on the map 21
6.5 Changing the turning radius 22
6.6 Changing the sensing radius 23
6.7 Discusion 23

7 Conclusion **25**

Bibliography **27**

List of Figures

1.1 [1]: Carbonix Volanti: fixed wing carbon composite industrial drone 1

2.1 An example of the Dubins manuever connecting points p_i and p_j using the departure angle θ_i and arrival angle θ_j, adapted from [2] 4

3.1 A solution for the DTSP for a given sequence of the targets, using the uniformed sampling and the informed sampling. Source of the picture: [2] 6

4.1 In a chromosome, two indices $i = 3, j = 6$ are chosen and the mutation is done around them, creating a new chromosome. . . . 8

4.2 Representation of the crossover operator. In step 1, red gene is chosen from parent 1. In step 2, red genes are no longer considered, as they were used already. The first viable gene is chosen from the parent 2 according to the sequence generated at the start of the operator. In the example, (1,2,2,1,2). By repeating the process the offspring is created. 9

4.3 Example of the ring evolution towards the targets. After the adaptation process is finished the headings are determined, and the final Dubins path is constructed [3] 11

4.4 A search graph showing how the headings are connected in the neuron ring, source: [3] 11

4.5 Graphic showing the winner selection procedure and the point o_p towards which the network is adapted [3] 12

5.1 Ilustration of the best solutions provided by the Memetic algorithm and the Hybrid algorithm in their initial population. This example has the basic setting from which all other experiments are made. That is $\delta = 4$, $\rho = 4$, with 30 targets. 14

5.2 The visiting point of the target O_i is moved while converting the solution from one representation to the other. The original location of the visiting point is the neuron ν_i. The converting function moves the visiting point to the point p_i on the boundary of the neighborhood in the direction of the heading, resulting in the change of Dubins path. 15

6.1 Test of the Hyrid algorithm with different number of SOM initializations, conducted on 10 intsances, with settings $n = 100, D = 20, \delta = 4, \rho = 4$. 18
6.2 Test of the Hyrid algorithm with few and many SOM initializations, conducted on 10 instances, with settings $n = 100, D = 20, \delta = 4, \rho = 4$. 18
6.3 Test conducted on 10 maps with basic settings $n = 30, D = 20, \delta = 4, \rho = 4$. 19
6.4 Test conducted on 100 instances with settings $n = 30, \delta = 4, \rho = 4$ for 10 different values of D. 20
6.5 Test conducted on 100 instances with settings $D = 20, \delta = 4, \rho = 4$ for 10 different values of n. 21
6.6 Test conducted on 100 instances with settings $n = 30, D = 20, \delta = 4$ for 10 different values of ρ. 22
6.7 Test conducted on 100 instances with settings $n = 30, D = 20, \rho = 4$ for 10 different values of δ. 23

List of Algorithms

1 The function transforming the representation of the solution from the SOM alghorithm into the Memetic 15
2 The pseudocode of the Hybrid algorithm describing also the Memetic part presented in [4] 16

Abbreviations

UAV	Unmanned Aerial Vehicle
TSP	Traveling Salesman Problem
TSPN	Traveling Salesman Problem with Neighborhoods
DTSP	Dubins Traveling Salesman Problem
DTSPN	Dubins Traveling Salesman Problem with Neighborhoods
SOM	Self-Organizing Map
GSOA	Growing Self-Organizing Array
DTP	Dubins Touring Problem
AA	Alternating Algorithm
ATSP	Asymmetric Traveling Salesman Problem
ETSP	Euclidean Traveling Salesman Problem
GTSP	Generalised Traveling Salesman Problem
GTSPN	Generalised Traveling Salesman Problem with Neighborhoods

Symbols Used

ρ	Turning radius limit
n	Natural number
δ	Sensing radius
v	Constant forward velocity
q	State of the Dubins vehicle
p	Position of the Dubins Vehicle in $\mathbb{R}^2$ (waypoint)
(x, y)	Coordinates in $\mathbb{R}^2$
θ	Heading of the Dubins vehicle
$SE(2)$	Special Euclidean Space $\mathbb{R}^2 \times S$
S	Sequence of waypoints s_i
P	Set of waypoints p_i
u	Control input
O	Target location
r	Vehicle number
N	Neural network (neuron ring)
ν_i	Vehicle configuration in the input space
m	Number of neurons in the ring
σ	Learning gain
μ	Learning rate
α	Gain decreasing rate
ν^*	Winner neuron
Θ_v	Set of heading values of the neuron v
p_o	Closest point on the Dubins path to the target o
ν_{prev}	Neuron on the path previous to the winner neuron determined so that the length of the Dubins path is minimized
ν_{next}	Neuron on the path next to the winner neuron determined so that the length of the Dubins path is minimized
D	The target density in testing instances
Q	Set of all possible configurations q_i of the Dubins Vehicle.

CHAPTER 1

Introduction

The problem addressed in this thesis is motivated by surveillance missions performed by Unmanned Aerial Vehicles (UAVs). In surveillance missions, the goal is to visit a set of locations to gather information on the objects of interest. An example can be data collection of household energy consumption, where data can be collected remotely from measuring devices. Another instance of a surveillance mission is taking snapshots of the given target locations. In that case, it is not necessary to visit the location of the object exactly, but it is sufficient to reach a location from which the object can be photographed with the requested level of details. There can be several optimization criteria used in surveillance mission planning. The most straightforward way is to shorten the time UAV spends on the mission because keeping the UAV airborne is the most expensive part of the mission. Therefore the mission needs to be planned such that the UAV visits all the required target locations with minimal possible time. If the UAV is moving with constant speed, the problem can be considered as a minimization of the total travel cost to visit all the target locations. A common type of UAV used for the surveillance missions is a fixed-wing aircraft, which is constrained by its turning radius. An example of such fixed-wing aircraft is shown in Figure 1.1.

The problem of finding minimal tour length visiting all the given locations is the Traveling Salesman Problem (TSP), which is known to be NP-hard (in its decision variant) [5], with

Figure 1.1: [1]: Carbonix Volanti: fixed wing carbon composite industrial drone

several existing approaches [6], [7]. However, the motivational scenario of the here studied problem is to find a smooth multi-goal trajectory that is suitable for UAVs such as fixed-wing aircraft, which is constrained by its minimal turning radius. Therefore, we focus on a solution of the TSP-like problems with curvature-constrained trajectories for which we consider the vehicle motion constraints modeled as Dubins vehicle [8].

Dubins vehicle models a vehicle that moves only forward with a constant speed and is limited by a minimum turning radius ρ. Then, the TSP becomes the Dubins Traveling Salesman Problem (DTSP) [9]. The task is to find the shortest path connecting a given set of points with a curvature constraint, such that the path visits all the given locations. In this case, when it is not required to visit the exact locations, we can save additional resources by only visiting their vicinity. That leads to a generalization of the DTSP, where particular waypoints can be chosen from an area surrounding the object of interest. This generalization is called the Dubins Traveling Salesman Problem with Neighborhoods (DTSPN) [10].

A novel hybrid algorithm for the DTSPN is introduced in this thesis. It leverages on to the existing methods: Memetic algorithm [4] and Unsupervised learning-based algorithm, based on Sel-Organizing Maps (SOM) [3]. Although, the SOM-based algorithm has evolved and the authors have introduced an updated version, called GSOA: Growing Self-Organizing Array - Unsupervised learning for the Close Enough Traveling Salesman Problem and other routing problems, we use the original abbreviation (SOM) for the rest of the thesis.

Both of these algorithms have their pros and cons. But these pros and cons can be considered complementary to each other. While the SOM algorithm can be considered a quick constructive heuristic, it does not improve the provided solution with more computational time at its disposal. The memetic algorithm is relatively slow in providing the first competitive solution, but with enough computational time, it can converge to high-quality solutions, eventually to the optimum. In chapter 4, both algorithms are presented to the reader in detail, as they are essential for the rest of the thesis.

Chapter 5 presents the proposed hybrid algorithm. The Hybrid generates solutions using the SOM and uses them for the initial population of the memetic part. It can provide high-quality solutions quickly, while still being able to improve upon them, when provided with more computational time.

In chapter 6, the empirical results of the proposed solutions are reported and discussed. The proposed hybrid algorithm is compared to the two algorithms from which it was developed. In chapter 7, the thesis is concluded.

CHAPTER 2

Problem Statement

The problem studied in this thesis is motivated by surveillance missions performed by the UAVs, where a set of locations to visit is given. The problem of connecting all the points in a plane and determining the order of their visits is known as the Traveling Salesman Problem (TSP). The target location does not always have to be visited directly. Hence, it is enough to approach the object of interest from a certain distance, and the particular waypoint can be chosen from an area surrounding the target. With this generalization, the problem becomes the Traveling Salesman Problem with Neighborhoods (TSPN). The shape of the surrounding area can vary in real situations, but for the computation, it has to be determined first. For simplicity, we consider that the area around each target is of the disk shape with a center in the target location with a given radius δ. In our case, each location has to be visited by the UAV, and the curvature constrained non-holonomic vehicle.

A mathematical model of the vehicle is needed for computations. Such a model has been proposed in [8]. The model is called the Dubins vehicle, which is a vehicle moving always forward with constant speed v and its motion limited by the minimal turning radius ρ. At each point in time, the state of the vehicle is described by its position in the plane and its heading. The represantation of the state is $q = (p, \theta)$, where $p \in \mathbb{R}^2$ is the position $p = (x, y)$ and $\theta \in \mathbb{S}^1$ is the heading of the vehicle, i.e., $q \in SE(2)$, that can be expressed by the equation [2].

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = v \begin{bmatrix} \cos\theta \\ \sin\theta \\ u\,\rho^{-1} \end{bmatrix}, \quad |u| \leq 1, \tag{2.1}$$

where u is the control input.

Dubins shows in [8], that the optimal path connecting two states $q_1 \in SE(2)$ and $q_2 \in SE(2)$ can be constructed only of the circular segments (C) with maximal possible turning radius and straight-line segments (L). Where the circular segments are either turn to the left (L) or to the right (R). Path constructed from these segments connecting two states of the Dubins vehicle is further called Dubins maneuver. Dubins maneuver can be of two kinds. One composed of only circular segments (CCC) and the other composed of two circular segments connected by a straight line (CSC). Thus, the options for Dubins maneuver are LRL, RLR for the CCC case and LSL, LSR, RSL, RSR for the CSC case. For a demonstration, an example

of the Dubins maneuver is visualized in Figure 2.1.

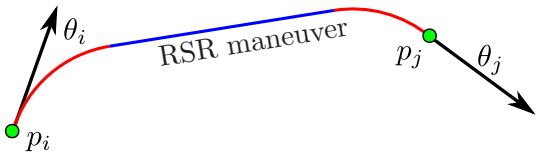


Figure 2.1: An example of the Dubins manuever connecting points p_i and p_j using the departure angle θ_i and arrival angle θ_j, adapted from [2]

The DTSP stands to find the optimal sequence, of the visits to targets as in the TSP. DTSP includes determination of the optimal headings at each target, to connect all the points by a Dubins path. The Dubins Traveling Problem with Neighborhoods (DTSPN) is the problem to find the shortest possible curvature-constrained path connecting all the target regions.

A formal definition of the DTSPN is adopted from [10], as follows

Problem 1 (DTSPN)

$$\begin{aligned} \underset{\Sigma,Q}{\textit{minimize}} \quad & \mathcal{L}\left(\mathrm{q}_{\sigma_\mathrm{n}}, \mathrm{q}_{\sigma_1}\right) + \sum_{\mathrm{i}=1}^{\mathrm{n}-1} \left(\mathrm{q}_{\sigma_\mathrm{i}}, \mathrm{q}_{\sigma_{\mathrm{i}+1}}\right) \\ \textit{subject to} \quad & |p_i \in R_i, q_i| < \delta, \end{aligned}$$

where $\mathcal{R} = (R_1, \ldots, R_n)$ is the set of n regions $R_i \subset \mathbb{R}^2$ to be visited by Dubins vehicle. $\Sigma = (\sigma_1, \ldots, \sigma_n)$ is the ordered permutation of $\{1, \ldots, n\}$, p_i is the point of visit to the region R_i, $q_i \in SE\,(2)$ is the state of the Dubins vehicle, and δ is the sensing radius. The DTSPN is an optimization problem over all possible permutations Σ and all configurations $Q = \{q_1, \ldots, q_n\}$, where $\mathcal{L}\,(q_i, q_j)$ is Dubins distance between q_i and q_j.

CHAPTER 3

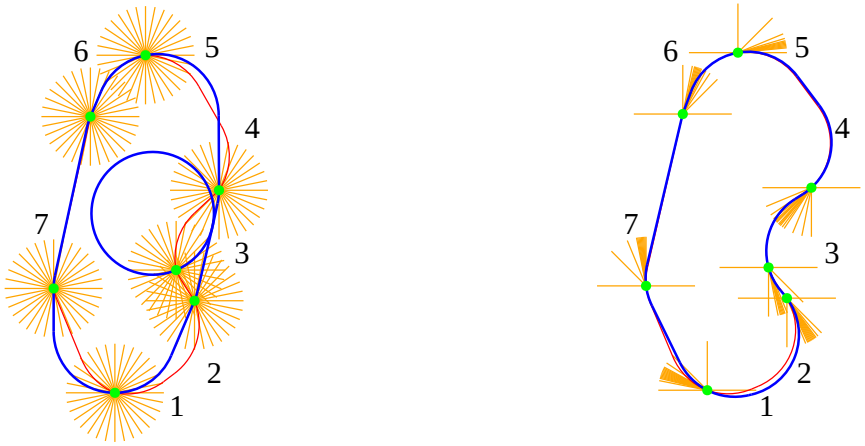
Related Work

In this chapter, existing approaches for the DTSP and the DTSPN are discussed. As Dubins has shown in [8], the shortest path for Dubins vehicle connecting two points in a plane is one of the six Dubins maneuvers consisting only of straight-line segments and arc curves with the minimal turning radius. However, the path expects that headings of the vehicle are known for both points connected by the Dubins maneuver. In the case of planning the path for the DTSP, the headings at waypoints are unknown. Thus, the solutions for the Euclidean TSP cannot be applied directly, and the problem of finding the optimal headings for each waypoint needs to be solved. For the sequence of n waypoint locations $p_1, \ldots, p_n$, the problem to find optimal headings $\theta_1, \ldots, \theta_n$ at each waypoint in order to minimize the total length of Dubins path connecting all the waypoints is a continuous optimization problem known as the Dubins Touring Problem (DTP) [2].

One of the first solutions to the DTP is the Alternating Algorithm (AA) described in [11]. Another solution to the DTP is used in [12]. The headings are determined for even edges and then optimal Dubins maneuvers are determined for odd edges. Another solution was proposed in [2], a refinement procedure to create an informed sampling method for solving the DTP. The comparison between the informed sampling and uniform sampling is shown in Figure 3.1. Using the informed sampling method, the authors claim to be able to find a solution as close to optimum as 0.1 %.

Approaches to solving the DTSP and the DTSPN found in literature can be divided into four main groups. The first group is decoupled approaches, which solve the problem of determining the headings and the problem of finding the sequence separately. The second group is formed by transformation approach algorithms where the idea is to sample headings of a possible set of discrete values first and then transform the problem to Asymmetric TSP (ATSP). The third group represents evolutionary algorithms with high computational requirements but with a chance of providing solutions of high quality. The fourth group is the approaches based on unsupervised learning of self-organizing maps.

The Decoupled approach to the DTSP is presented in [11], where the authors first obtain the upper bound on the point-to-point problem. The effectiveness of decoupling methods mainly relies on the similarities between the DTSP and the ETSP. It makes them unsuitable for situations where the Euclidean distance between waypoints is too rough of an approximation of the Dubins maneuver to the minimal turning radius.



(a) Uniform sampling

(b) Informed sampling

Figure 3.1: A solution for the DTSP for a given sequence of the targets, using the uniformed sampling and the informed sampling. Source of the picture: [2]

An example of the transformation approach to the DTSP can be a graph-based algorithm presented in [13]. The authors use a sampling method to cast the DTSPN to the Generalised Traveling Salesman Problem (GTSP) with intersecting node sets, which can be described by a directed graph. Then, using the Noon-Bean transformation, the GTSP is transformed into ATSP. And the optimal solution of the GTSP is recovered from the solution of the ATSP. Transformation methods are highly dependant on the sampling density of headings. To acquire better results, they usually require tremendous computational resources.

The third group represents evolutionary algorithms, which improve solutions by mutating existing solutions and creating crossbreeds of these solutions in generations. The fourth group is algorithms based on neural networks. We chose two algorithms from these last two groups to create the hybrid algorithm. As these algorithms are essential for this work, they are described in detail in the following chapter.

CHAPTER 4

Source Algorithms

From the known algorithms for solving the DTSPN, two are chosen in this thesis. The first is a Memetic algortihm [4]. It has been chosen for its ability to improve the provided solution in time. The second chosen algorithm is the unsupervised learning algorithm [3] that has been chosen for its ability to provide a competitive solution quickly. The proposed Hybrid algorithm combines these two algorithms into one, adopting advantages of both.

Memetic Algorithm

The chosen Memetic algorithm for the DTSPN has been presented in [4]. This approach first generates a population of random valid solutions to the DTSPN. Then, the individuals in the population can be mutated and crossbred, creating new individuals for the next population. The higher quality solutions are kept for the next generations, thus improving the found solution. The process can be repeated for the time provided to the algorithm, eventually converging to high-quality solutions. The main specifics of the chosen Memetic algorithm [4] are presented in the rest of this section.

The authors of [4] introduce several optimizations of the general evolutionary approach to address the DTSPN. The first introduced optimization is terminal heading relaxation. The article shows that reaching the target from an initial state with a fixed position and initial heading always reduces to circular arcs and straight-line segments. Also, these paths to the target are symmetric to both sides from the initial position. This is a special situation of the paths with terminal headings studied by Dubins [8]. This improvement allows determining the terminal heading from its relative position to the initial heading, once that is fixed. Therefore only the initial heading needs to be found, reducing the overall difficulty of the DTSPN.

The article presents and uses a boundary-based encoding scheme. Using the fact that the UAV has to always pass through the boundary of the target region, this point on the boundary can be used as the visiting point of each region. Using the position of the target as a center of the disk area of the neighborhood, the polar angle can be used to describe every point on the boundary. The encoding scheme is following. The waypoints $P = (p_1, p_2, \ldots, p_n)$, where $p_i = (x_i, y_i)$ is the visiting point of the region. Their sequence is $S = (s_1, s_2, \ldots, s_n)$ and θ_i represents the initial heading at the waypoint. Then the representation of the visiting point

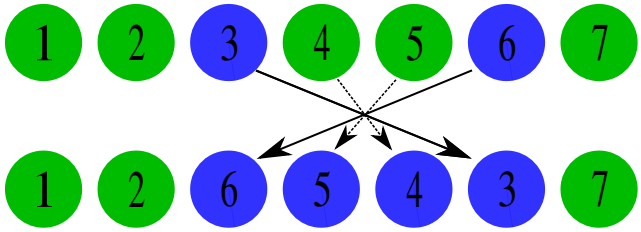


Figure 4.1: In a chromosome, two indices $i = 3, j = 6$ are chosen and the mutation is done around them, creating a new chromosome.

is

$$p_i = (x_{O_i} + \delta_i \cos\theta_i, y_{O_i} + \delta_i \sin\theta_i)\,.$$

According to the authors of [4], the fact, that only the sequence of visits of each target and initial heading at the entry point needs to be determined, the overall complexity of the DTSPN reduces from $4n$ to $2n$, where n is the number of targets to be visited. The authors also mention the fact that in real situations, the data collection is not instant but can take some time. When the visiting point of each region is on the border, the UAV is not guaranteed to spend enough time in the target region. This issue is solved by calculating with neighborhoods where the diameter of the disk area is reduced by a constant.

Another reduction to the computational difficulty of the DTSPN presented in [4] is an approximate gradient-based search. The authors present that if the visiting sequence of two solutions is the same, then the difference is only in visiting points of each region. While changing the visiting point of one target region affects all the other target regions, the farther from the changed point, the effect weakens. Thus to reduce the computational cost, only a part of the solution is adapted when one visiting point is changed.

The evolutionary algorithm has two mutation operators and a crossover. The swapping mutation operator changes the sequence of the visited targets. Randomly choosing two indices $i, j \in 1, 2, \ldots, n, i \neq j$. It reverses the gene in the chromosome in part bordered by the two chosen indices, creating a new solution. The example of the mutation can be seen in Figure 4.1, where indices $i = 3, j = 6$ are chosen and the sequence between them is reversed.

The other mutation operator is shifting the visiting point of the region. Index i is again randomly chosen and the polar angle of the gene is reset within the interval $(0, 2\pi]$. The i-th gene in the chromosome is changed by changing the point of the visit to the region.

The crossover operator creates a child chromosome by combining two parent chromosomes. First, a random sequence is generated. For each position in the chromosome, the sequence determines the parent who will provide the gene to the offspring. The first gene is copied from the chosen parent to the offspring and is no longer considered in either parent. The process is repeated with the remaining sequences until the offspring is complete. In the example in Figure 4.2, the operator is shown, with the chosen sequence of parents (1,2,2,1,2), and the first step is highlighted in red.

The memetic algorithm has been chosen mainly for its ability to converge to the high-quality solutions and when enough computational time is available. On the other hand, its main disadvantage is that the first solution with solution quality competitive to simple heuristics (e.g., such as are SOM-based) needs relatively high computational requirements and cannot be provided quickly. Therefore, we can change that, by initializing the starting generation with a feasible solution generated by a faster method rather than using random values.

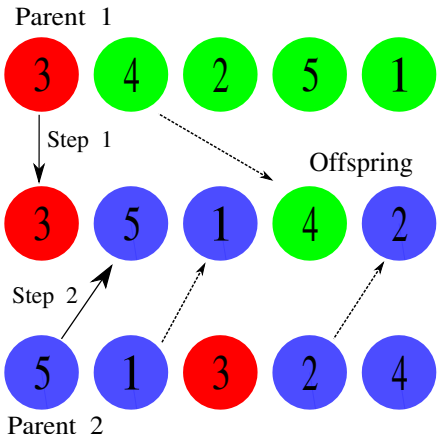


Figure 4.2: Representation of the crossover operator. In step 1, red gene is chosen from parent 1. In step 2, red genes are no longer considered, as they were used already. The first viable gene is chosen from the parent 2 according to the sequence generated at the start of the operator. In the example, (1,2,2,1,2). By repeating the process the offspring is created.

Unsupervised learning algorithm

The second chosen algorithm is the unsupervised learning algorithm based on Self-Organizing Maps (SOM) [3]. This approach uses an artificial neural network to solve the DTSPN. The neurons in the network, which is called a neuron ring, represent the state of the Dubins vehicle. The ring is adapted towards the target locations, and once each target is covered by the path connecting the neurons, the final path can be determined. Thus solving the DTSPN. Regarding the results reported in [3], the SOM-based approach quickly provides high-quality solutions in comparison to existing approaches.

The algorithm implements a method to find the waypoint in the disk area around the target during the selection of the winner neuron, which has been firstly proposed in [14] and then used in [15] and further improved in [16]. In [3], the method has also been used to solve the DTSPN for multiple vehicles. The unsupervised learning method [3] is detailed in the rest of this section.

Self Organizing Maps for the TSP and the DTSP

SOM is a type of artificial neural network using unsupervised learning to adapt its neurons. It can be used for mapping high-dimensional data into a low dimensional grid [17]. Therefore it is a useful tool for data visualization that could not be displayed otherwise. It is also used for clustering data and other classification problems. SOM for such type of a problem is typically a 2D map. However, SOM for the TSP maps the input space and the targets into the neural network with a one-dimensional array of the output units [18]. The neuron weights and the input signals share the same space. Thus, the connected neuron ring represents the path between the target locations. [19].

When using SOM for solving the ETSP, neural network $N = (\nu_1, \ldots, \nu_m)$ is created, where ν_i is a neuron represeting the location of the vehicle in the input space $\mathbb{R}^2$, and m is the number of neurons in the ring. The final solution for the ETSP is found by connecting neurons by straight lines [19]. For the SOM-based solution of the DTSP, the neuron contains

the information about Dubins vehicles heading. Therefore, each neuron ν_i represents the state of the Dubins vehicle $\nu_i \in SE(2)$. The final Dubins path is constructed by connecting the neurons by corresponding Dubins maneuvers. Although the final solutions of the ETSP and the DTSP differ in constructing the final path and in what the neurons represent, the same learning framework can be utilized to find the final state of the network. To complete the description, the framework is sourced directly from [3].

1. *Initialization*: For n target locations O, create a ring of neurons with randomly initialized weights, e.g., with $2n$ neurons [15]. Initialize the learning parameters as follows: the learning gain $\sigma = 10$, the learning rate $\mu = 0.6$, the gain decreasing rate $\alpha = 0.1$, and set the learning epoch counter $i = 1$.

2. *Randomizing*: Create a random permutation of locations $\Pi(O)$ to avoid local minima.

3. *Learning epoch*: For each $o \in \Pi(O)$

   (a) *Select winner* neuron v^* for o as the best matching neuron i.e., the closest neuron to o.

   (b) *Adapt* v^* and its neighbors to o using the neighbouring function (4.1), i.e., set the neuron weights to the new locations v determined as:

$$v' = v + \mu f(\sigma, d)(o - v)$$

4. *Update learning parameters*: $\sigma = \sigma(1 - i\alpha), i = i + 1$

5. *Termination condition*: If solution is not improving or $i \geq i_{max}$ Stop the adaptation. Otherwise go to step 2.

6. *Construct the final (Dubins) tour using the last winners*

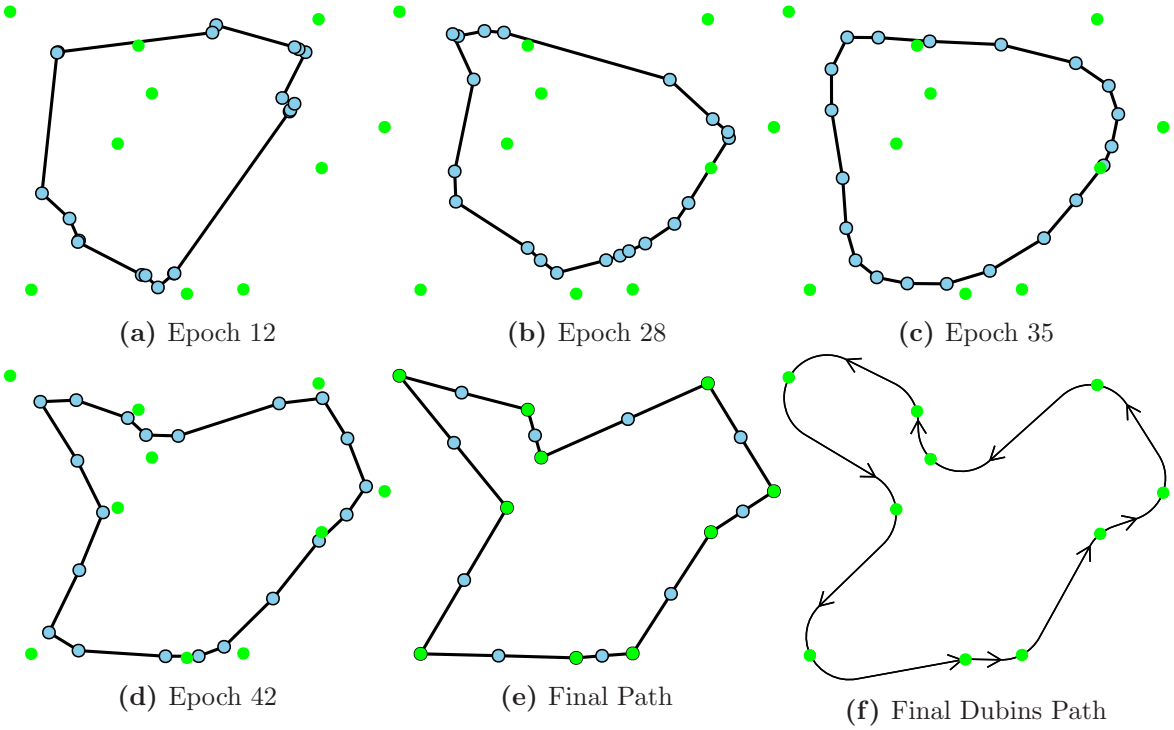
The used neighbouring function follows the existing SOM for the TSP [19] and it has the form

$$f(\sigma, d) = \begin{cases} e^{\frac{-d^2}{\sigma^2}} \text{ for } d < 0.2m^r \\ 0 \text{ otherwise} \end{cases}, \tag{4.1}$$

which decreases the power of adaptation of the neighbouring nodes to the winner neuron ν^* with increasing distance d of the neuron to ν^* counted in the number of neurons in the ring. The adaptation can be viewed as a movement of the neurons to new position ν' which replaces the neuron weights ν. Figure 4.3 shows the evolution of the ring of neurons towards the target locations.

SOM for the DTSPN

In solution of the DTSPN, the neurons in the ring represent the waypoints on Dubins path. Unlike in the SOM-based solution of the TSP, to connect the neurons with Dubins maneuvers into the resulting path, each neuron needs to keep the expected headings [20]. The headings have a major effect on the resulting path after connecting the neurons. Thus each neuron $\nu_i \in N$ holds a set of headings $\Theta_i = \{\theta_i^{-k}, \theta_i^{-k+1}, \ldots, \theta_i^2, \theta_i^k\}$. To find the optimal solution, the heading that matches the shortest possible path needs to be determined. To acquire this heading, the neuron ring can be treated as a search graph, where the headings of the



(a) Epoch 12 **(b)** Epoch 28 **(c)** Epoch 35

(d) Epoch 42 **(e)** Final Path **(f)** Final Dubins Path

Figure 4.3: Example of the ring evolution towards the targets. After the adaptation process is finished the headings are determined, and the final Dubins path is constructed [3]

neighboring neurons are connected. The heading can be determined from the graph by a feedforward search with the computational complexity bounded by $O\left(ms^3\right)$, where m is the current number of neurons in the ring and s is the number of headings that each neuron keeps in its structure. The search graph representation is shown in Figure 4.4.

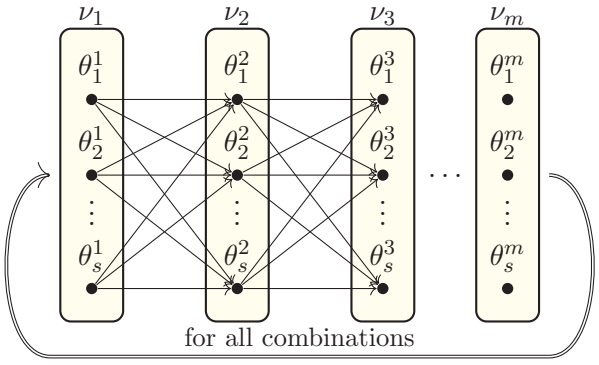


Figure 4.4: A search graph showing how the headings are connected in the neuron ring, source: [3]

The path aquired after this procedure is then used when searching for the winner neuron ν^*. The procedure of adapting the network towards a target O used in the algorithm starts by finding the closest point of the current path p_O to the target O. If there is no neuron, within location of the point, new neuron is created and its state is set to the location. The vehicle heading θ_p from the point p_O is set as the main heading for the new neuron and the other headings are set around θ_p as $\Theta_{\nu^*} = \{\theta_p, \theta_p^1, \ldots, \theta_p^i, \theta_p^{-1}, \ldots, \theta_p^{-i}\}$, where $\theta_p^i = \theta_p + i\pi/l, 1 \leq i \leq l$ and l is set to $l = 12$. The process is shown in Figure 4.5. Next, the point O_p is found on

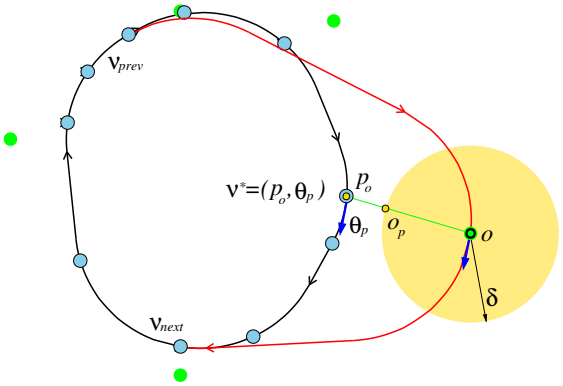


Figure 4.5: Graphic showing the winner selection procedure and the point o_p towards which the network is adapted [3]

the bound of the region of target O as the point on the line connecting point p_O and target O. The point O_p is at the distance equal to the radius δ from O. Then, the network adapts towards O_p rather than to O to save the travel distance. In a case that the winner neuron is already within the sensing radius δ from O, the network is not adapted at all, as the target can already be covered from p_O. After the adaptation of the winner neuron, the neighboring neurons are adapted as well. The neighborhood is defined by two neurons ν_{prev} and ν_{next}. These neurons are determined to minimize the expected Dubins path towards O. According to the equation

$$L_g = l\left(\nu_{prev}, (O, \theta)\right) + l\left((O, \theta), \nu_{next}\right), \tag{4.2}$$

The neurons ν_{prev} and ν_{next} are determined from equation 4.1. Where θ is one of the headings of the winner neuron ν^*. ν_{prev} and ν_{next} are from the activation bubble around ν^*. For the neuron to belong into the activation bubble, its neighbouring function needs to be above the activation limit, which is set to 10^{-5}. As the neighbouring function depends on the correct value of the learning gain σ, which decreases during the learning, the neurons ν_{prev} and ν_{next} may not be found. In that case, only the winner neuron ν^* is adapted towards O_p. Otherwise all neurons between ν_{prev} and ν_{next} are adapted towards O_p including the winner neuron. The adaptation is made so that the winner neuron ν^* is moved to the location of O_p and then Dubins maneuvers are determined between ν_{prev}, ν^* and ν_{next}. If a new neuron has been added during the winner selection, one neuron between ν_{prev} and ν_{next} is removed, if such a neuron exists.

The Unsupervised learning algorithm has been chosen for its ability to provide a competitive solution quickly. However, its main disadvantage is that once it converges to the solution, the solution is not improved with more computational time.

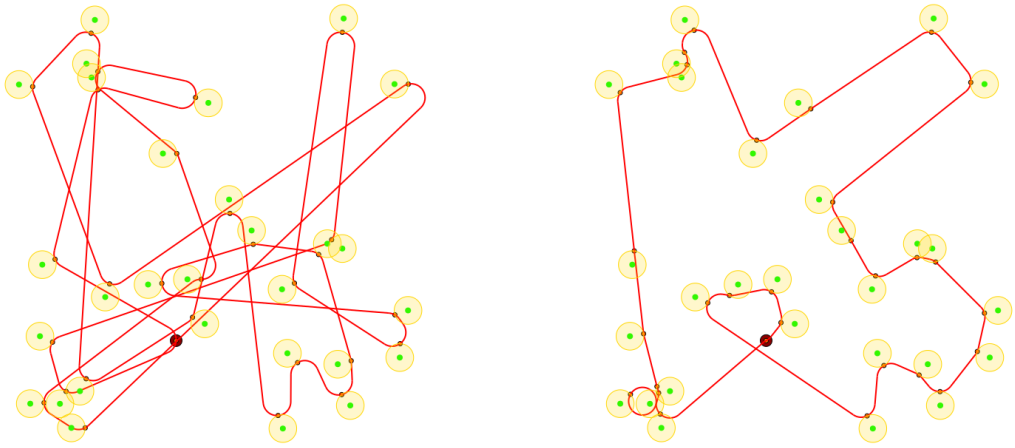
CHAPTER 5

Proposed Hybrid Algorithm

The proposed Hybrid algorithm combines the advantages of the two studied algorithms, SOM-based and memetic. The unsupervised learning algorithm [3] can quickly provide a high-quality solution. However, it does not further improve the solution, even when additional computational time is available. The Memetic algorithm [4] is able to provide high-quality solutions for the given problem. In a case, the local optimum has been reached. The Memetic algorithm can leave the local optimum and improve the solution further if enough computational time is given. This makes it convenient for a combination with the unsupervised learning algorithm. Because the solution quickly provided by the SOM can be further improved.

The Hybrid algorithm generates the initial population by the SOM [3]. The SOM generates high-quality solutions in a reasonably short time. Therefore the first generation of the Memetic algorithm is filled with the solutions provided by the SOM. The solutions are improved using mutation and crossbreeding. To improve the diversity in the population, the Hybrid algorithm alternates solutions generated by the SOM with random valid solutions. In Figure 5.1, two valid solutions to a problem are shown. One is a randomly generated solution, which was the best solution in the initial population of the Memetic algorithm. The other is a solution, that was the best for the initial population of the Hybrid algorithm for the same instance. The picture shows that the initial population has much better quality solutions than the initial population in the original Memetic algorithm. The different solutions generated by the SOM take turns in populating the initial population while alternating with random solutions to get higher diversity in the initial population.

The main issue of using the SOM generated solutions as the first-generation solutions for the Memetic algorithm is the different encoding for the path in both algorithms. The Hybrid approach needs to transform the SOM generated solution into the encoding of the Memetic algorithm. The SOM uses the neurons as its waypoints. In a valid solution, each target needs to be covered by one or more neurons from the final neural network. From the neuron point of view, each neuron can cover one or more targets in the final network. The neurons are connected to the final path. Each neuron has the link to the next neuron in the ring, which determines the order in which the targets are visited. Each neuron knows its location and the direction in which Dubins vehicle will be heading when leaving the point represented by the location of the neuron. The Memetic algorithm uses a system where one solution is represented by the targets. To determine the order of the visits, each target remembers its



(a) Initial solution by Memetic [4]

(b) Initial solution by the proposed Hybrid

Figure 5.1: Ilustration of the best solutions provided by the Memetic algorithm and the Hybrid algorithm in their initial population. This example has the basic setting from which all other experiments are made. That is $\delta = 4$, $\rho = 4$, with 30 targets.

number in the sequence. For each target, one point of its neighborhood is chosen as the point of the visit. This point is always at the border of the neighborhood, and the heading of the vehicle is also remembered.

The function transforming the representation of a solution from the SOM to the Memetic needs to address two main problems. The first is the fact that the number of neurons can be different from the number of actual targets that need to be visited. To determine the sequence of the neurons which will be used in the new solution, the function iterates through all the neurons, and with each neuron, it searches for all the targets covered by the neuron. Labeling the target as covered. Only uncovered targets are considered by the rest of the neurons. Therefore each target has exactly one neuron chosen to cover it. In this sequence of neurons, each neurons heading and the location is stored. The sequence determines the order of the locations. The second issue that needs to be addressed is that the representation of the waypoints in the Memetic algorithms is always at the border of the neighborhood for each target. The location of the neurons is not always on the boundary of the neighborhood. The location of the neuron is moved to the boundary of the circular neighborhood. It is moved on the line defined by the target location and the position of the neuron, to the point where the line crosses the boundary of the neighborhood. The change of the visiting point in the neighborhood is portrayed in Figure 5.2. The original visiting point is the location of the neuron ν_i. The converting function moves the visiting location to point p_i to the boundary. The change affects the final Dubins path through the neighborhood of the target O_i.

This adjustment means that the solution used in the first generation of the Memetic part of the Hybrid may not be identical to the solution provided by the SOM itself. By moving the location, additional curves might arise, prolonging the initial length of the solution. To be able to transform the solution, the points in which the Dubins path enters the neighborhoods would have to be found. This would require additional computational time. However, the memetic algorithm itself addresses this issue in the first new generation, due to the approximate-gradient search optimization described in [4]. The pseudocode of the transformation function

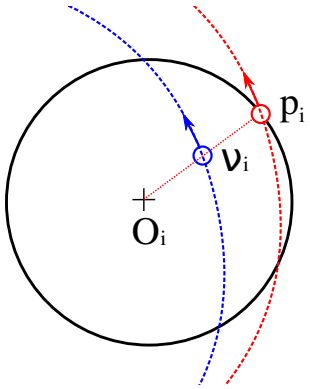


Figure 5.2: The visiting point of the target O_i is moved while converting the solution from one representation to the other. The original location of the visiting point is the neuron ν_i. The converting function moves the visiting point to the point p_i on the boundary of the neighborhood in the direction of the heading, resulting in the change of Dubins path.

can be seen in Algorithm 1

Algorithm 1: The function transforming the representation of the solution from the SOM alghorithm into the Memetic

Data: N^* - ring of winner neurons, set of target locations O, C empty array of the covered targets
Result: f - the solution for the Memetic algorithm

```
foreach ν_i ∈ N* do
    foreach o ∈ ∏(O) do
        if |ν_i, o| < δ ∩ o ∉ C then
            C ← o;
            f ← ν_i;
            // Add the location and heading of the neuron to the solution
        end
    end
end
foreach f_i ∈ f do
    f_i ← moveToEdge(f_i);
end
```

The Hybrid algorithm runs the SOM k times and stores all the solutions. The memetic part is then initialized. The initial population has size of $500 + 20n$, where n is the number of targets, which is suggested in [4]. Filling this population, the k SOM generated solutions take turns, and the pattern is always one SOM solution and one random solution. From this point, the Memetic part of the Hybrid algorithm is ran until no more computational time is left. Then the best found solution is returned. The pseudocode for the complete Hybrid algorithm can be seen in Algorithm 2.

Algorithm 2: The pseudocode of the Hybrid algorithm describing also the Memetic part presented in [4]

Data: t_{max} - The time limit for the algorithm, p_m - Probability of the mutation, k - number of SOM initializations, n_F - number of solutions in one generation of the Memetic algorithm

Result: f_{best} - Final solution

```
for i ∈ k do
    N*_i ← SOM();
    f_som ← convert(N*_i, O);
    if |f_som| < |f_best| then
        f_best ← f_som;
    end
end
for i ∈ n_F do
    f_i ← f_som ∨ randomSolution();
    // SOM solutions take turns and alternate with random solutions
    F ← f_i;
end
while t < t_max do
    for i ∈ n_F do
        f_i = tournament(F);
        f_j = tournament(F);
        f ← crossover(f_i,f_j);
        m = rnd();
        if m < p_m then
            f ← mutate(f);
        end
        F' ← f;
        if |f| < |f_best| then
            f_best ← f;
        end
    end
    F ← F';
end
```

CHAPTER 6

Results

In this chapter, we report on the performance of the proposed Hybrid algorithm. The Hybrid algorithm is a combination of the Unsupervised learning algorithm and the Memetic algorithm. Therefore in the series of conducted tests, it has been compared to the two original algorithms. The Hybrid solution aims to provide competitive solutions quicker than the Memetic algorithm and improve them with more computational time, unlike the SOM-based algorithm.

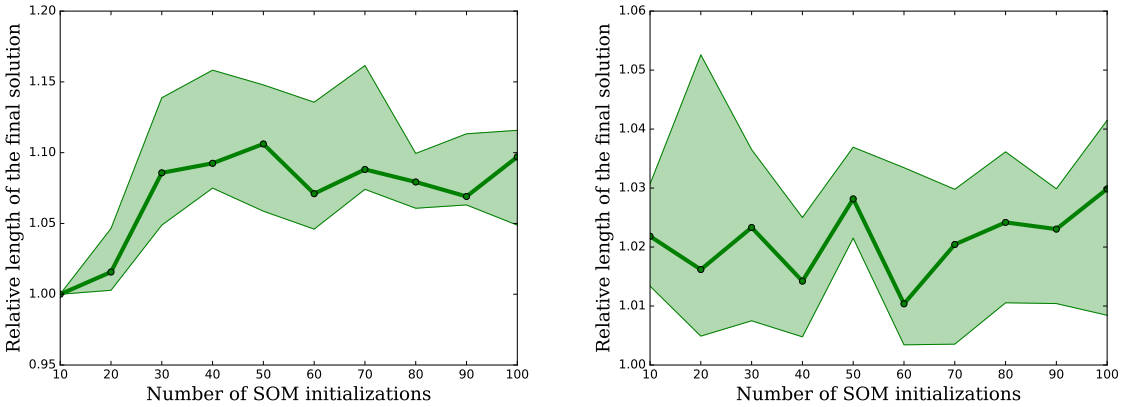
Each testing problem has been randomly generated. The parameters of the instances are the number of targets n, sensing radius of each target δ, representing the size of the neighborhood. Turning radius of the Dubins vehicle ρ and the density of the targets D. That is computed as the dependency of the size of the square area S on n.

$$D = \frac{S}{n}$$

The basic settings chosen for the experiments are $n = 30, D = 20, \delta = 4, \rho = 4$. Afterwards, one parameter is changed and 10 instances with different values of the parameter are tested with each algorithm. In each setting, 10 different instances have been generated and tested. All of the tests have been conducted on a server cluster with 20 CPU nodes, each having 24 cores/48 threads 3.2GHz (2 x Intel Xeon 6146) with 384GB RAM. Each algorithm has been run on every individual problem instance 10 times. All the tests were run with maximum provided time set to 30 minutes. The results are shown in relative length to the best found solution for each instance in each setting. A non-parametric confidence interval of 60% is shown, and the median is highlighted for each algorithm.

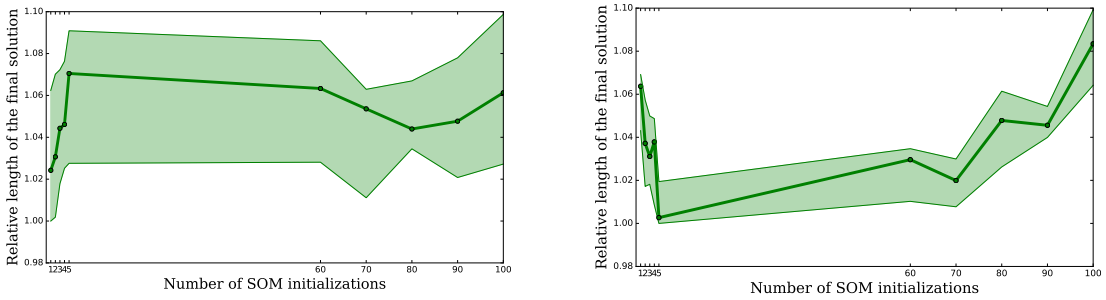
SOM initialization count for the Hybrid algorithm

The decision needed to be made about how many times the SOM should be run for the initialization of the proposed Hybrid algorithm. The test had been conducted on 10 problem instances with $n = 100, D = 20, \delta = 4, \rho = 4$. Figure 6.1 shows that after 30 minutes of computational time, the differences between settings are within a 2% interval, while with only 2 minutes of computational time, the best results are provided by the Hybrid initialized with least amount of different runs of the SOM. Another test is shown in Figure 6.2, where the Hybrid algorithm has been run with small and large numbers of different initializations for



(a) Presented are results found in 2 minutes **(b)** Presented are results found in 30 minutes

Figure 6.1: Test of the Hyrid algorithm with different number of SOM initializations, conducted on 10 intsances, with settings $n = 100, D = 20, \delta = 4, \rho = 4$.



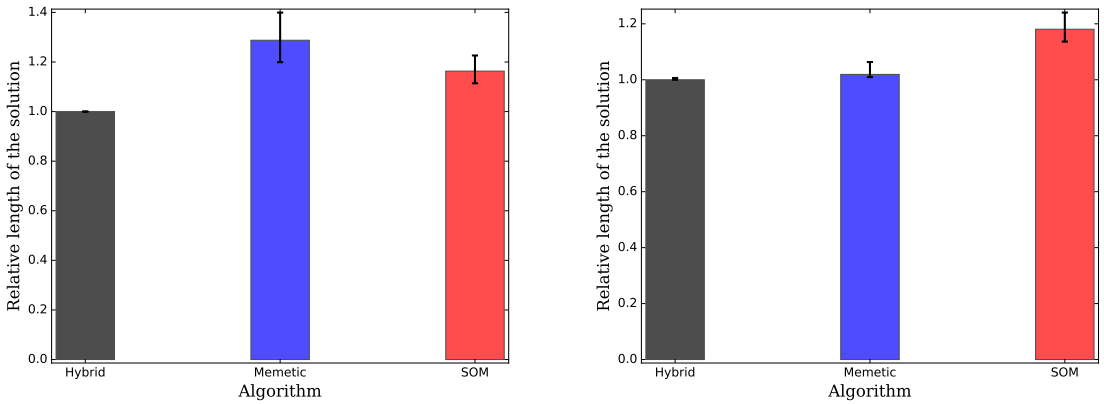
(a) Presented are results found in 10 seconds. **(b)** Presented are results found in 10 minutes.

Figure 6.2: Test of the Hyrid algorithm with few and many SOM initializations, conducted on 10 instances, with settings $n = 100, D = 20, \delta = 4, \rho = 4$.

10 minutes, and once again, it shows that the Hybrid with a smaller number of initializations does not have worse performance than with a high number of initializations. The results of the second test after 10 seconds are also shown, where in short amounts of time, running the SOM fewer times means that the Hybrid gets to the Memetic part sooner and can start improving provided solutions. With longer computational times, the Memetic algorithm can provide high-quality solutions. Based on the performance evaluation, the count of initializations chosen for the Hybrid algorithm is $k = 3$ for the rest of this thesis.

Basic settings

For the basic settings $n = 30, D = 20, \delta = 4, \rho = 4$ the results for each algorithm are shown after 10 seconds and after 50 seconds in Figure 6.3. The Hybrid algorithm has the best results in both scenarios. The memetic algortihm provides solutions of competitive quality to the Hybrid algorithm after 50 seconds, but not as reliably.

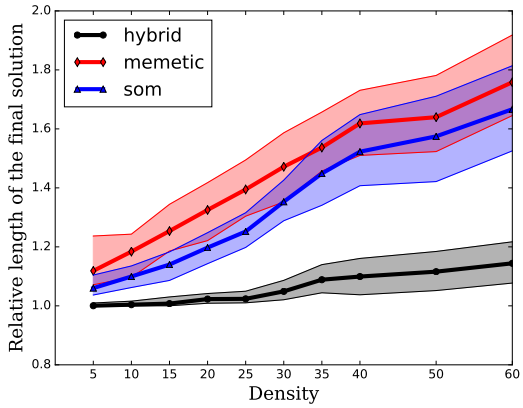


(a) Presented are solutions found in 10 seconds **(b)** Presented are solutions found in 50 seconds

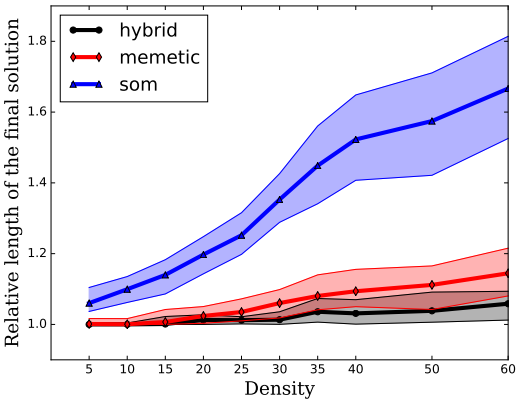
Figure 6.3: Test conducted on 10 maps with basic settings $n = 30, D = 20, \delta = 4, \rho = 4$.

Changing the density of the map

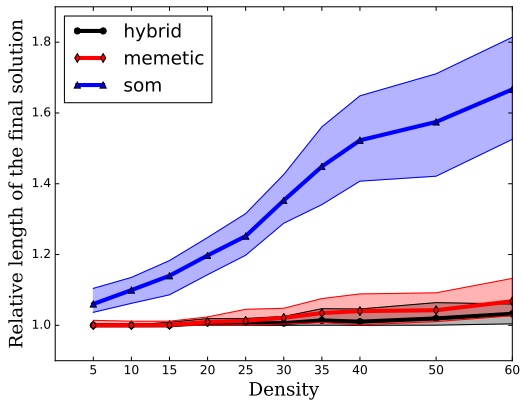
The settings for this test are $n = 30, D = 20, \rho = 4$ and the δ changes from 1 to 10. The results for each algorithm are shown. In Figure 6.4.



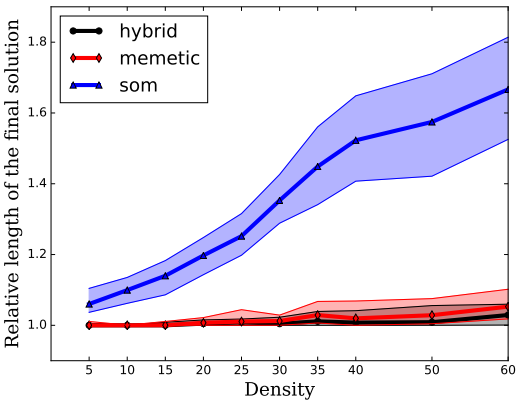
(a) Presented are results found in 10 seconds



(b) Presented are results found in 2 minutes



(c) Presented are results found in 15 minutes

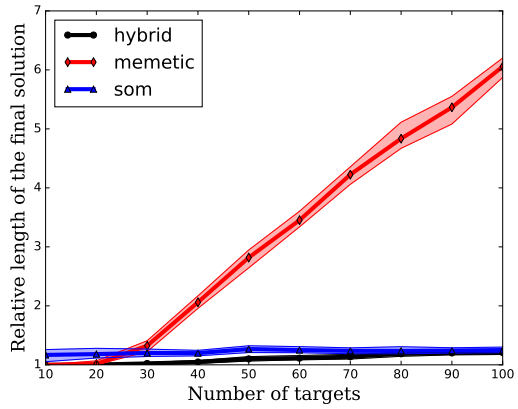


(d) Presented are results found in 30 minutes

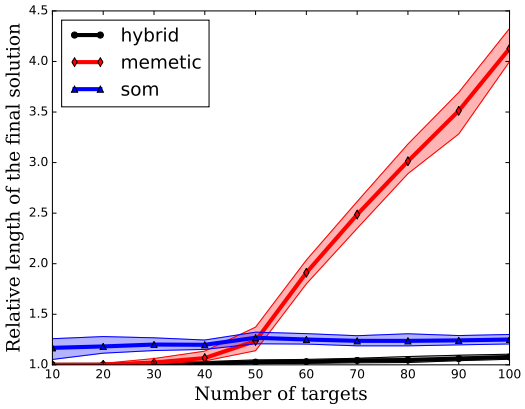
Figure 6.4: Test conducted on 100 instances with settings $n = 30, \delta = 4, \rho = 4$ for 10 different values of D.

Changing the number of targets on the map

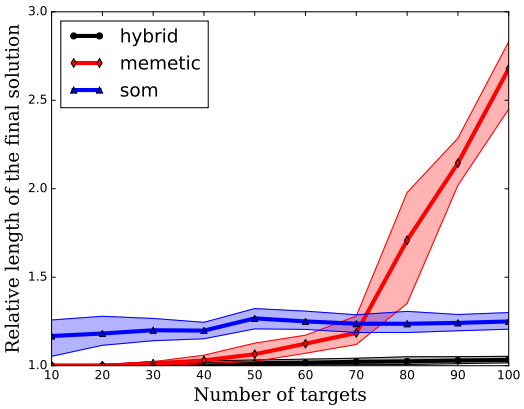
The settings for this test are $D = 20, \rho = 4, \delta = 4$ and the n changes by 10 from 10 to 100. The results for each algorithm are shown. In Figure 6.5.



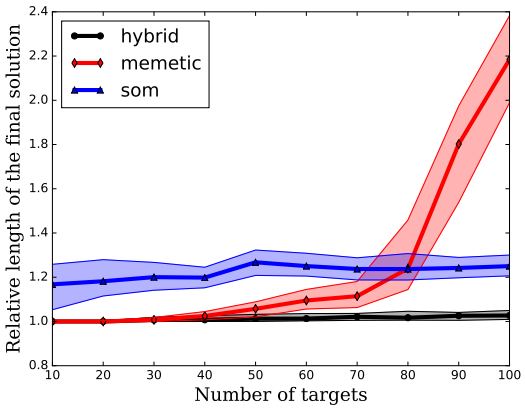
(a) Presented are results found in 10 seconds



(b) Presented are results found in 2 minutes



(c) Presented are results found in 15 minutes

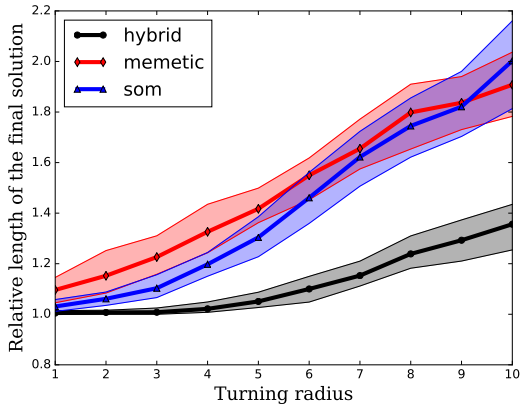


(d) Presented are results found in 30 minutes

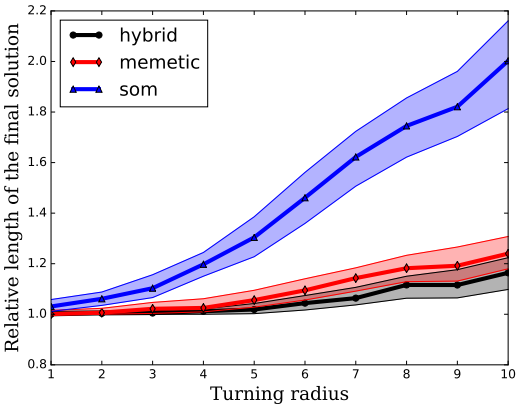
Figure 6.5: Test conducted on 100 instances with settings $D = 20, \delta = 4, \rho = 4$ for 10 different values of n.

Changing the turning radius

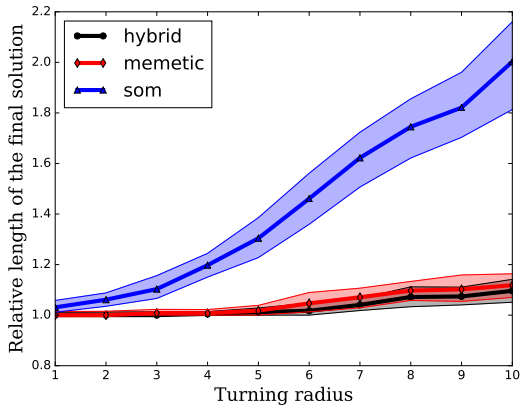
The settings for this test are $n = 30, D = 20, \delta = 4$ and the ρ changes from 1 to 10. The results for each algorithm are shown. In Figure 6.6.



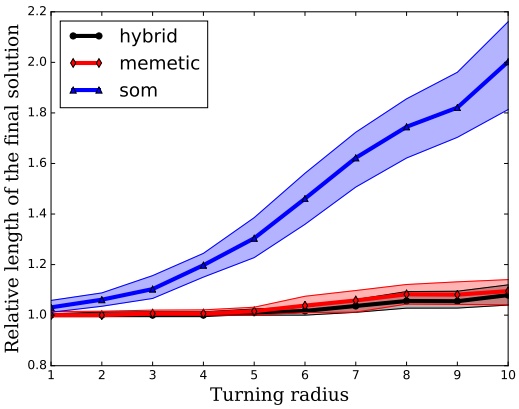
(a) Presented are results found in 10 seconds



(b) Presented are results found in 2 minutes



(c) Presented are results found in 15 minutes

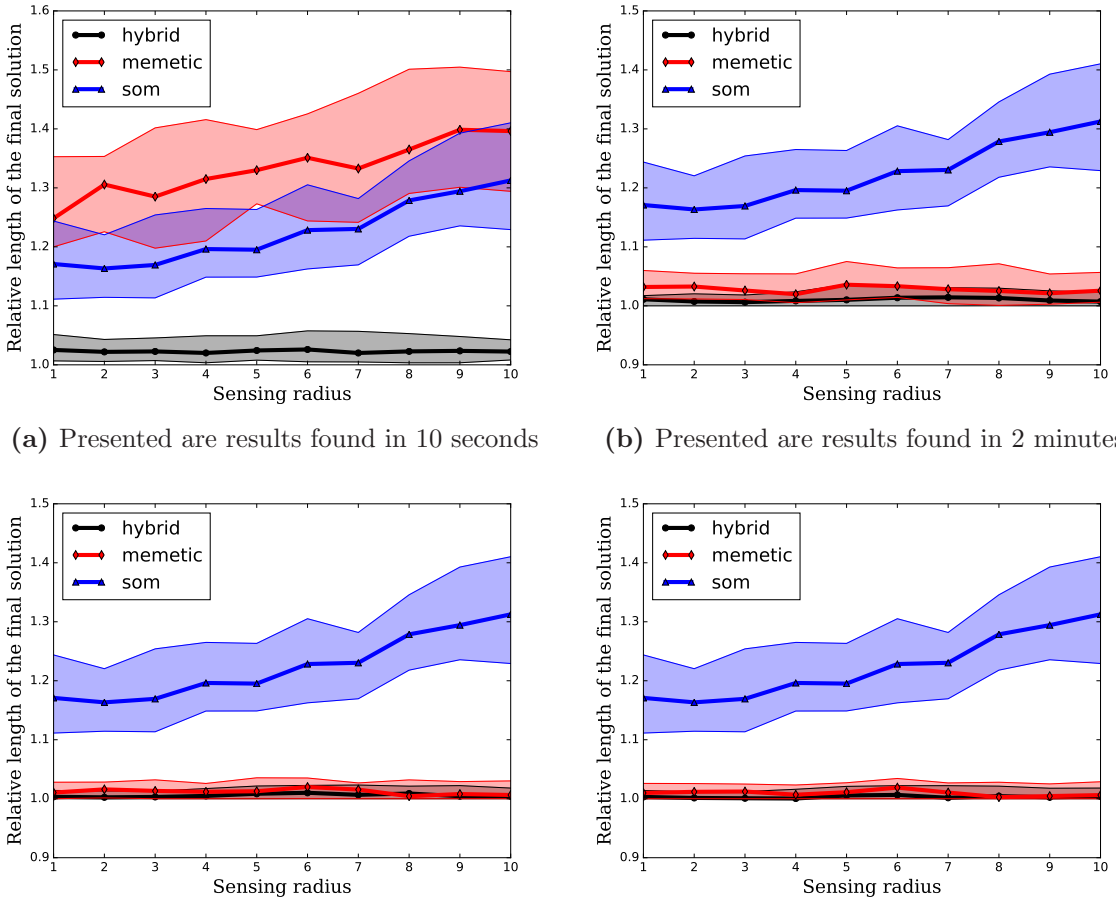


(d) Presented are results found in 30 minutes

Figure 6.6: Test conducted on 100 instances with settings $n = 30, D = 20, \delta = 4$ for 10 different values of ρ.

Changing the sensing radius

The settings for this test are $n = 30, D = 20, \rho = 4$ and the δ changes from 1 to 10. The results for each algorithm are shown. In Figure 6.7.



(a) Presented are results found in 10 seconds

(b) Presented are results found in 2 minutes

(c) Presented are results found in 15 minutes

(d) Presented are results found in 30 minutes

Figure 6.7: Test conducted on 100 instances with settings $n = 30, D = 20, \rho = 4$ for 10 different values of δ.

Discusion

The results suggest that the Hybrid algorithm can provide higher quality solutions faster than the Memetic algorithm. The Hybrid is then able to improve upon the solutions with more computational time. The results provided by the Hybrid algorithm in these tests are of a higher quality than the results provided by the SOM-based algorithm even in short amounts of time. With long computational times, The Memetic algorithm can provide solutions of simillar quality to those provided by the Hybrid algorithm. The most significant difference is made by adding more targets to the instance, where the Memetic algorithm can need a lot of resources before being able to provide competitive solution.

CHAPTER 7

Conclusion

The thesis focuses on solutions to the Dubins Traveling Salesman Problem with Neighborhoods (DTSPN). The motivation for solving the DTSPN are surveillance missions performed by the Unmanned Aerial Vehicles (UAV). First, the Traveling Salesman Problem (TSP) and the Traveling Salesman Problem with Neighborhoods (TSPN) are presented. These are problems of connecting locations of interest by a path visiting all of the locations minimizing the traveled distance. However, the model for vehicle constrained by its turning radius is needed in planning surveillance missions with the UAVs. Dubins vehicle [8] is used. When searching for a curvature constrained path connecting all locations of interest, the problem can be addressed as the Dubins Touring Problem (DTP). After presenting these prerequisites, the thesis presents the Dubins Traveling Salesman Problem (DTSP) and Dubins Traveling Salesman with Neighborhoods (DTSPN) to the reader.

The main contribution of this thesis is a novel Hybrid algorithm for solving the DTSPN. The algorithm is a combination of the unsupervised learning algorithm based on Self Organizing Map (SOM) [3] and the Memetic algorithm [4]. Both these algorithms are presented to the reader as they are essential for the proposed Hybrid algorithm. The SOM algorithm can provide a competitive solution quickly but is not able to improve the solution with more computational time. The Memetic algorithm is more demanding to provide the first competitive solution. On the other hand, it can converge to high-quality solutions. The Hybrid algorithm combines the advantages of both algorithms by initializing the first generation of the memetic approach by quality solutions provided by SOM, while also being able to improve the solution over time.

The results suggest that the proposed Hybrid algorithm has the best performance with fewer different SOM solutions in the initial population, as there were no significant improvements in solution quality with high numbers of different SOM solutions. The proposed Hybrid algorithm can provide high-quality solutions within short periods of time and is able to improve the solutions with more resources at its disposal.

Bibliography

[1] https://newatlas.com/carbonix-volanti-vtol-fixed-wing-industrial-uav/48253. Accesed: 17.5.2019.

[2] J. Faigl, P. Váňa, M. Saska, T. Báča, and V. Spurný, "On solution of the dubins touring problem," in *2017 European Conference on Mobile Robots (ECMR)*, pp. 1–6, Sept 2017.

[3] J. Faigl and P. Váňa, "Unsupervised learning for surveillance planning with team of aerial vehicles," in *2017 International Joint Conference on Neural Networks (IJCNN)*, pp. 4340–4347, May 2017.

[4] B. X. X. Zhang, J. Chen and Z. Peng, "A memetic algorithm for path planning of curvature-constrained uavs performing surveillance of multiple ground targets," *Chinese Journal of Aeronautics*, vol. 27, no. 3, pp. 622–633, 2014.

[5] V. C. D. Applegate, R. Bixby and W. Cook, *The Traveling Salesman Problem: A Computational Study*. Princeton University Press, 2007.

[6] S. Gupta and P. Panwar, "Solving travelling salesman problem using genetic algorithm," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 3, pp. 376–380, 01 2013.

[7] S. Saud, H. Kodaz, and İ. Babaoğlu, "Solving travelling salesman problem by using optimization algorithms," *KnE Social Sciences*, vol. 3, no. 1, pp. 17–32, 2018.

[8] L. E. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *American Journal of Mathematics*, vol. 79, no. 3, pp. 497–516, 1957.

[9] J. Ny, E. Feron, and E. Frazzoli, "On the dubins traveling salesman problem," *IEEE Transactions on Automatic Control*, vol. 57, pp. 265–270, Jan 2012.

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

[11] K. Savla, E. Frazzoli, and F. Bullo, "On the point-to-point and traveling salesperson problems for dubins' vehicle," in *Proceedings of the 2005, American Control Conference, 2005.*, pp. 786–791 vol. 2, June 2005.

[12] R. Pěnička, J. Faigl, P. Váňa, and M. Saska, "Dubins orienteering problem," *IEEE Robotics and Automation Letters*, vol. 2, pp. 1210–1217, April 2017.

[13] J. Isaacs and J. Hespanha, "Dubins traveling salesman problem with neighborhoods: A graph-based approach," *Algorithms*, vol. 6, pp. 84–99, 02 2013.

[14] J. Faigl, "Approximate solution of the multiple watchman routes problem with restricted visibility range," *IEEE Transactions on Neural Networks*, vol. 21, pp. 1668–1679, Oct 2010.

[15] J. Faigl and L. Přeučil, "Self-organizing map for the multi-goal path planning with polygonal goals," in *Artificial Neural Networks and Machine Learning – ICANN 2011* (T. Honkela, W. Duch, M. Girolami, and S. Kaski, eds.), (Berlin, Heidelberg), pp. 85–92, Springer Berlin Heidelberg, 2011.

[16] J. Faigl and G. A. Hollinger, "Unifying multi-goal path planning for autonomous data collection," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2937–2942, Sep. 2014.

[17] T. Kohonen, "The self-organizing map (som)," 2001.

[18] B. Angéniol, G. de La Croix Vaubois, and J.-Y. L. Texier, "Self-organizing feature maps and the travelling salesman problem," *Neural Networks*, vol. 1, no. 4, pp. 289 – 293, 1988.

[19] E. Cochrane and J. Beasley, "The co-adaptive neural network approach to the euclidean travelling salesman problem," *Neural Networks*, vol. 16, no. 10, pp. 1499 – 1525, 2003.

[20] J. Faigl and P. Váňa, "Self-organizing map for the curvature-constrained traveling salesman problem," in *Artificial Neural Networks and Machine Learning – ICANN 2016* (A. E. Villa, P. Masulli, and A. J. Pons Rivero, eds.), (Cham), pp. 497–505, Springer International Publishing, 2016.