CZECH TECHNICAL UNIVERSITY IN PRAGUE

Faculty of Electrical Engineering

BACHELOR'S THESIS

David Zahrádka

Motion planning for team of Unmanned Aerial Vehicles with flight time constraint and Dubins vehicle model

Department of Cybernetics

Thesis supervisor: **Ing. Robert Pěnička**

Prohlášení autora práce

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 dne 25.5.2018

[illegible]

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name:	**Zahrádka David**	Personal ID number:	**434763**
Faculty / Institute:	**Faculty of Electrical Engineering**		
Department / Institute:	**Department of Control Engineering**		
Study program:	**Cybernetics and Robotics**		
Branch of study:	**Systems and Control**		

II. Bachelor’s thesis details

Bachelor’s thesis title in English:

Motion planning for team of Unmanned Aerial Vehicles with flight time constraint and Dubins vehicle model

Bachelor’s thesis title in Czech:

Plánování pohybu týmu bezpilotních prostředků s omezením délky letu a použitím modelu Dubinsova vozítka

Guidelines:

1. Get familiar with motion planning for team of robots over multiple target locations with route length constraint (i.e. Team Orienteering Problem) [2] for information gathering tasks.
2. Design selected method for Team Orienteering Problem [1,2] with Euclidean norm distances between target locations.
3. Extend designed method for motion planning with Dubins vehicle model [3] suitable for Unmanned Aerial Vehicles.
4. The deigned method should consider information gathering within circular neighborhood of given target locations [4].

Bibliography / sources:

[1] Pieter Vansteenwegen, Wouter Souffriau, Dirk Van Oudheusden, 'The orienteering problem: A survey,' in European Journal of Operational Research, vol. 209, pp. 1-10, 2011.
[2] I-Ming Chao, Bruce L. Golden, Edward A. Wasil, 'The team orienteering problem,' in European Journal of Operational Research, vol. 88, pp. 464-474, 1996.
[3] R. Pěnička, J. Faigl, P. Váňa and M. Saska, 'Dubins Orienteering Problem,' in IEEE Robotics and Automation Letters, vol. 2, no. 2, pp. 1210-1217, 2017.
[4] J. Faigl and R. Pěnička, 'On close enough orienteering problem with Dubins vehicle,' in IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, Canada, pp. 5646-5652, 2017.

Name and workplace of bachelor’s thesis supervisor:

Ing. Robert Pěnička, Multi-robot Systems, FEE

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

Date of bachelor’s thesis assignment: **16.01.2018** Deadline for bachelor thesis submission: **25.05.2018**

Assignment valid until: **30.09.2019**

Ing. Robert Pěnička	prof. Ing. Michael Šebek, DrSc.	prof. Ing. Pavel Ripka, CSc.
Supervisor’s signature	Head of department’s signature	Dean’s signature

© ČVUT v Praze, Design: ČVUT v Praze, VIC

Acknowledgements

I would like to thank my advisor for his invaluable help and guidance, my family and friends for their never-ending support, the legendary emperor Shennong for the invention of tea and Hedwig Norris for her patient assistance during my long debugging sessions.

Abstract

This thesis deals with Dubins Team Orienteering Problem with Neighbourhoods, a novel routing problem formulation. The goal is to collect a reward by visiting locations' close vicinity using multiple curvature-constrained vehicles, each limited by specified travel budget. This problem formulation is useful in data-collection scenarios with fixed-wing or multirotor UAVs with constant forward speed, where remote data collection is possible and where it is necessary to consider the limited amount of fuel or battery capacity. A Greedy Randomised Adaptive Search Procedure with Path Relinking is presented as a solution to the Dubins Team Orienteering Problem with Neighbourhoods. The algorithm was tested in a real-life experiment scenario using multiple Micro Aerial Vehicles.

Abstrakt

Tato práce se zabývá optimalizačním problémem Dubins Team Orienteering Problem with Neighbourhoods. Jeho cílem je maximalizovat odměnu získávanou navštěvováním blízkého okolí zadaných lokací s předem stanovenou odměnou pomocí několika neholonomních vozítek. Praktické využití této optimalizační úlohy může být například ve scénářích s dálkovým sběrem sensorických dat pomocí skupiny bezpilotních letounů s jistým minimálním poloměrem zatáčení a omezeným palivem či kapacitou baterií. V této práci je jakožto řešení Dubins Team Orienteering Problem with Neighbourhoods představena heuristika Greedy Randomised Adaptive Search Procedure with Path Relinking. Výsledky byly otestovány experimentem v reálných podmínkách se skupinou bezpilotních multikopter.

Contents

List of Figures

1 Introduction

As the usage of autonomous vehicles in the industry increases, the demand for effective control of autonomous vehicles rises proportionally. This includes route planning algorithms, which can increase the productivity of autonomous vehicle deployment when effective. Since optimisation problems are computationally complex, various algorithms and heuristics surface and compete in which is closer to an optimal solution and in the economical usage of execution time. One of the problem formulations used for route planning optimisation is the Orienteering Problem (OP), and its generalisations.

The OP features a set of locations each with their assigned score. The starting point and the ending point are known and fixed. The travel cost needed to travel from location i to location j is known for all location pairs. The travel budget used to visit locations is limited by a given parameter. The goal of the OP is to find a single route, limited by the travel budget parameter, which maximises the total collected score while visiting each location at most once.

The OP can be described as a variant of the Travelling Salesman Problem (TSP). However, compared to the TSP, in which the goal is to minimise the travel distance while visiting all locations, in the OP the goal is to maximise the score collected by visiting nodes within a limited travel budget. The OP is thus a better way to model situations where visiting all locations is impossible, for example in applications where the vehicles used have a limited amount of fuel.

The Team Orienteering Problem (TOP) is an OP that maximises the total score of multiple routes, each limited by a travel budget. Since the travel budget applies to each route individually, this allows the total reward collected by all routes to be higher than it would be possible for only one route. This is very useful in situations where it is possible to deploy multiple UAVs simultaneously.

The Dubins Orienteering Problem (DOP) is an OP that considers only curvature-constrained vehicles called Dubins vehicles. This allows us to plan routes for, for example, unmanned planes, or certain Unmanned Ground Vehicles [UGV] with limited turning radius.

The Orienteering Problem with Neighbourhoods (OPN) or Close Enough Orienteering Problem is another variation of the OP. In the OPN, the reward can be collected by visiting a neighbourhood around each location instead of each location's exact coordinates. This allows increasing the collected reward in situations where it is sufficient to visit the location's neighbourhood defined by a specific neighbourhood radius.

The Dubins Team Orienteering Problem with Neighbourhoods (DTOPN) is a novel problem formulation and an algorithm that solves it is the subject of this thesis. It can be described as a combination of TOP, DOP and OPN, where the goal is to maximise the collected score of multiple routes by visiting an area around each location, with each path limited with travel budget and undertaken by a Dubins vehicle. The DTOPN can be used to model situations where it is required to map an area using multiple Unmanned Aerial Vehicles (UAVs), the budget being their battery, flight time or maximal travelled distance and the curvature constraint representing the inability of unmanned planes to change their direction on a single spot, or a scenario with UAVs remotely collecting sensory data.

A Greedy Randomised Adaptive Search Procedure is introduced as a solution to the DTOPN. It is a flexible algorithm originally used to solve the TOP extended to consider the Dubins vehicle's minimal turning radius requirements and the locations' respective neighbourhoods.

The algorithm was tested in a real-life experiment which modelled data-collection scenario using three Micro Aerial Vehicles (MAVs) (Fig. 1) initially developed for multi-robot applications [48]. Three trajectories were generated (TOP, DTOP, DTOPN) for the MAVs to follow. The goal was to visit locations scattered on a field and visually inspect them with onboard cameras, thus collecting their associated reward.

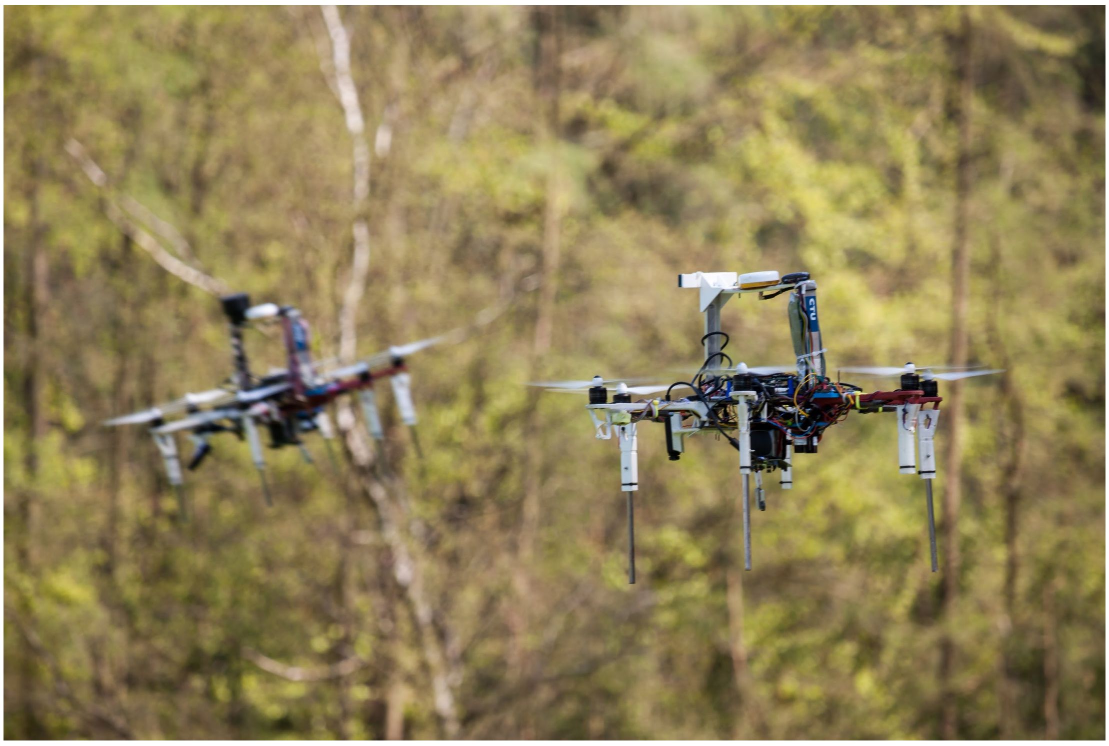


Figure 1: Two MAVs mid-flight used in the experimental verification of proposed method. (Courtesy of Matěj Petrlík)

2 State of the art

In this section, the two of the relevant problem definitions are explained, together with their derivatives, the real-life applications and algorithms that were used to solve them with particular attention to Team Orienteering Problem, Dubins Orienteering Problem and Orienteering Problem with Neighbourhoods, as they are the foundation of this thesis. Furthermore, the Greedy Randomised Adaptive Search Procedure with Path Relinking is introduced.

2.1 Travelling Salesman Problem

The most classical routing problem formulation is the TSP. It is widely studied in many fields, including computer science, operational research and robotics. The problem is defined as follows: A salesman wants to visit each of a set of cities exactly once and return to the starting city with minimal distance travelled [45]. Many algorithms were developed that solve it, since it is easy to formulate and, as an NP-hard problem [27], is difficult to solve. For example, the Ant Colony System as proposed by *Dorigo et al.* in [17] and further thoroughly described in [16]. It also serves as a basis for other problem definitions that generalise it by imposing other restrictions that extend its usefulness for many real-life applications.

One of those is the Dubins Travelling Salesman Problem (DTSP). It was first proposed by *K. Savla et al.* in 2005 [49]. Compared to the original (Euclidean) TSP, the DTSP uses the Dubins vehicle model [18] that features curvature constraints. The applied model allows planning routes for vehicles with limited turning radius, for example, fixed-wing UAVs, VTOL UAVs with constant forward speed or specific kinds of ground vehicles. However, this extension raises the computational capacity requirements, since it is necessary to calculate the best way to connect different headings for each pair of visited cities.

In [49], *K. Savla et al.* also proposed a simple solution to the DTSP with their Alternating Algorithm. It is based on the optimal solution of the Euclidean TSP to determine a sequence of visits to the goals. Dubins vehicle's headings are then found in the way that even edges are connected by straight line segments, and the odd edges correspond to the optimal Dubins manoeuvres. Since it addresses the DTSP in two consecutive steps, it can be called a decoupled approach. A similar approach has been utilised in the Receding Horizon Approach by *X. Ma and D. A. Castanon* in 2006 [35]. Another approach is to use sampling-based algorithms, in which the possible headings are sampled into a finite discrete set, and the DTSP is transformed into the Generalized Asymmetric TSP and further into Asymmetric TSP by Noon-Bean transformation [37]. The third approach is to use evolutionary algorithms [66], [67].

Profit-Based TSP is a generalisation of the traditional TSP, where it is not necessary to visit all cities. Each city has a pre-defined profit (value). The objective of the Profit Based Travelling Salesman Problem is to find a tour with a satisfying collected profit (maximised) and the least travel cost (minimised). One of the specific Profit based generalisations is the Selective Travelling Salesman Problem (STSP). The objective of STSP is to find a tour that maximises the profit collected in such a way that the travel cost does not exceed a predetermined value T_{max}. This is also called the Orienteering Problem and will be further explained in the next section.

For applications where the deployment of multiple salesmen is possible, the Multiple Travelling Salesman Problem (mTSP) generalisation of the TSP is very useful. It consists of determining a set of routes for m salesmen whom all start from and turn back to a home city (depot). Although the TSP has received a great deal of attention, the research on the mTSP is limited [3]. The mTSP has many specific versions. In single-depot version, every salesman starts and ends their tour at a single point. In multi-depot version, the salesmen can either return to their original depot after completing their tour or to any depot with the restriction that the number of salesmen at each depot remains the same after the travel. The former is called the fixed-destination case; the latter is named as the nonfixed-destination case. The number of salesmen also varies. An interesting version where the number of salesmen is not pre-determined exists. It associates a fixed cost to every salesman that incurs whenever this salesman is used in the solution. In this case, the minimisation of the number of salesmen to be activated in the solution is also of concern.

Using multiple salesmen is very useful when applied to modelling real-life situation, especially in routing and scheduling problems, as using more salesmen at once leads to an increase in the amount of cities visited in the same amount of time, and in real life applications, it is often possible to use more than one subject at the same time. Example applications on which the mTSP and it's variations can be applied are print press scheduling [25] [54], crew scheduling [31], the school-bus routing problem [1], mission planning [6] [7], hot rolling scheduling [56] and in the design of global navigation satellite system surveying networks [46]. Another problem is discussed in [40] that deals with balancing the workload among salesmen, which is relevant to work scheduling applications. Other interesting examples are the overnight security service problem, investigated by *Calvo and Cordone* [10], or the subproblem in the scheduling of quay cranes in ship operation planning by *Kim and Park* [41], where a branch and bound algorithm is used to find tight lower bounds.

The Dubins Travelling Salesman Problem with Neighbourhoods is also an important TSP generalisation. The term was first used by *Isaacs et al.* in 2011 [26], however, the first to tackle this problem was *K. Obermeyer* in 2009 [38]. It utilises the Dubins vehicle and combines it with the idea of neighbourhoods. The neighbourhoods are circular areas around each city in which the city's designated reward can be collected. Allowing to collect the reward from near distance of cities is a very useful extension since if the real-life application does not require visiting the exact location, travel cost can be minimised more effectively.

However, this also means that not only headings have to be determined, but also the particular points of visits can be selected from an infinite set. To tackle this optimisation problem, the headings and neighbourhoods can be sampled, which results in a finite set of heading and neighbourhood samples. This is called a sampling-based approach. Similar to the DTSP, approaches for the Euclidean TSP cannot be used. *Obermeyer et al.* [38] propose a genetic algorithm to address the TSPN. The authors further proposed an algorithm based on a randomised sampling resolution complete approach that transforms the TSPN into a variant of the Generalized TSP. It is then further transformed into the Asymmetric TSP that is solved by the LKH algorithm [39].

2.2 Orienteering Problem

The OP is a variation of the Travelling Salesman Problem that features a travel budget that must be met. This makes it a very useful problem formulation for real-life scenarios in robotics since it allows to account for travel limitations, such as battery life or limited fuel amounts. It combines two NP-hard optimisation problems, the Knapsack and the Travelling Salesman Problem [45]. The similarity of the OP to the Knapsack problem is in the goal to maximise collected reward by selecting a subset of target locations to be visited within the budget. This is combined with the TSP, in which the goal is to minimise the travel cost associated with visiting cities on a route.

The Euclidean OP (EOP), also known as the STSP, was introduced in 1987 by *Golden, Levy and Vohra* [24]. Its first application was however mentioned in 1984 by *A. Tsiligirides* in [58], using an example of a travelling salesman with not enough time to visit all possible cities. In [58], *Tsiligirides* also proposed two heuristics for tackling the Orienteering Problem. One is stochastic ($S - Algorithm$) and the second one is deterministic ($D - Algorithm$).

Another mention was in [24] by *Golden et al.* where a different practical application is presented, consisting of a fleet of trucks delivering to a large number of customers on a daily basis. The customer's fuel inventory level should be maintained at an adequate level at all times. That is, each customer has a known tank capacity, and its fuel level is expected to remain above a prespecified critical value which may be called the resupply point. The deliveries follow a "push system" in the sense that they are scheduled by the firm based on a forecast of the customers' tank levels. Stockouts are costly and are to be avoided whenever possible. The forecasted inventory level can be considered as a measure of urgency for each customer. The higher the urgency measure, the more important is it to service that customer immediately. A primary goal is to select a subset of customers to visit each day that urgently require service and that are clustered geographically in such a way as to foster the construction of efficient truck trips. The orienteering problem can be used to solve the subset selection step in the larger inventory/routing problem when urgency is replaced with score [24].

One widely used approach is the branch and bound method proposed by *Laporte and Martello* [30] in 1990 and *Ramesh et al.* in 1992 [44]. The branch-and-bound method later evolved into branch-and-cut algorithms which allowed to solve more complex instances. One of the branch-and-cut algorithms was proposed by *Fischetti et al.* in [23].

In 1996, *Chao et al.* [12] introduced a five-step heuristic that only considered reachable nodes and in result outperformed most previously presented approaches. Over the time, many other algorithms and heuristics were developed, as solving the Orienteering Problem is a difficult task to optimise effectively, as the computational power required is vast. For example, artificial neural network approach was proposed in 1995 [64], genetic algorithm was proposed in 2001 by *Tasgetiren et al.* [57] and Ant Colony optimization by *Liang et al.* [33] in 2002.

From the recently proposed solutions, the Discrete Strengthened Particle Swarm Optimization by *Sevkli and Sevilgen* [50] managed to improve one best known solution. Other recently proposed solutions are the Multi-Level Variable Neighbourhood Search by *Liang et al.* [32] and Greedy Randomised Adaptive Search Procedure with Path Relinking by *Campos et al.* [11]. Another proposed solution to the OP is the Variable Neighborhood Search (VNS) approach. It is a metaheuristic proposed in 1997 for solving combinatorial optimisation problems. The basic idea behind it is to search the solution space with a systematic change of neighbourhood [36]. To reduce the computational time required to solve the problem, *Liang et al.* [32] developed the Multi-level VNS, which allows solving certain identical instances concurrently. This lead to significant speed increases.

Other examples of practical applications of the Orienteering Problem can be the Mobile Tourist Guide, as proposed by *Souffriau et al.* [52] or even military applications, as proposed by *Wang et al.* [65]. In the Mobile Tourist Guide example[52], tourists visiting a city or a region are often unable to visit everything they are interested in. Thus, they have to select what they believe to be the most valuable attractions. Making a feasible plan in order to visit these attractions in the available time span is often a difficult task. These problems are also called Tourist Trip Design Problems [63]. This application requires high-quality solutions in only a few seconds of calculation time. A similar Tourist Trip Design Problem of selecting the most interesting combination of attractions is mentioned by *Wang et al.* in [65]. The military application considers a submarine or an unmanned aircraft involved in surveillance activities. The length of their expedition is limited by a fuel or time constraint, and the goal is to visit and photograph the best subset of all possible vertices.[65]

The Team Orienteering Problem extends the classical Orienteering Problem by using multiple subsets of nodes without intersections to maximise the collected reward. It is similar to the Multiple Travelling Salesmen Problem with the difference of the travel budget which originates from the Orienteering Problem. It was first introduced by *Chao et al.* in 1996 [13], even though a similar problem was introduced already by *Butt and Cavalier* in 1994 called Multiple Tour Maximum Collection Problem [8].

Chao et al. also proposed a heuristic based on their five-step heuristic used on the EOP [12]. Instead of selecting the best path, p best paths are selected, and two reinitialisation steps are applied instead of one. They also introduced benchmark instances for the TOP, which were partially based on datasets by *A. Tsiligirides* [58]. In 1999, *Butt and Ryan* [9] published an exact algorithm for the TOP that used column generation. *Boussier et al.* in 2007 used the same column generation approach, but coupled it with branch-and-bound to produce branch-and-price scheme [5]. They further used acceleration techniques to improve the performance. In 2010, *Bouly et al.* proposed a Memetic Algorithm [4] which managed to improve 5 of the best known solutions. It used GA combined with local search techniques and used an encoding process based on the Optimal Split procedure by *G. Ulusoy* [59]. Other algorithms include a branch-and-cut proposed in 2013 by *Dang et al.* [14], which was based on a linear formulation with a polynomial number of binary variables and managed to improve 29 best known solutions, a Particle Swarm Optimization inspired algorithm by *Dang et al.* [15], a Multi-start Simulated Annealing by *Lin et al.* [34] and a Pareto Mimic Algorithm by *Ke et al.* [28]. All of these mentioned algorithms were able to improve some of the existing best solutions.

The practical application of the TOP can be explained on recruitment planning of athletes from high schools. A recruiter has to visit several schools in a given number of days. He visits the schools beforehand and rates their recruitment potential. As the recruiter's time is limited, he has to choose the schools to visit each day to maximise the recruiting potential. Another example was proposed by *Tang and Miller-Hooks* [55] in which an application of TOP on routing technicians with the goal to service customers was mentioned. Each path represents a single technician who can only work a limited number of hours a day. Thus, not all customers requiring service can be included in the technicians' daily schedules. A subset of customers will have to be selected, taking into account customer importance and task urgency. The Home Fuel Delivery Problem, as described by *Golden et al.* in [24] in relationship with the EOP, is also applicable, since using only one truck to deliver fuel would provide less satisfied customers than using a fleet.

The Orienteering Problem with Neighbourhoods was first proposed in 2016 by *Faigl et al.* [20]. The OPN is a generalisation of the Orienteering Problem in which reward collection within given non-zero communication range of target locations is possible. Unlike the OP, the OPN requires to determine the position within the circular neighbourhoods of each target location and is, therefore, more computationally demanding due to the increased problem size. In order to be able to solve the OPN with reasonable computational power, the neighbourhood can be sampled to a finite number of new locations, similar to headings in the DOP.

In [20], *Faigl et al.* also proposed the use of Self Organizing Map-based algorithm to solve the OPN. It was based on the unsupervised learning of Self-Organizing Map used to solve the Prize-Collecting Travelling Salesman Problem and used neural network approach. It does not reliably converge to stable results, but it produces feasible solutions every run.

The practical application of the OPN is in situations where remote data collection is possible because it allows planning routes through a close vicinity of each location. This results in a larger collected reward, as the travel price can be lower when it is not necessary to visit the exact location, thus making it possible to visit more locations. Larger collected reward then translates to more data collected in the same amount of time.

The Dubins Orienteering Problem, like the Dubins Travelling Salesman Problem, utilises the Dubins vehicle with limited turning radius. The term was first used in 2017 by *Pěnička et al.* [42]. Compared to the Dubins Travelling Salesman Problem, the DOP is better suited for real-life applications in robotics, since it not only considers the curvature constraints of fixed-wing aeroplanes but also the fuel or battery limits of UAVs.

In [42] an algorithm by *Pěnička et al.* based on the Variable Neighborhood Search algorithm originally used on OP was proposed and verified in real-life testing scenarios. The algorithm iteratively performs shake and local search procedures. The shake routine randomly changes the best achieved solution to escape from a local maximum. The local search procedure then improves the solution created by shake. It was also discovered that the collected reward decreases with increasing Dubins vehicle turning radii.

The Dubins Orienteering Problem with Neighbourhoods (DOPN) combines both the limited curvature constraint of Dubins vehicle and the ability to measure data within a predefined circular neighbourhood around each target location. It is also known as the Close Enough Orienteering Problem with Dubins Vehicle [21]. Because of the neighbourhood extension, the collected reward is higher than in DOP, even though it uses the same Dubins vehicle model. It was proposed in 2017 by *Pěnička et al.* [43] together with a VNS-based metaheuristic. It produced feasible results with a larger collected reward than in DOP by saving travel costs and even outperformed the only existing SOM-based approach for non-overlapping neighbourhoods in the Euclidean Orienteering Problem with Neighbourhoods.

The Dubins Team Orienteering Problem with Neighbourhoods is a new routing problem formulation that combines the TOP and DOPN. Its objective is to find a set of p length limited paths between starting and ending locations with maximal sum of collected rewards containing nodes from a specified set of locations with pre-determined reward. Because it combines two NP-hard problems, the DTOPN itself is NP-hard. The same sampling procedure, as was used in the DOPN to reduce the number of possible headings and neighbourhood locations to a finite number, can be applied to the DTOPN.

The practical application can be surveillance missions or remote data collecting scenarios using multiple curvature-constrained vehicles, such as fixed-wing or multirotor UAVs with constant forward speed. This can prove crucial in situations where it is dangerous or even impossible to slow the aircraft.

2.3 Greedy Randomised Adaptive Search Procedure

Since great computational power is required to solve the DTOPN, a fast algorithm providing good results is required. Ideal candidates are algorithms able to solve the TOP since the Dubins and Neighbourhoods extension only modifies the criteria of path creation.

According to a survey published by *P. Vansteenwegen* [62], the best scoring TOP heuristics are Slow VNS by Archetti et al., which managed to have the lowest average gap between the generated and best-known solution, and Ant Colony Optimization by *Ke et al.* (2008), which managed to find the highest number of best solutions.

The first to focus on obtaining good results in only a few seconds of computational time were *Vansteenwegen et al.* using a Guided Local Search approach [60] and VNS approach [61]. Their proposed Skewed Variable Neighbourhood Search reduced the minimal achieved computational time from 63.6 seconds to 3.8, making it significantly faster than other heuristics while still managing to provide good results.

Very good results were achieved by the Greedy Randomised Adaptive Search Procedure with Path Relinking extension by *W. Souffriau* (2010) [51]. It is capable of two modes of work, the Fast Path Relinking (FPR) and Slow Path Relinking (SPR). In the FPR mode, the algorithm ended up the second fastest with an average gap in between the fast performing SVNS by Vansteenwegen et al. and the good results providing Ant Colony Optimization by Ke et al.

In the SPR mode, it achieved lower average gap than VNS by Archetti et al. and found more best solutions found than Ant Colony Optimization by Ke et al. while maintaining faster execution time than both of them. This makes it a very flexible algorithm that is well-suited for reliable and fast solving of the NP-hard DTOPN and therefore is proposed in this thesis as a solution to the problem.

3 Problem formulation

In this section, the Team Orienteering Problem, Dubins TOP and DTOP with Neighbourhoods are mathematically formulated as optimisation problems together with their constraints.

3.1 Team Orienteering Problem

The TOP is a combination of the EOP and mTSP. The goal is to determine P paths, each limited with T_{max}, that maximise the total collected reward. Each path is a subset $S_k^p \subset S$, where $S = \{s_1, ..., s_n\}$ is a set of target locations. The origin and ending locations are given and represented as s_1 and s_n. Each considered target location s_i is defined by its position denoted as $s_i \in \mathbb{R}^2$ and its reward r_i. The reward of both the starting and the ending location is assumed to be zero $r_1 = r_n = 0$ and is strictly positive for all other locations. The goal of the TOP is to determine for each of P paths, $p = (1, \dots, P)$, a set of k target locations that define the subset $S_{k_p}^p$.

The sequence of their visits can be described as a permutation over k_p target location of path p using $\Sigma_{k_p}^p = (\sigma_1^p, \dots, \sigma_{k_p}^p)$, with constraints $1 \leq \sigma_i^p \leq n$, $\sigma_i^l \neq \sigma_j^m$ for $i \neq j \wedge l \neq m$ and $\sigma_1^p = 1$, $\sigma_{k_p}^p = n$, where σ_i^p represents the target location index. These ensure that each node is visited by at most one path and at most once.

For the Euclidean distance $\mathcal{L}_e(s_{\sigma_i^p}, s_{\sigma_j^p})$ between two locations $s_{\sigma_i^p}$ and $s_{\sigma_j^p}$ both belonging to path p, the TOP can be formulated as the following optimisation problem:

$$\begin{aligned}
\underset{\Sigma_{k_p}^p, k_p, p \in (1,\dots,P)}{Maximise} \quad R &= \sum_{p=1}^{P} \sum_{i=1}^{k_p} r_{\sigma_i^p} \\
s.t. \quad \sum_{i=2}^{k_p} \mathcal{L}_e(s_{\sigma_{i-1}^p}, s_{\sigma_i^p}) &\leq T_{max} \quad \forall p \in (1, \dots, P) \ , \\
\sigma_i^l &\neq \sigma_j^m \text{ for } i \neq j \wedge l \neq m \ , \\
\sigma_1^p &= 1 \ , \ \sigma_{k_p}^p = n \quad \forall p \in (1, \dots, P) \ ,
\end{aligned} \tag{1}$$

where R represents the total collected reward.

The TOP can also be formulated as an integer problem with these decision variables: If in a path p a visit to vertex i is followed by a visit to vertex j, $x_{ijp} = 1$. Otherwise, $x_{ijp} = 0$. If vertex i is visited in path p, $y_{ip} = 1$. Otherwise, $y_{ip} = 0$. The position of vertex i in path p is represented as u_{ip}.

$$Maximise \sum_{p=1}^{P} \sum_{i=2}^{N-1} r_i y_{ip}, \tag{2}$$

$$s.\ t.\ \sum_{p=1}^{P} \sum_{j=2}^{N} x_{1jp} = \sum_{p=1}^{P} \sum_{i=1}^{N-1} x_{iNp} = P, \tag{3}$$

$$\sum_{p=1}^{P} y_{kp} \leq 1; \forall k = 2, ..., N-1, \tag{4}$$

$$\sum_{i=1}^{N-1} x_{ikp} = \sum_{j=2}^{N} x_{kjp} = y_{kp}; \forall k = 2, ..., N-1; \forall p = 1, ..., P, \tag{5}$$

$$\sum_{i=1}^{N-1} \sum_{j=2}^{N} t_{ij} x_{ijp} \leq T_{max}; \forall p = 1, ..., P, \tag{6}$$

$$2 \leq u_{ip} \leq N; \forall i = 2, ..., N; \forall p = 1, ..., P, \tag{7}$$

$$u_{ip} - u_{jp} + 1 \leq (N-1)(1 - x_{ijp}); \forall i, j = 2, ..., N; \forall p = 1, ..., P, \tag{8}$$

$$x_{ijp}, y_{jp} \in 0, 1, \forall i, j = 1, ..., N; \forall p = 1, ..., P. \tag{9}$$

The objective of the TOP is to maximise the total reward of all vehicle tours, as shown in (2). In this formulation, constraint (3) ensures that there are p tours starting and ending at vertex 0. Constraints (4) ensure that each vertex can be visited at most once, except for vertex 0. The connectivity of the tour is ensured by constraints (5) and limitation on the total duration for each tour is imposed by constraints (6). Sub-tours are prohibited by (7). Constraints (8) and (9) set integral requirements on the decision variables.

The TOP formulated in (2)-(9) is NP-hard. This is because when $p = 1$, the TOP reduces to a Selective Travelling Salesman Problem, which is NP-hard [30].

3.2 Dubins Team Orienteering Problem

In the DOP, the state of the Dubins vehicle $q = (x, y, \theta)$ consists of its position in plane $s = (x, y) \in \mathbb{R}^2$ and its heading ($\theta \in \mathbb{S}^1$), i.e., $q \in SE(2)$. The vehicle model is thus non-holonomic. The minimal turning radius ρ influences the length of the shortest path between two states. The kinematic model of Dubins vehicle with a constant forward velocity v and control input u can be described as:

$$\dot{q} = \begin{bmatrix} \dot{s} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = v \begin{bmatrix} cos\theta \\ sin\theta \\ \frac{u}{\rho} \end{bmatrix}, u \in [-1, 1]. \tag{10}$$

For model (10), the shortest path between two states consists only of a straight line (L-segment) and arcs with the radius ρ (C-segment). The optimal path is then one of two possible manoeuvres CCC, CLC, where C can be either a left-turning or right-turning arc [18], resulting in six possible combinations. These are further denoted as Dubins manoeuvres.

While the Dubins maneuver for any two states q_i and q_j with its length $\mathcal{L}_d(q_i, q_j)$ can be determined analytically, in the DOP it is necessary to determine the particular headings θ_i and θ_j of the vehicle at corresponding locations s_i, s_j, respectively.

Each target location s_i is thus in the DOP considered as the state $q_i = (s_i, \theta_i)$ and in addition to the determination of the subset S_k of the k locations in the route and the permutation $\Sigma = (\sigma_1, ..., \sigma_k)$, the DOP intends to find the corresponding heading angles $\Theta = (\theta_{\sigma_1}, ..., \theta_{\sigma_k})$

The Dubins Orienteering Problem for the model can be then formulated as the optimisation problem

$$\begin{gathered} \underset{k, S_k, \Sigma, \Theta}{Maximise} \ R = \sum_{i=1}^{k} r_{\sigma_i}, \\ s.\ t.\ \sum_{i=2}^{k} \mathcal{L}_d(q_{\sigma_{i-1}}, q_{\sigma_i}) \leq T_{max}, \\ \sigma_1 = 1, \sigma_k = n. \end{gathered} \tag{11}$$

In contrast to the Euclidean OP, the DOP considers the Dubins vehicle model and the path is constructed using the Dubins manoeuvres between the adjacent target locations (states). The optimisation problem is not only over all possible subsets and respective permutations of the target locations (k, Σ), but also over all possible heading angles Θ of the target locations. This makes the problem computationally challenging as the already NP-hard EOP is extended to optimise over heading angles.

For the DTOP, the described problem formulation must be modified to accommodate multiple paths. Each of the locations s_i^p is then considered as the state $q_i^p = (s_i^p, \theta_i^p)$ and the permutation becomes $\Sigma_{k_p} = (\sigma_1^p, ..., \sigma_{k_p}^p)$ with corresponding heading angles $\Theta_p = (\theta_{\sigma_1^p}, ..., \theta_{\sigma_{k_p}^p})$ The optimisation problem is thus:

$$\begin{aligned} \underset{\Sigma_{k_p}^p, \Theta_p, k_p, p\in(1,\ldots,P)}{Maximise} \quad R &= \sum_{p=1}^{P} \sum_{i=1}^{k_p} r_{\sigma_i^p} \\ s.t. \quad &\sum_{i=2}^{k_p} \mathcal{L}_d(q_{\sigma_{i-1}^p}, q_{\sigma_i^p}) \leq T_{max} \quad \forall p \in (1, \ldots, P) \ , \\ &\sigma_i^l \neq \sigma_j^m \text{ for } i \neq j \wedge l \neq m \ , \\ &\sigma_1^p = 1 \ , \ \sigma_k^p = n \quad \forall p \in (1, \ldots, P) \ . \end{aligned} \tag{12}$$

3.3 Dubins Team Orienteering Problem with Neighbourhoods

In the Dubins Team Orienteering Problem with Neighbourhoods, the reward can be collected by visiting a circular neighbourhood around each location. The specific neighbourhood is described by the neighbourhood radius parameter δ that defines a δ-radius disk centred at the respective target location coordinates. It is expected for all target locations to have the same value of δ, except the starting location s_0 and the ending location s_k with zero neighbourhood radius.

In contrast to the DTOP where k, Σ and Θ are determined, the DTOPN also requires determination of particular locations of the waypoints $W \subseteq \mathbb{R}^2$ at which the rewards are collected, where the waypoints are within δ distance from the respective target locations, i.e., $w_{\sigma_i} \in W,\ s_{\sigma_i} \in S_k$ and $|(w_{\sigma_i}, s_{\sigma_i})| \leq \delta$. The Dubins Orienteering Problem with Neighbourhoods can be then described as the optimisation problem:

$$\begin{aligned} \underset{k,S_k,W,\Sigma,\Theta}{Maximise}\ R &= \sum_{i=1}^{k} r_{\sigma_i}, \\ \text{s. t.} \sum_{i=2}^{k} &\mathcal{L}_d(q)\sigma_{i-1}, q_{\sigma_i}) \leq T_{max}, \\ &q_{\sigma_i} = (p_q\sigma_i, \theta_{\sigma_i}, w_{\sigma_i} \in W_k, \theta_{\sigma_i} \in \Theta). \end{aligned} \tag{13}$$

Four important variables must be determined to solve the DTOPN. These are, similar to the TOP, S_{k_p} and Σ_{k_p}, where S_{k_p} represents the locations in each route and thus influences the total collected rewards R, and the permutation Σ_{k_p} that defines the length of each path p over S_{k_p} constrained by the budget T_{max}. Furthermore, the DTOPN solution contains the sequence of heading angles Θ_{k_p} at the target locations that influence the length of each path because of the curvature constraints of the Dubins vehicle.

The final path length is also influenced by the neighbourhoods of the respective target locations implied by $||w_{\sigma_i}, s_{\sigma_i}|| < \delta$. This results in additional search among the waypoints $W_{k_p} = (w_{\sigma_1}, ..., w_{\sigma_{k_p}})$ and is the reason why the DTOPN is more challenging than the DTOP or TOP since it adds additional part of the continuous optimisation for the locations of the waypoints in $\mathbb{R}^2$.

In the DTOPN, the same applies, only modified to accommodate multiple routes. The variables become $w_{\sigma^p_{k_p}} \in W^p_{k_p} = (w_{\sigma^p_1}, \ldots, w_{\sigma^p_{k_p}}), s_{\sigma^p_i} \in S^p_{k_p}$ where $|(w_{\sigma^p_i}, s_{\sigma^p_i})| \leq \delta$. The DTOPN can be described as the optimisation problem:

$$\begin{aligned}
\underset{\Sigma^p_{k_p},\Theta_p,W^p_{k_p},k_p}{Maximise}\; R &= \sum_{p=1}^{P}\sum_{i=1}^{k_p} r_{\sigma^p_i} \\
s.t. \; \sum_{i=2}^{k_p} &\mathcal{L}_d(q_{\sigma^p_{i-1}}, q_{\sigma^p_i}) \leq T_{max} \quad \forall p \in (1, \ldots, P) \;, \\
&q_{\sigma^p_i} = (w_{\sigma^p_i}, \theta_{\sigma^p_i}), w_{\sigma^p_i} \in W_{k_p}, \theta_{\sigma^p_i} \in \Theta_{k_p} \forall p \in (1, \ldots, P), \\
&||w_{\sigma^p_i}, s_{\sigma^p_i}|| \leq \delta \; \forall i \in (2, k-1) \forall p \in (1, \ldots, P), \\
&||w_{\sigma^p_1}, s_{\sigma^p_1}|| = 0, ||w_{\sigma^p_k}, s_{\sigma^p_k}|| = 0, \\
&\sigma^l_i \neq \sigma^m_j \text{ for } i \neq j \wedge l \neq m \;, \\
&\sigma^p_1 = 1 \;, \sigma^p_k = n \quad \forall p \in (1, \ldots, P) \;.
\end{aligned} \tag{14}$$

4 GRASP with Path Relinking for the DTOPN

The algorithm (1) consists of four procedures that are repeated indefinitely until the termination condition is met. The termination condition, in this case, is a number i of iterations without improvement.

The four procedures are Construct (2), Local Search (3), Link to Elites (4) and Update Elites (5). Construct and Local Search create and subsequently improve an independent solution, which is then improved using the best solutions already found by Link to Elites. Update Elites then updates the elite pool by inserting or replacing existing elites with worse score than the solution created by (2 - 4).

Algorithm 1 GRASP-PR

```
1: while Nr of iterations without improvement do
2:     Construct
3:     Local Search
4:     Link to Elites
5:     Update Elites
6: end while
```

4.1 Construct Procedure

The Construct procedure outlined in Algorithm 2 is responsible for generating new solutions that are subsequently improved. It is characterised by Greediness, a parameter randomly drawn from uniform distribution $< 0, 1 >$. Greediness describes the exact ratio between greediness and randomness of the currently constructed solution. Its random nature helps in generating various distinctive solutions and exploring new possible means to achieve a higher reward.

Algorithm 2 Construct

```
1: Draw greediness
2: Candidates = $l$ - infeasible nodes
3: Set $h_{threshold}$
4: Discard candidates below $h_{treshold}$
5: while Candidates left to consider do
6:     if Random candidate fits in a route then
7:         Insert to a route
8:     else
9:         Mark candidate as considered
10:    end if
11: end while
```

First, all infeasible nodes are discarded. Feasibility is checked by calculating the price of each node l, which is equal to

$$t_l = t_{il} + t_{lj} \tag{15}$$

where i represents the starting node and j represents the ending node and t_l is the travel distance to visit the node. If $t_l < budget$, the node is feasible and inserted into a candidate list. Otherwise, it is discarded.

Afterwards, each node's heuristic value is calculated:

$$h_l = \frac{r_l}{t_l}, \tag{16}$$

and threshold is set as

$$h_{threshold} = (h_{max} - h_{min}) \cdot greediness\ . \tag{17}$$

The candidates that fulfil the requirement $h_l > h_{threshold}$ are inserted into a restricted candidate list.

Then, all P paths of current solution are populated by randomly selected nodes from the restricted candidate list so that the total length of a path p with k_p nodes

$$t^p_{total} = \sum_{i=2}^{k_p} t_{i-1,i} \tag{18}$$

does not violate the travel budget. The procedure ends when there are no feasible nodes left to include and returns P feasible paths of the current solution.

4.2 Local Search Procedure

The Local Search procedure summarised in Algorithm 3 tries to improve the solution using four different ways until finding the local optimum. It alternates between reducing the total price of the P solution paths and increasing the value by considering non-included nodes.

Algorithm 3 Local Search

```
1: while Improvement do
2:     Simplified 2-opt
3:     Swap
4:     Replace
5:     Insert
6: end while
```

First, a simplified 2-opt is applied. If a price of a route can be decreased by exchanging two locations already present in the route, they are swapped. This ensures that the total price of all P paths is minimal. The simplified 2-opt is used because of the increased computational requirements that the Dubins and neighbourhoods extensions pose.

Next, the neighbourhood between paths of the solution is explored in such a way that minimises the total price. If an improvement in the total price can be achieved by swapping two locations already present in the solution, they are swapped. This continues until no possible price improvement can be achieved by exchanging nodes between routes.

Afterwards, Replace tries to improve the total claimed score by replacing nodes currently in the solution with feasible nodes not included. The algorithm considers all non-included feasible nodes and finds their cheapest insertion place in the current solution. If it would violate the travel budget, the node with the lowest score that could be feasibly replaced is found and replaced. The Replace procedure executes moves that result in the best collected score increase, and, in case of a tie, the best collected score increase and the lowest solution price increase.

Finally, Insert tries to insert non-included nodes into the current solution in a way that would not increase the total price over the travel budget.

The procedure ends when the specified number of iterations without improvement is reached and returns the local optimum. The specific number of iterations without improvement is dependent on the algorithm mode used. It is lower when using the Fast Path Relinking mode (FPR) than when using the Slow Path Relinking mode (SPR). This parameter influences the speed and quality of results produced by the algorithm. Higher number of iterations without improvement leads to slower execution time with better results provided and vice versa.

4.3 Link to Elites Procedure

The Link to Elites procedure described in Algorithm 4 serves as a long-term memory component which ensures that the complete independence between solutions generated by Construct and Local Search procedures is avoided.

The procedure takes two solutions as an argument, a starting solution and a guiding solution, and explores the neighbourhood between them. It is executed for the current solution as a starting solution and subsequently all members of the elite pool as a guiding solution and vice versa. Every time the procedure uses an elite as a guiding solution, it's age increments by 1. This ensures that the algorithm does not get stuck in the local optimum and explores as many feasible solutions as possible. The procedure returns the best solution found.

Algorithm 4 Link to Elites

```
1: Set intersections = common nodes of starting and guiding solutions
2: Set locationsToAdd = guiding solution minus intersections
3: while locationsToAdd is not empty do
4:     while At least one route feasible do
5:         Insert locations, allow infeasibility
6:     end while
7:     while At least one route over budget do
8:         Remove locations from infeasible paths
9:         Local Search
10:    end while
11: end while
```

The first step is creating a list of intersections between starting and guiding solutions, which contains all the nodes both solutions have in common. Then, a list of nodes to add is found by removing all intersection nodes from the guiding solution. This prevents adding duplicate locations into the solution.

Furthermore, the similarity between the starting solution and the guiding solution is calculated. If the similarity is above a certain threshold, the procedure does not continue. This helps to ensure that the resulting solutions improved by the Link to Elites procedure are new and different from existing elite solutions, which is necessary for discovering new and better solutions. It also helps to speed up the algorithm by minimising the possibility of generating a solution identical to one already generated.

While the list of nodes to add is not empty, the procedure considers inserting nodes from the list of nodes to add to feasible paths. This differs from the way the Construct procedure works because even moves that would violate the travel budget are considered.

When all m paths are infeasible, the algorithm restores feasibility by removing nodes.

For each node, it's heuristic value

$$h^{-1} = \frac{t_l}{r_l} \tag{19}$$

is calculated. The node with the highest h^{-1} is then removed. This continues until all paths are feasible again. The algorithm iterates through all routes of the solution and removes a node from each infeasible route. The Local Search procedure is then called to optimise the solution price after iterating through all routes, which helps in preventing unnecessary location removal. The procedure ends when the list of nodes to add is empty and returns a list of elite candidate solutions.

4.4 Update Elites Procedure

The Update Elites procedure outlined in Algorithm 5 takes the best solution found by Link to Elites and compares it to recently best found solutions called Elites. There is a maximum of 10 Elites which are stored in the Elite Pool.

Algorithm 5 Update Elites

```
 1: for all Elite do
 2:     if Elite used as a guiding or starting solution then
 3:         Elite.age+=1
 4:     end if
 5: end for
 6: if Elite.age ≥ max age then
 7:     Remove it
 8: end if
 9: if Solution == an Elite then
10:     Elite.age = 0
11: else if Elite pool not full then
12:     Add solution to Elite Pool
13: else
14:     Find worst Elite
15:     if Solution reward > worst Elite then
16:         Replace the Elite
17:     end if
18: end if
```

If any Elite solution reached the age of

$$max(10, \frac{i}{10})$$

where i is the number of iterations without improvement, it is removed. If the solution found is equal to an existing elite solution, the elite's age is reset to 0. If not, and the elite pool has not reached the maximum number of members (10), the solution is added. Otherwise, if the solution's collected score is higher than the worst elite's, it is replaced. This populates the Elite Pool, which is necessary for the solutions in order to be at least partially non-random.

4.5 Dubins Extension

To use the GRASP algorithm for the DTOPN, the algorithm must first be extended in order to consider the Dubins Vehicle constraints.

The procedure uses a sampling-based approach to create a discrete set of headings for each location by proportionally sampling possible heading angles with the provided headings resolution R_D. This is necessary because otherwise, the set of different heading angles would be infinite. The exact number of samples generated this way is given by the heading resolution parameter R_D. Next, the method used to calculate distances between each location is extended to calculate the distance between every sampled heading of each location using Dubins manoeuvres with the given minimal turning radius as a parameter. This ensures that locations requiring sharp turns to visit are now penalised since curved routes have larger travel cost than straight lines. The procedure is described in Algorithm 6.

Algorithm 6 Calculate DTOP distances

```
1: for all Pairs of locations, headings do
2:     Calculate distance
3: end for
```

The method responsible for adding locations to routes is modified to use the set of distances created by the previously mentioned procedure (6). Furthermore, when adding new locations, the optimal heading sample is calculated based on the travel distance and based on the entry heading of location. This ensures that the travel cost associated with Dubins manoeuvres is minimised.

Finding the optimal headings that minimise the travel cost can be described as a Dubins Touring Problem [22]. To minimise the length of a path, a graph search algorithm is used to determine the sequence of headings that produce a path with minimal travel cost. The algorithm is developed to utilise a dynamic programming technique [42] to store distances $||(q_{\sigma_1}, q_{\sigma_i})||$ and $||(q_{\sigma_i}, q_{\sigma_k})||$ for every location i to decrease the computational requirements needed to re-calculate distances when adding, exchanging or removing locations. The graph is visualised in Fig. 2.

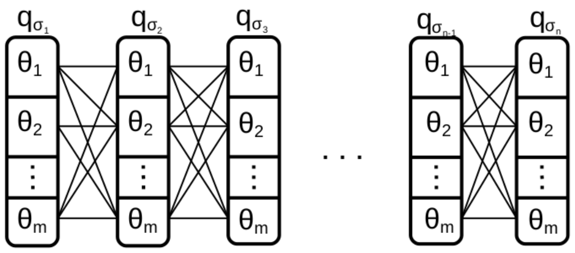


Figure 2: Search graph with m uniformly sampled headings at each target location $q_{\sigma_i}, 0 \leq i \leq k$. Graph search algorithm over all heading sample combinations is utilised to find the path with minimal length connecting specified target locations $(q_{\sigma_1}, \ldots, q_{\sigma_k})$.

4.6 Neighbourhoods extension

The neighbourhoods extension is required for the GRASP algorithm to be able to solve the DTOPN. The neighbourhoods extension utilises sampling-based operation to reduce the number of samples for each vertex. This is necessary to reduce the number of neighbourhood samples since it is otherwise infinite.

Using a given neighbourhood resolution parameter R_N, the circular radius of location neighbourhoods is equidistantly sampled into a finite number of points. This results in a set of new locations, called neighbourhood locations, in a circle of diameter δ around each of target locations. The method responsible for calculating distances between locations is then modified to iterate through every newly created neighbourhood location and use their coordinates to calculate distances in addition to iterating through all locations and all heading samples. The modified Calculate distances procedure is summarised in Algorithm 7.

Algorithm 7 Calculate DTOPN distances

```
1: for all Pairs of locations, neighbourhoods, headings do
2:     Calculate distance
3: end for
```

The method responsible for adding locations is then modified to find the neighbourhood location and heading with lowest price increase when adding a new location. This ensures that every constructed solution uses routes with minimised travel costs. It uses distances calculated by the previously mentioned Algorithm 7.

To find the optimal neighbourhood sample, a similar graph search algorithm used to find optimal headings is used (Fig. 2). The information about which neighbourhood location was visited is then stored in the route data, since it is necessary for further optimisation and to eliminate the need to calculate the best headings and neighbours for every location when only one has changed.

5 Experimental Results

In this chapter, the testing instances are presented, and the results of the application of the GRASP-PR algorithm on the TOP, DTOP and DTOPN are compared, together with the influence of specific parameters on provided solutions. Furthermore, the real-life experiment used to verify the performance is presented, and its results are discussed.

In this section, the modes Slow Path Relinking (SPR) and Fast Path Relinking (FPR) correspond to the parameter number of iterations without improvement, which is set as 300 for SPR and 10 for FPR. These values are based on the results in [51] where they were found as the best-performing combination.

5.1 Testing Instances

To test the algorithm, the TOP instances by *Chao et al.* are used. The instance sets used are Chao 4-7 [13]. Each of the test instance sets contains multiple datasets with the same locations, but with different travel budget and number of paths parameters. Every dataset contains a set of location coordinates with their corresponding rewards. The datasets in Chao 4-7 are geometric (square, diamond). Every dataset name contains information about the budget, the number of paths used and the set it belongs to. For example, in dataset *p6.2.j*, the *p6* signifies it belongs to the set Chao 6, *2* means two paths and the letter *j* represents the travel budget T_{max}, where *a* is the lowest from the set.

5.2 Implementation performance

In order to establish a common ground for the results of the DTOPN solutions and other problem formulations, the results of this implementation of the GRASP-PR algorithm are compared to the existing computational results of GRASP-PR for the TOP. Table 1 contains results of both a single run of the algorithm and the best result of 10 runs, both using a sample of datasets from Chao 4-7, compared to the results of GRASP-PR presented in [51] which are denoted as *SPR10 Original* and *FPR10 Original*.

The single-run results in Table 1 have a significant gap between its collected reward and the reference. However, in situations where it is crucial to produce solutions in a small amount of time, the results of a single run could be acceptable when exchanged for faster execution time.

The maximal results obtained after 10 runs of SPR (SPR10) and 10 runs of FPR (FPR10) still present a noticeable gap, albeit lower, when compared to the best achieved GRASP results. This is a result of different implementations.

Table 1: Reward collected by the GRASP algorithm used in this thesis compared to results by *W. Souffriau et al.*[51].

Dataset	FPR10 Original	SPR10 Original	FPR	SPR	FPR10	SPR10
p4.2.b	341	341	244	252	224	252
p4.3.j	858	861	658	761	774	785
p4.4.j	732	732	482	486	550	534
p5.2.n	925	925	830	900	910	900
p5.3.n	1070	1070	725	735	735	735
p5.4.p	760	760	675	740	710	740
p6.2.j	942	948	912	918	912	924
p6.3.l	1002	1002	894	942	912	978
p6.4.k	528	528	444	468	462	474
p7.2.j	632	646	559	624	597	628
p7.3.i	480	485	394	432	413	435
p7.4.i	364	366	308	312	293	303

Table 2: Average gap between results in [51] and presented implementation of GRASP-PR.

Gap FPR	Gap SPR	Gap FPR10	Gap SPR10
17.49%	12.80%	14.83%	12.00%

5.3 Reward collection

Both the Dubins extension and neighbourhoods extension influence the total collected reward. In Tables 3 and 4, the collected reward and execution times for a sample of the aforementioned datasets are presented. The parameters used for the DTOP were Dubins resolution $R_D = 8$ and Dubins radius $\rho = 0.5$. The same R_D and ρ were used for the DTOPN together with neighbourhood resolution $R_N = 8$ and neighbourhood radius $\delta = 0.5$. Example TOP trajectory is shown in Fig. 3a, DTOP in Fig. 3b and DTOPN in Fig. 3c.

As shown in Table 3, the average collected reward in the DTOP instances was lower than in the TOP. This is due to the curvature constraints of the Dubins vehicle which increase the length of arcs connecting the locations, thus increasing the travel costs. In the DTOPN instances, the average collected reward was highest, since the neighbourhoods extension lowers the travel budget required to visit locations.

The data presented in Table 4 show that the execution time required to solve the TOP, DTOP and DTOPN instances varies. This is due to the randomised nature of the algorithm. However, even with the variations, a trend of increasing execution time with each extension is present. While the DTOPN can save the travel costs expended to visit locations, thus increasing the collected reward, the execution time required increases as well, due to the additional neighbourhood samples.

Table 3: Reward collected in TOP, DTOP and DTOPN instances

Dataset	FPR TOP	FPR DTOP	FPR DTOPN	SPR TOP	SPR DTOP	SPR DTOPN
p4.2.b	244	238	247	252	238	259
p4.3.j	713	200	694	785	718	812
p4.4.j	482	15	505	534	488	563
p5.2.n	830	391	920	900	850	950
p5.3.n	725	690	820	735	690	840
p5.4.p	675	655	745	740	700	845
p6.2.j	912	906	1086	924	870	1164
p6.3.l	894	900	978	978	888	1098
p6.4.j	306	282	420	330	312	462
p7.2.j	559	582	543	628	607	656
p7.3.i	394	377	321	435	419	461
p7.4.i	308	273	309	303	273	311
Average	586.8	459.1	632.3	628.7	587.5	701.8

Table 4: Execution time in seconds used to solve TOP, DTOP and DTOPN instances.

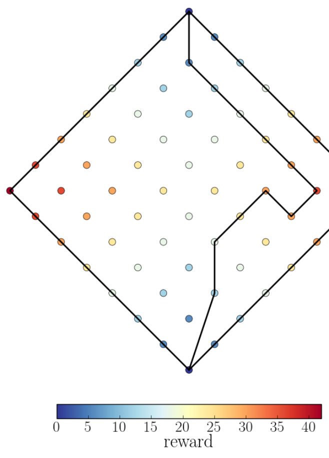
Dataset	FPR TOP	FPR DTOP	FPR DTOPN	SPR TOP	SPR DTOP	SPR DTOPN
p4.2.b	0.14	1.06	4.87	1.35	2.03	30.56
p4.3.j	0.95	0.90	12.25	18.30	32.38	128.52
p4.4.j	0.03	0.28	2.78	1.52	2.34	11.58
p5.2.n	0.62	1.37	54.12	18.82	53.94	468.16
p5.3.n	0.80	2.79	12.50	5.60	16.46	121.47
p5.4.p	0.41	0.80	2.18	2.21	6.90	37.20
p6.2.j	4.37	18.54	39.42	23.76	43.257	674.95
p6.3.l	0.89	7.82	4.10	15.38	33.82	61.08
p6.4.j	0.02	0.42	1.28	0.31	0.60	8.55
p7.2.j	1.50	11.89	10.85	35.17	43.73	524.67
p7.3.i	0.35	0.93	2.65	2.04	3.86	30.56
p7.4.i	0.05	0.57	2.85	0.42	0.53	4.88
Average	0.84	3.95	12.32	10.41	19.99	175.18

5.4 Dubins Resolution Parameter

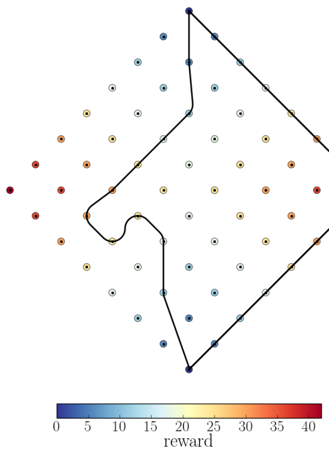
The Dubins resolution parameter R_D specifies how many heading samples are created for each location. It affects the total collected reward of the DTOPN solution since, with a higher number of samples, more precise results are computed. This, however, also influences the computational time required, because the higher number of samples increases the total number of possible combinations of locations and headings. The collected reward and computational times obtained for different values of Dubins resolution R_D can be seen in Tables 5 and 6, respectively. The results are for the SPR with Dubins radius $\rho = 0.5$, neighbourhood radius $\delta = 0.5$ and neighbourhood resolution $R_N = 8$. The graph in Fig. 4 shows that for higher resolutions, larger average rewards are collected, but with increasing execution time. However, due to the randomised nature of the algorithm, spikes in execution time can appear, as visible for $R_D = 10$.

Table 5: Collected reward for different Dubins resolution parameters in the DTOPN.

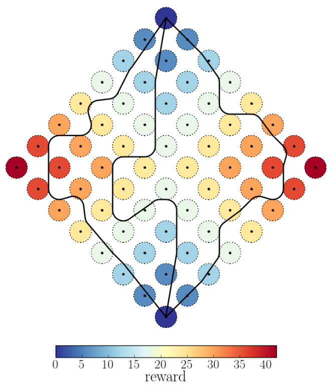
Dataset	$R_D = 2$	$R_D = 5$	$R_D = 6$	$R_D = 8$	$R_D = 10$	$R_D = 12$
p4.2.f	652	737	756	736	735	757
p4.3.j	771	837	818	901	871	848
p4.4.j	478	567	552	568	559	575
p5.2.n	930	1000	1030	1005	1010	1040
p5.4.p	740	820	815	830	840	850
p6.2.j	912	1104	1098	1176	1188	1206
p6.3.l	900	1044	1050	1104	1080	1146
p7.2.j	635	656	646	662	643	675
p7.3.i	439	459	458	466	465	475
p7.4.i	319	329	336	335	347	331
Average	678	755	756	778	774	790



(a) TOP trajectory of the p6.3.j dataset with total collected reward 804.



(b) DTOP trajectory of the p6.3.j dataset with total collected reward 762, $\rho = 0.5,\ R_D = 8$.



(c) DTOPN trajectory of the p6.3.j dataset with total collected reward 1008, $\rho = 0.5,\ \delta = 0.5,\ R_D = R_N = 8$.

Table 6: Execution times in seconds for different Dubins resolution parameters in the DTOPN.

Dataset	$R_D = 2$	$R_D = 5$	$R_D = 6$	$R_D = 8$	$R_D = 10$	$R_D = 12$
p4.2.f	294.97	1653.94	4273.44	6411.86	10805.97	5525.15
p4.3.j	119.36	451.43	259.58	1239.79	1489.73	377.90
p4.4.j	8.48	20.82	22.81	33.75	70.53	63.82
p5.2.n	291.94	745.01	2021.45	2116.62	2810.56	2124.74
p5.4.p	31.24	95.70	82.31	73.27	110.91	82.19
p6.2.j	238.19	2503.63	1939.44	3379.40	9704.51	3753.00
p6.3.l	146.50	235.95	378.42	271.47	503.55	706.72
p7.2.j	286.47	656.73	566.31	1231.43	1474.84	1065.48
p7.3.i	21.66	58.52	55.27	94.00	266.11	218.13
p7.4.i	1.95	6.79	7.39	13.75	36.52	36.26
Average	144.08	642.85	960.64	1486.53	2727.32	1395.34

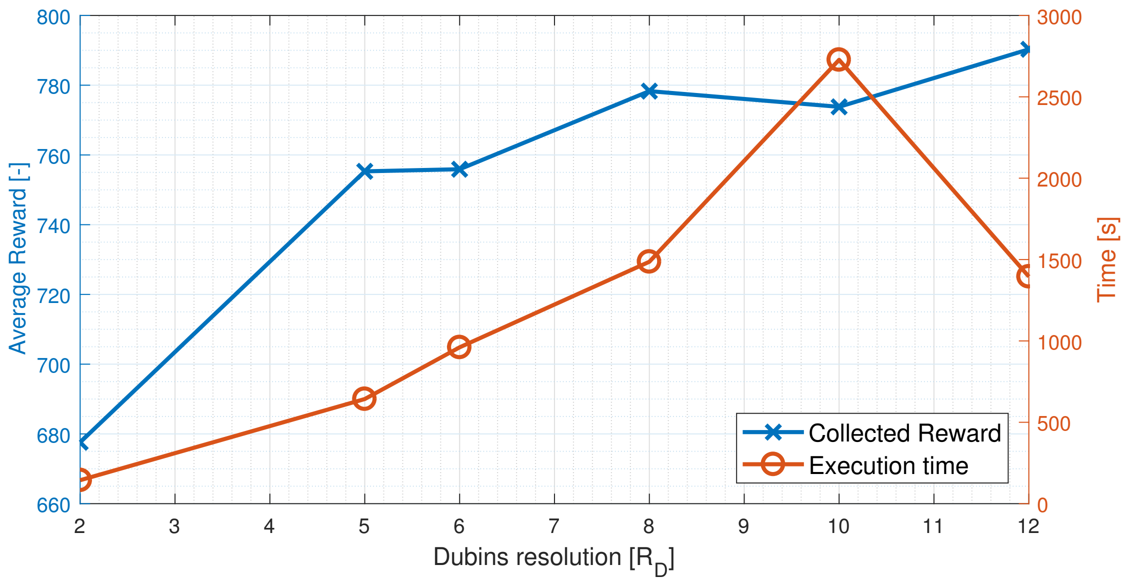


Figure 4: The dependency of the average collected reward and execution time of the SPR DTOPN solution on the Dubins resolution parameter R_D.

5.5 Neighbourhood Resolution Parameter

Similar to the Dubins resolution, the neighbourhood resolution parameter R_N specifies the preciseness of neighbourhood sampling. Fig. 5 shows that higher resolution increases the execution time, but also tends to increase the average collected reward. Results for different neighbourhood resolution parameters can be seen in Tables 7 and 8. The results are for the SPR with parameters Dubins resolution $R_D = 8$, Dubins radius $\rho = 0.5$ and neighbourhood radius $\delta = 0.5$.

Table 7: Collected reward for various neighbourhood resolution parameters in the DTOPN.

Dataset	$R_N = 2$	$R_N = 5$	$R_N = 6$	$R_N = 8$	$R_N = 10$	$R_N = 12$
p4.2.f	681	733	735	731	767	748
p4.3.j	785	863	824	885	852	831
p4.4.j	543	556	574	573	565	580
p5.2.n	940	1020	1010	1030	1000	1025
p5.4.p	795	825	840	845	850	840
p6.2.j	1026	1116	1122	1206	1230	1206
p6.3.l	978	1056	1074	1176	1182	1134
p6.4.j	450	474	438	480	462	486
p7.2.j	628	655	667	669	654	644
p7.3.i	441	453	457	469	471	468
p7.4.i	316	326	327	331	351	346
Average	689	734	733	763	762	755

Table 8: Execution time in seconds used for multiple neighbourhood resolution parameters in the DTOPN.

Dataset	$R_N = 2$	$R_N = 5$	$R_N = 6$	$R_N = 8$	$R_N = 10$	$R_N = 12$
p4.2.f	221.21	1790.93	4047.73	3702.72	7750.84	13369.86
p4.3.j	95.96	396.85	730.24	1353.88	957.98	3351.53
p4.4.j	9.45	20.20	582.55	66.67	99.71	168.82
p5.2.n	247.70	1234.80	1073.23	5216.12	2499.61	8669.91
p5.4.p	15.34	67.96	116.01	312.25	270.63	282.32
p6.2.j	153.22	2580.67	2617.67	9061.83	12244.12	7273.73
p6.3.l	65.42	211.78	189.04	987.64	1190.16	791.27
p6.4.j	3.12	13.24	23.25	74.11	42.58	48.60
p7.2.j	99.62	735.87	982.64	3675.43	3968.50	3053.61
p7.3.i	11.32	63.14	105.41	137.28	239.68	249.34
p7.4.i	1.99	13.11	17.74	24.62	36.50	133.95
Average	84.03	648.05	905.56	2237.50	2663.66	3399.36

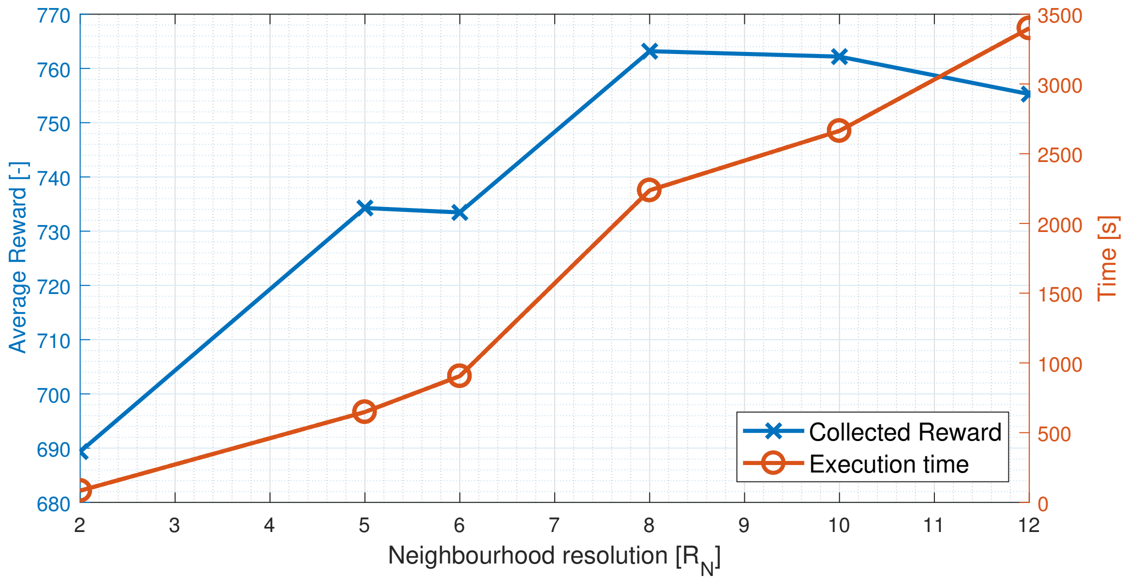


Figure 5: The dependency of the average collected reward and execution time of the SPR DTOPN solution on the neighbourhood resolution parameter R_N.

5.6 Dubins Radius Parameter

One of the important parameters that influence the results of the algorithm is the Dubins radius ρ. It represents the minimal turning radius of the Dubins vehicle and is usually determined by the parameters of the vehicle itself. For larger Dubins radii, the average collected reward is significantly lower, as seen in Fig. 6. The Dubins radius parameter does not seem to have a significant impact on the execution time.

Results for different Dubins radii ρ can be seen in Tables 9 and 10 showing the average collected rewards and computational times for a sample of datasets. The SPR mode of the algorithm was used with Dubins resolution $R_D = 8$, neighbourhood resolution $R_N = 8$ and neighbourhood radius $\delta = 0.5$.

Table 9: Collected rewards for various Dubins radii parameters ρ.

Dataset	$\rho = 0.1$	$\rho = 0.3$	$\rho = 0.5$	$\rho = 0.8$	$\rho = 1$
p4.3.j	873	870	875	848	810
p4.4.j	557	562	552	568	529
p5.4.p	910	905	875	880	885
p6.3.l	1218	1194	1110	1008	1032
p6.4.j	468	486	480	486	462
p7.2.j	652	658	648	646	664
p7.3.i	474	466	458	463	475
p7.4.i	352	355	318	316	319
Average	688	687	665	652	647

Table 10: Execution times in seconds for various Dubins radii parameters ρ.

Dataset	$\rho = 0.1$	$\rho = 0.3$	$\rho = 0.5$	$\rho = 0.8$	$\rho = 1$
p4.4.j	146.58	148.49	181.15	136.75	160.80
p5.4.p	262.94	264.46	212.31	419.12	516.08
p6.3.l	797.03	1202.80	998.29	1086.25	3057.83
p6.4.j	61.74	84.22	116.87	63.35	96.36
p7.2.j	9949.67	7082.09	3159.35	3815.03	8041.03
p7.3.i	402.02	314.71	82.58	512.73	444.57
p7.4.i	59.41	72.57	82.58	58.17	62.87
Average	1781.61	1694.72	807.09	1309.59	1867.04

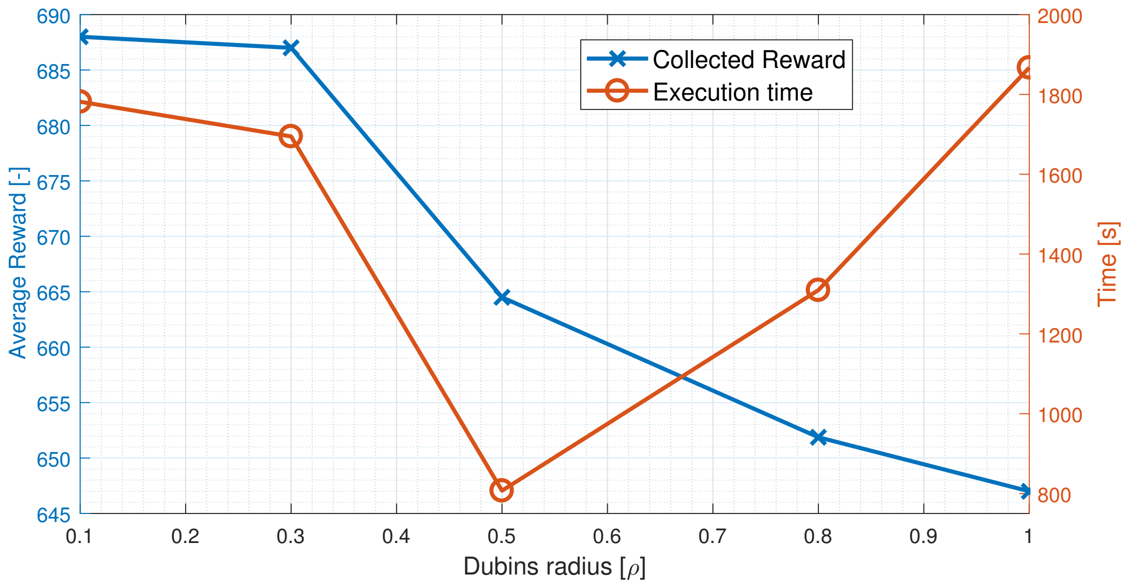


Figure 6: The dependency of the average collected reward and execution time of the SPR DTOPN solution on the Dubins radius parameter ρ.

5.7 Neighbourhood Radius Parameter

Similar to the Dubins radius, the neighbourhood radius δ has a large impact on the produced results. It acts opposite to the Dubins radius in the sense that the higher the radius, the higher the average reward collected, as seen on Fig. 7. It represents the radius (size) of the area around each location that must be visited in order to claim the associated reward. Results for sample datasets with different neighbourhood radii δ are presented in Tables 11 and 12. The results were obtained using SPR with parameters Dubins radius $\rho = 0.5$, Dubins resolution $R_D = 8$ and neighbourhood resolution $R_N = 8$.

Table 11: Collected reward for various neighbourhood radii parameters δ.

Dataset	$\delta = 0.1$	$\delta = 0.3$	$\delta = 0.5$	$\delta = 0.8$	$\delta = 1$
p4.4.j	518	535	552	631	643
p5.4.p	775	840	875	1030	1140
p6.3.l	978	1014	1110	1200	1308
p6.4.j	336	342	480	528	624
p7.2.j	612	640	648	681	711
p7.3.i	453	453	458	491	500
p7.4.i	291	330	318	365	379
Average	615	648	634	779	827

Table 12: Execution times in seconds for various neighbourhood radii parameters δ.

Dataset	$\delta = 0.1$	$\delta = 0.3$	$\delta = 0.5$	$\delta = 0.8$	$\delta = 1$
p4.4.j	133.26	125.14	181.15	368.44	245.79
p5.4.p	546.25	203.48	212.31	649.77	571.33
p6.3.l	6008.63	2372.39	998.29	1363.49	1166.95
p6.4.j	37.40	32.43	116.87	112.74	176.75
p7.2.j	2796.12	3651.98	3159.35	5975.68	7976.21
p7.4.i	49.18	66.25	82.58	89.69	60.03
p7.3.i	264.37	264.37	82.58	715.74	664.22
Average	1968.26	2286.62	690.45	2839.67	1965.20

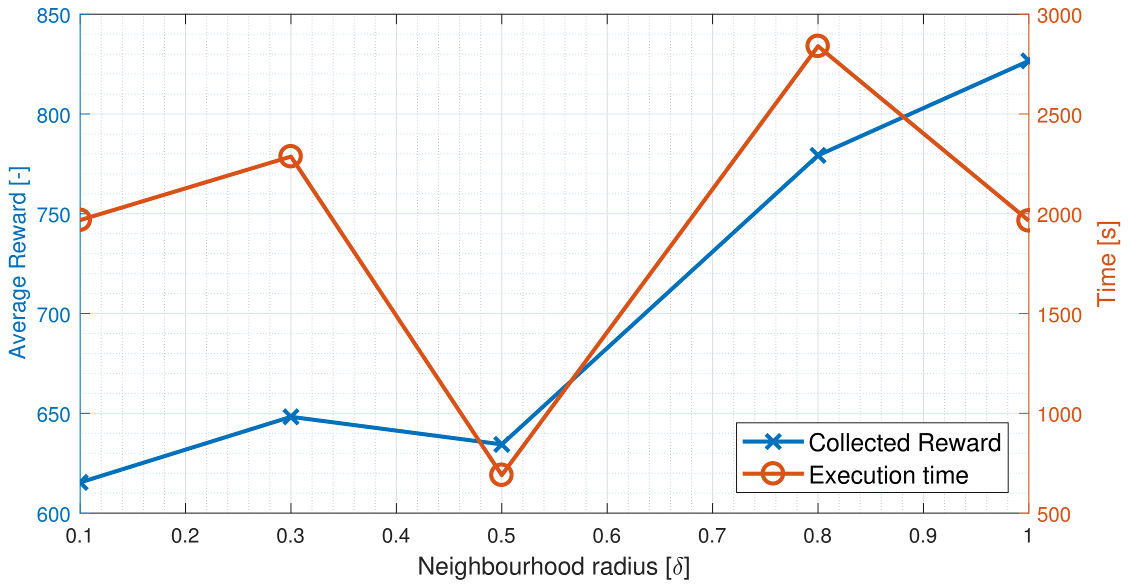


Figure 7: The dependency of the average collected reward and execution time of the SPR DTOPN solution on the neighbourhood radius parameter δ.

5.8 Experimental verification with Micro Aerial Vehicles

The GRASP-PR algorithm for the DTOPN was tested in real-life conditions using a group of three unmanned hexarotor MAVs, which were initially developed for multi-robot applications [48, 47, 53]. The MAVs were equipped with relative localisation system, Real Time Kinematic (RTK) GPS, onboard PC, autopilot and down-facing camera. Figure 8 shows the onboard equipment.

The relative localisation system is used by the onboard Collision Avoidance System (CAS) to prevent mid-air collisions within the group [29, 19]. They are also equipped with an onboard Model Predictive Controller (MPC) used for stabilisation using only onboard sensors [2]. RTK GPS provides an accurate reading of UAVs position, which is necessary for precision of pre-planned missions.

In the experiment, three different trajectories were prepared (TOP, DTOP and DTOPN), as shown in Fig. 11. They used locations represented by coloured blocks as seen in Figs. 9 and 10, each with its assigned reward, which were scattered on a field with an area of approximately 100×50 meters. The utilised T_{max} budget was 90 and it represented maximal travel distance for each UAV in meters.

The goal was to maximise the reward collected by visually inspecting the locations of interest using the onboard cameras as seen in Fig. 10. In the TOP instance, the MAVs could not inspect required locations because of their constant forward speed resulting in a non-zero minimal turning radius. In both the DTOP and the DTOPN, the MAVs could inspect the target locations because of the Dubins extension, but in the DTOPN case, the number of locations visited was higher.

In all of the instances, the actual trajectories of the MAVs were influenced by the onboard CAS, which predicted possible collisions on their routes despite the collision-free pre-planned trajectories. This resulted in inaccurate trajectory following since, because of the incorrectly predicted collisions, the MAVs were delayed at the start and moved with reduced speed, resulting in the MPC skipping trajectory waypoints in order to finish the flight on time.

Table 13: Reward collected in TOP, DTOP and DTOPN instances using FPR with parameters $T_{max} = 90$, $\rho = 6.15\ m, \delta = 4\ m,\ a_{max} = 2.6\ m \cdot s^{-2}$, $v_c = 4\ m \cdot s^{-1}$.

TOP	DTOP	DTOPN
78	67	83

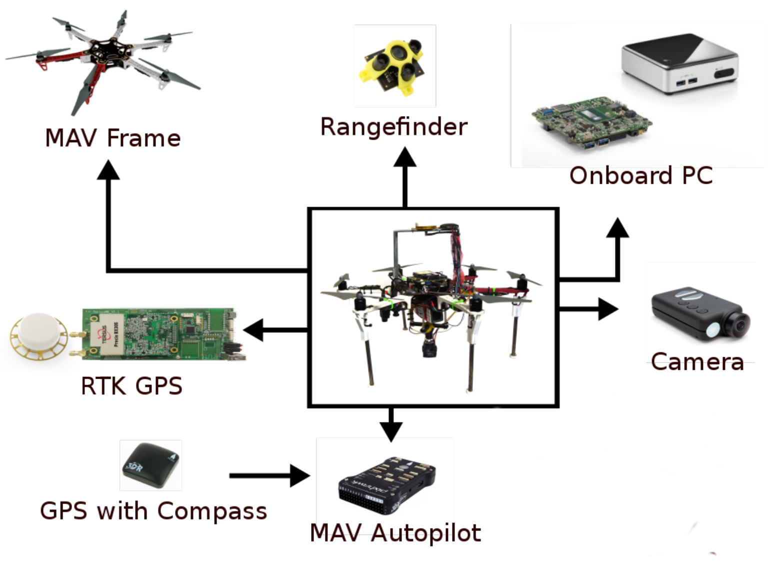


Figure 8: Visualisation of hexarotor UAV hardware used in the real-life experiment.

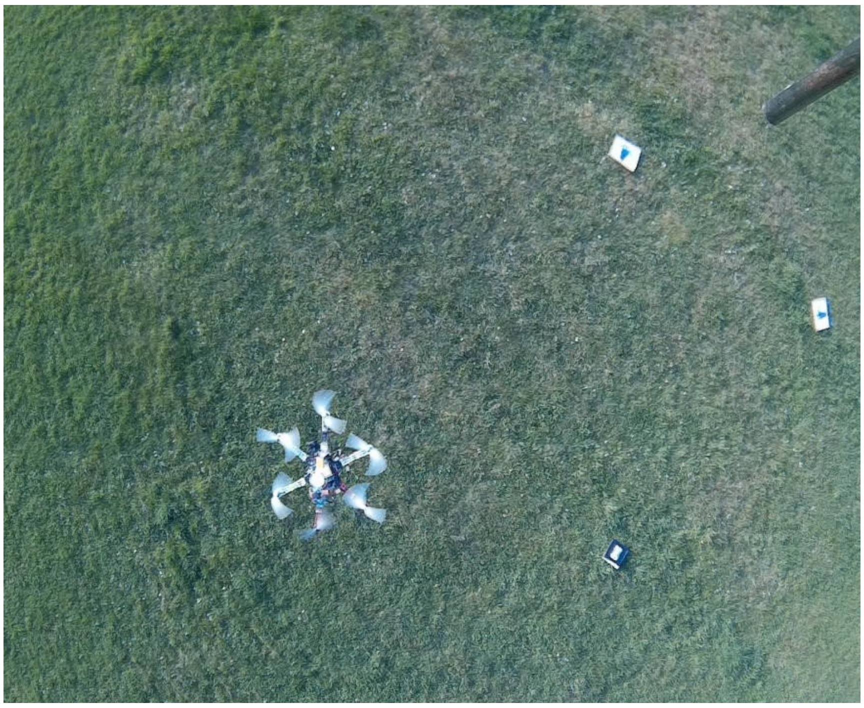


Figure 9: *UAV 1* hovering above its starting location (marked by a black block on the ground), taken by onboard camera of *UAV 2*.

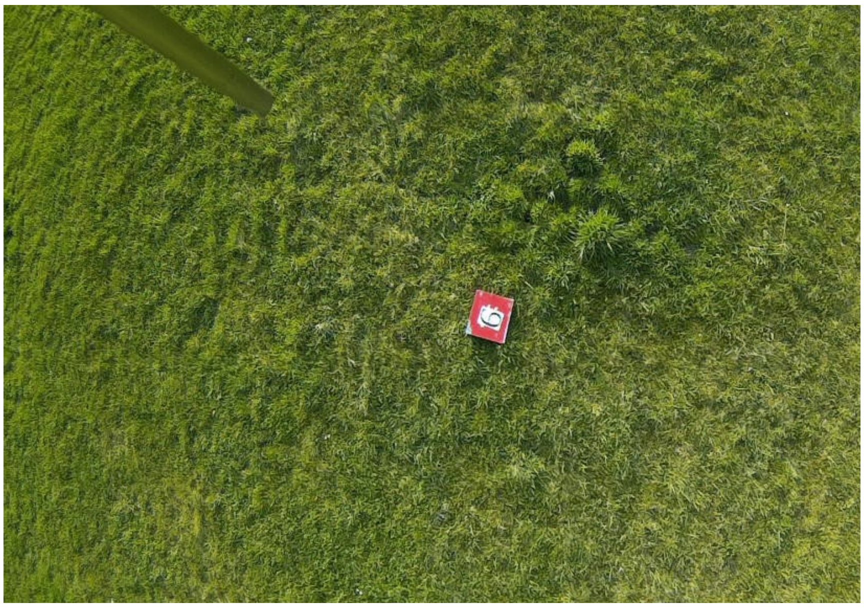


Figure 10: Image from onboard camera of *UAV 1* taken while inspecting a target location marked with a coloured block with a reward label.

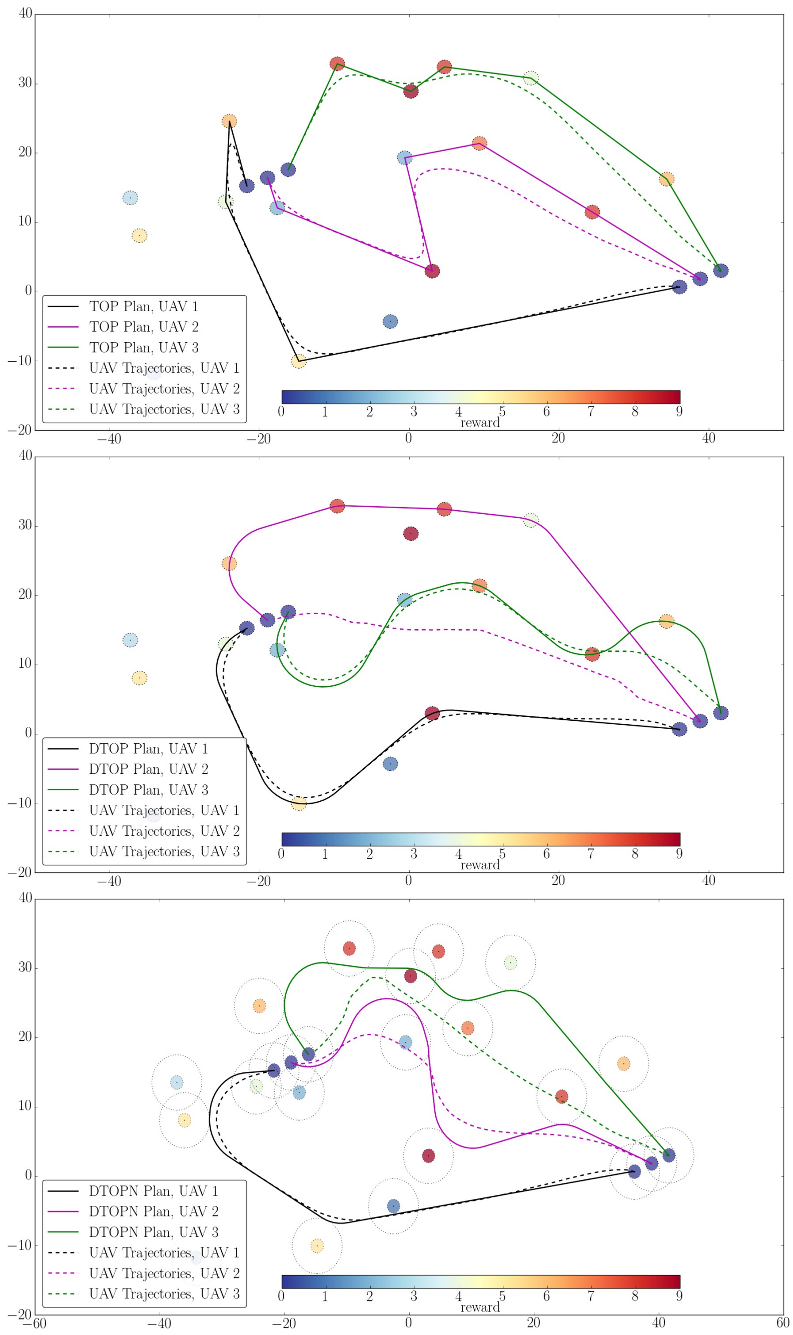


Figure 11: Three pre-planned trajectories for the TOP, DTOP and DTOPN with a visualisation of MAV trajectories flown in real-life experiment based on data from onboard positioning systems. The axes are shown in meters.

6 Conclusion

In this thesis, a Greedy Randomised Adaptive Search Procedure with Path Relinking was proposed as a solution to the Dubins Team Orienteering Problem with Neighbourhoods. The algorithm was based on GRASP-PR proposed by *W. Souffriau et al.*, originally for the TOP. It was extended to consider locations neighbourhoods and curvature constraints of the Dubins vehicle. This extension significantly increased the total collected reward while respecting the minimal turning radius of the vehicle in exchange for higher execution times.

Both the collected reward and computational time depend on four variables. The Dubins radius ρ is indirectly proportional to the average collected reward. This is due to larger travel costs when using arcs respecting the curvature constraints of the Dubins vehicle. The neighbourhood radius δ is directly proportional to the collected reward, since by allowing the vehicle to collect location's associated reward from it's close vicinity instead of the exact coordinates the travel distance is smaller, which lowers the travel costs. For higher Dubins and neighbourhood resolution of sampling, the average collected reward increases, since they increase the preciseness of the algorithm. However, they increase the average computational times.

The DTOPN is a useful as a problem formulation for data collecting scenarios allowing remote data collection involving multiple MAVs with limited travel budget and turning radius. The results of the algorithm were tested in real-life experiment modelling a data-collection scenario using multiple MAVs. The experiment, albeit influenced by the onboard collision avoidance system, proved that the MAVs could fly through the generated DTOPN trajectories while collecting higher reward than in DTOP, while in the TOP the MAVs missed the target locations.

References

[1] RD Angel, WL Caudle, R Noonan, and ANDA Whinston. Computer-assisted school bus scheduling. *Management Science*, 18(6):B–279, 1972.

[2] Tomas Baca, Giuseppe Loianno, and Martin Saska. Embedded model predictive control of unmanned micro aerial vehicles. In *Methods and Models in Automation and Robotics (MMAR), 2016 21st International Conference on*, pages 992–997. IEEE, 2016.

[3] Tolga Bektas. The multiple traveling salesman problem: an overview of formulations and solution procedures. *Omega*, 34(3):209–219, 2006.

[4] Hermann Bouly, Duc-Cuong Dang, and Aziz Moukrim. A memetic algorithm for the team orienteering problem. *4or*, 8(1):49–70, 2010.

[5] Sylvain Boussier, Dominique Feillet, and Michel Gendreau. An exact algorithm for team orienteering problems. *4or*, 5(3):211–230, 2007.

[6] Barry L Brumitt and Anthony Stentz. Dynamic mission planning for multiple mobile robots. In *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, volume 3, pages 2396–2401. IEEE, 1996.

[7] Barry L Brumitt and Anthony Stentz. Grammps: A generalized mission planner for multiple mobile robots in unstructured environments. In *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, volume 2, pages 1564–1571. IEEE, 1998.

[8] Steven E Butt and Tom M Cavalier. A heuristic for the multiple tour maximum collection problem. *Computers & Operations Research*, 21(1):101–111, 1994.

[9] Steven E Butt and David M Ryan. An optimal solution procedure for the multiple tour maximum collection problem using column generation. *Computers & Operations Research*, 26(4):427–441, 1999.

[10] R Wolfler Calvo and Roberto Cordone. A heuristic approach to the overnight security service problem. *Computers & Operations Research*, 30(9):1269–1287, 2003.

[11] Vicente Campos, Rafael Martí, Jesús Sánchez-Oro, and Abraham Duarte. Grasp with path relinking for the orienteering problem. *Journal of the Operational Research Society*, 65(12):1800–1813, 2014.

[12] I-Ming Chao, Bruce L Golden, and Edward A Wasil. A fast and effective heuristic for the orienteering problem. *European journal of operational research*, 88(3):475–489, 1996.

[13] I-Ming Chao, Bruce L. Golden, and Edward A. Wasil. The team orienteering problem. *European Journal of Operational Research*, 88(3):464 – 474, 1996.

[14] Duc-Cuong Dang, Racha El-Hajj, and Aziz Moukrim. A branch-and-cut algorithm for solving the team orienteering problem. In *International Conference on AI and OR Techniques in Constriant Programming for Combinatorial Optimization Problems*, pages 332–339. Springer, 2013.

[15] Duc-Cuong Dang, Rym Nesrine Guibadj, and Aziz Moukrim. An effective pso-inspired algorithm for the team orienteering problem. *European Journal of Operational Research*, 229(2):332–344, 2013.

[16] Marco Dorigo and Gianni Di Caro. Ant colony optimization: a new meta-heuristic. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, volume 2, pages 1470–1477. IEEE, 1999.

[17] Marco Dorigo and Luca Maria Gambardella. Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on evolutionary computation*, 1(1):53–66, 1997.

[18] 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*, 79(3):497–516, 1957.

[19] Jan Faigl, Tomáš Krajník, Jan Chudoba, Libor Přeučil, and Martin Saska. Low-cost embedded system for relative localization in robotic swarms. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 993–998. IEEE, 2013.

[20] Jan Faigl, Robert Pěnička, and Graeme Best. Self-organizing map-based solution for the orienteering problem with neighborhoods. In *Systems, Man, and Cybernetics (SMC), 2016 IEEE International Conference on*, pages 001315–001321. IEEE, 2016.

[21] Jan Faigl and Robert Pěnička. On close enough orienteering problem with dubins vehicle. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5646–5652, 2017.

[22] Jan Faigl, Petr Váňa, Martin Saska, Tomáš Báča, and Vojtěch Spurnỳ. On solution of the dubins touring problem. In *Mobile Robots (ECMR), 2017 European Conference on*, pages 1–6. IEEE, 2017.

[23] Matteo Fischetti, Juan Jose Salazar Gonzalez, and Paolo Toth. Solving the orienteering problem through branch-and-cut. *INFORMS Journal on Computing*, 10(2):133–148, 1998.

[24] Bruce L Golden, Larry Levy, and Rakesh Vohra. The orienteering problem. *Naval research logistics*, 34(3):307–318, 1987.

[25] Samuel Gorenstein. Printing press scheduling for multi-edition periodicals. *Management Science*, 16(6):B–373, 1970.

[26] 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.

[27] Richard M Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer, 1972.

[28] Liangjun Ke, Laipeng Zhai, Jing Li, and Felix TS Chan. Pareto mimic algorithm: An approach to the team orienteering problem. *Omega*, 61:155–166, 2016.

[29] Tomáš Krajník, Matías Nitsche, Jan Faigl, Petr Vaněk, Martin Saska, Libor Přeučil, Tom Duckett, and Marta Mejail. A practical multirobot localization system. *Journal of Intelligent & Robotic Systems*, 76(3-4):539–562, 2014.

[30] Gilbert Laporte and Silvano Martello. The selective travelling salesman problem. *Discrete applied mathematics*, 26(2-3):193–207, 1990.

[31] Jan Karel Lenstra and AHG Rinnooy Kan. Some simple applications of the travelling salesman problem. *Journal of the Operational Research Society*, 26(4):717–733, 1975.

[32] Yun-Chia Liang, Sadan Kulturel-Konak, and Min-Hua Lo. A multiple-level variable neighborhood search approach to the orienteering problem. *Journal of Industrial and Production Engineering*, 30(4):238–247, 2013.

[33] Yun-Chia Liang, Sadan Kulturel-Konak, and Alice E Smith. Meta heuristics for the orienteering problem. In *Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on*, volume 1, pages 384–389. IEEE, 2002.

[34] Shih-Wei Lin. Solving the team orienteering problem using effective multi-start simulated annealing. *Applied Soft Computing*, 13(2):1064–1073, 2013.

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

[36] Nenad Mladenović and Pierre Hansen. Variable neighborhood search. *Computers & operations research*, 24(11):1097–1100, 1997.

[37] 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.

[38] Karl Obermeyer. Path planning for a uav performing reconnaissance of static ground targets in terrain. In *AIAA Guidance, Navigation, and Control Conference*, page 5888, 2009.

[39] Karl Obermeyer, Paul Oberlin, and Swaroop Darbha. Sampling-based roadmap methods for a visual reconnaissance uav. In *AIAA Guidance, Navigation, and Control Conference*, page 7568, 2010.

[40] C Okonjo-Adigwe. An effective method of balancing the workload amongst salesmen. *Omega*, 16(2):159–163, 1988.

[41] Young-Man Park and Kap Hwan Kim. A scheduling method for berth and quay cranes. In *Container Terminals and Automated Transport Systems*, pages 159–181. Springer, 2005.

[42] Robert Pěnička, Jan Faigl, Petr Váňa, and Martin Saska. Dubins orienteering problem. *IEEE Robotics and Automation Letters*, 2(2):1210–1217, April 2017.

[43] Robert Pěnička, Jan Faigl, Petr Váňa, and Martin Saska. Dubins orienteering problem with neighborhoods. In *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1555–1562, June 2017.

[44] R Ramesh, Yong-Seok Yoon, and Mark H Karwan. An optimal algorithm for the orienteering tour problem. *ORSA Journal on Computing*, 4(2):155–165, 1992.

[45] Julia Robinson. On the hamiltonian game (a traveling salesman problem). Technical report, RAND PROJECT AIR FORCE ARLINGTON VA, 1949.

[46] Hussain Aziz Saleh and Rachid Chelouah. The design of the global navigation satellite system surveying networks using genetic algorithms. *Engineering Applications of Artificial Intelligence*, 17(1):111–122, 2004.

[47] M. Saska, T. Baca, J. Thomas, J. Chudoba, L. Preucil, T. Krajnik, J. Faigl, G. Loianno, and V. Kumar. System for deployment of groups of unmanned micro aerial vehicles in GPS-denied environments using onboard visual relative localization. *Autonomous Robots*, 41(4):919–944, 2017.

[48] Martin Saska, Vojtěch Vonásek, Tomáš Krajník, and Libor Přeučil. Coordination and navigation of heterogeneous mav–ugv formations localized by a ‘hawk-eye’-like approach under a model predictive control scheme. *The International Journal of Robotics Research*, 33(10):1393–1412, 2014.

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

[50] AİŞE ZÜLAL ŞEVKLİ and FATİH ERDOĞAN SEVİLGEN. Stpso: Strengthened particle swarm optimization. *Turkish Journal Of Electrical Engineering & Computer Sciences*, 18(6):1095–1114, 2010.

[51] Wouter Souffriau, Pieter Vansteenwegen, G Vanden Berghe, and DV Oudheusden. A greedy randomised adaptive search procedure for the team orienteering problem. In *EU/MEeting*, pages 23–24, 2008.

[52] Wouter Souffriau, Pieter Vansteenwegen, Joris Vertommen, Greet Vanden Berghe, and Dirk Van Oudheusden. A personalized tourist trip design algorithm for mobile tourist guides. *Applied Artificial Intelligence*, 22(10):964–985, 2008.

[53] Vojtech Spurny, Tomas Baca, and Martin Saska. Complex manoeuvres of heterogeneous mav-ugv formations using a model predictive control. In *Methods and Models in Automation and Robotics (MMAR), 2016 21st International Conference on*, pages 998–1003. IEEE, 2016.

[54] Joseph A Svestka and Vaughn E Huckfeldt. Computational experience with an m-salesman traveling salesman algorithm. *Management Science*, 19(7):790–799, 1973.

[55] Hao Tang and Elise Miller-Hooks. A tabu search heuristic for the team orienteering problem. *Computers & Operations Research*, 32(6):1379–1407, 2005.

[56] Lixin Tang, Jiyin Liu, Aiying Rong, and Zihou Yang. A multiple traveling salesman problem model for hot rolling scheduling in shanghai baoshan iron & steel complex. *European Journal of Operational Research*, 124(2):267–282, 2000.

[57] M Faith Tasgetiren. A genetic algorithm with an adaptive penalty function for the orienteering problem. *Journal of Economic & Social Research*, 4(2), 2002.

[58] Theodore Tsiligirides. Heuristic methods applied to orienteering. *Journal of the Operational Research Society*, 35(9):797–809, 1984.

[59] Gündüz Ulusoy. The fleet size and mix problem for capacitated arc routing. *European Journal of Operational Research*, 22(3):329–337, 1985.

[60] Pieter Vansteenwegen, Wouter Souffriau, Greet Vanden Berghe, and Dirk Van Oudheusden. A guided local search metaheuristic for the team orienteering problem. *European journal of operational research*, 196(1):118–127, 2009.

[61] Pieter Vansteenwegen, Wouter Souffriau, Greet Vanden Berghe, and Dirk Van Oudheusden. Metaheuristics for tourist trip planning. In *Metaheuristics in the service industry*, pages 15–31. Springer, 2009.

[62] Pieter Vansteenwegen, Wouter Souffriau, and Dirk Van Oudheusden. The orienteering problem: A survey. *European Journal of Operational Research*, 209(1):1–10, 2011.

[63] Pieter Vansteenwegen and Dirk Van Oudheusden. The mobile tourist guide: an or opportunity. *OR insight*, 20(3):21–27, 2007.

[64] Qiwen Wang, Xiaoyun Sun, Bruce L Golden, and Jiyou Jia. Using artificial neural networks to solve the orienteering problem. *Annals of Operations Research*, 61(1):111–120, 1995.

[65] Xia Wang, Bruce L Golden, and Edward A Wasil. Using a genetic algorithm to solve the generalized orienteering problem. In *The vehicle routing problem: latest advances and new challenges*, pages 263–274. Springer, 2008.

[66] Xin Yu and John Y Hung. A genetic algorithm for the dubins traveling salesman problem. In *Industrial Electronics (ISIE), 2012 IEEE International Symposium on*, pages 1256–1261. IEEE, 2012.

[67] 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.

Appendix A CD Content

In Table 14 are listed names of all root directories on CD.

Directory name	**Description**
thesis	the thesis in pdf format
thesis_sources	latex source codes
code_sources	GRASP-PR for DTOPN source codes

Table 14: CD Content

Appendix B List of abbreviations

In Table 15 are listed abbreviations used in this thesis.

Abbreviation	**Meaning**
MAV	Micro Aerial Vehicle
UAV	Unmanned Aerial Vehicle
MPC	Model Predictive Controller
CAS	Collision Avoidance System
OP	Orienteering Problem
TOP	Team Orienteering Problem
DOP	Dubins Orienteering Problem
DOPN	Dubins Orienteering Problem with Neighbourhoods
DTOP	Dubins Team Orienteering Problem
DTOPN	Dubins Team Orienteering Problem with Neighbourhoods
TSP	Travelling Salesman Problem
TSPN	Travelling Salesman Problem with Neighbourhoods
mTSP	Multiple Travelling Salesmen Problem
DTSP	Dubins Travelling Salesman Problem
STSP	Selective Travelling Salesman Problem
GRASP	Greedy Randomised Search Procedure
PR	Path Relinking
FPR	Fast Path Relinking
SPR	Slow Path Relinking
VNS	Variable Neighbourhood Search
GA	Genetic Algorithm
SOM	Self-Organizing Map

Table 15: Lists of abbreviations