Modeling, Identification and Control, Vol. 38, No. 1, 2017, pp. 21–32, ISSN 1890–1328

UAV Path Planning using MILP with Experiments

A. Albert [1] F.S. Leira [1] L. Imsland [1]

[1] *Department of Engineering Cybernetics, Norwegian University of Science and Technology, N-7491 Trondheim, Norway. E-mail:* {anders.albert, frederik.s.leira, lars.imsland}@itk.ntnu.no

Abstract

In this paper, we look at the problem of tracking icebergs using multiple Unmanned Aerial Vehicles (UAVs). Our solutions use combinatorial optimization for UAV path planning by formulating a mixed integer linear programing (MILP) optimization problem. To demonstrate the approach, we present both a simulation and a practical experiment. The simulation demonstrates the possibilities of the MILP algorithm by constructing a case where three UAVs help a boat make a safe passage through an area with icebergs. Furthermore, we compare the performance of three against a single UAV. In the practical experiment, we take the first step towards full-scale experiments. We run the algorithm on a ground station and use it to set the path for a UAV tracking five simulated icebergs.

Keywords: UAV, Path Planning, Mathematical Optimization

1 Introduction

Offshore operations in ice-infested arctic areas demand ice management. Ice management is all activities that aim to reduce or avoid the impact of any kind of ice features. Several authors have argued that Unmanned Aerial Vehicles (UAVs) are an efficient platform to perform detection and surveillance of ice features for ice management purposes, e.g. Eik (2008), Lesinskis and Pavlovics (2011).

Using mobile sensors, like a UAV, for information gathering is a popular research topic. One of the reasons for its popularity is the many applications ranging from inspections of power lines, ships, pipelines etc., monitoring of traffic and environment, to border patrol, police support, surveillance and reconnaissance and filming for the entertainment industry (Valavanis and Vachtsevanos, 2015).

In this paper, we propose to use UAVs for iceberg tracking, specifically our contribution is an optimization-based path-planning framework for this purpose. There are several different ways to approach UAV path planning using mathematical optimization. For example, the problem can be formulated as a continuous time optimal control problem. Then, by exploiting well-known techniques like single or multiple shooting the problem can be transformed into a large scale nonlinear programming problem, for which there exist many commercially available solvers. Two examples of this approach are Haugen and Imsland (2013) and Walton et al. (2014).

The framework we propose in this paper is based on combinatorial optimization. It is similar to the Traveling Salesperson Problem (TSP), which is to find the shortest path visiting each city in a given list of cities exactly once. A generalization of TSP is mixed integer linear programming (MILP). For information about modern MILP solvers and problems in general see e.g. Jünger (2009), Bixby (2002), and Chen et al. (2010). In this paper, we use the CPLEX solver from IBM (ibm, 2015).

There are multiple applications similar to iceberg tracking with UAVs where TSP formulations have been applied. We have the problem of doing surveillance with unmanned ground vehicles (UGSes) in combination with a UAV. The UGSes have good sensing, but poor communication capabilities compared to the UAV. The UAV collects information by visiting UGSes.

doi:10.4173/mic.2017.1.3

In Barton and Kingston (2013), the authors compare a TSP solution to an adaptive feedforward iterative learning control algorithm, based on the region of attraction for each UGS. Another similar problem is the heterogeneous, multiple depot, multiple UAV routing problem (HMDMURP). This is a generalization of TSP, where the salesmen are UAVs with different minimum turning radius and starting locations. The authors of Oberlin et al. (2010) come up with a approximate algorithm based on the transformation by Noon and Bean (1993). A related problem to HMDMURP is studied by Oh et al. (2015), which come up with a solution based on a modified algorithm for the Chinese Postman Problem to make it suitable for a group of Dubins Vehicles. The article Enright et al. (2005) focuses on the repairman problem, which is TSP with targets appearing according to a Poisson process. The authors calculate lower and upper bounds for a Dubins path and use these to construct a centralized planner to assign areas for a set of UAVs.

Furthermore, there are a number of approaches using MILP that does not use TSP as a basis for problem formulation, which are similar to iceberg tracking with UAVs. The problem of searching and tracking targets in an urban environment using fixed wing UAVs is tackled by Hirsch and Schroeder (2015). The authors formulate the problem mathematically as a MILP, and solve it using an approximate method consisting of a greedy randomized adaptive search procedure and a simulated annealing. The military application of assigning targets of different priority and path planning to combat UAVs is studied by Shetty et al. (2008). They also formulate the problem as a MILP and compare the CPLEX-algorithm (ibm, 2015) to an approximate approach consisting of a tabu search algorithm with regard to optimally and computational efficiency. Schouwenaars et al. (2001) and Ma and Miller (2005) use the CPLEX-algorithm to solve a path planning problem formulated as a MILP for a single or multiple UAVs.

Finally, there are many formulations similar to iceberg tracking that are not solved using a TSP framework. In Sinha et al. (2005) the authors consider tracking hostile targets moving in group using multiple UAVs. Their solution is an adaptive decentralized algorithm. This is similar to the radar resources distribution and combat management problem, which are solved with a multi-level tree algorithm in Asnis and Blackman (2011). The authors of Farmani et al. (2015) track moving targets in an urban environment, where they take into account UAV and gimbal poses. They also solve the problem decentralized by using an auction method. The path planning for each UAV is done with MPC (Model Predictive Control).

We choose to not consider gimbal pose, nonholonomic constraints (like the Dubins vehicle), decentralization, etc. for the benefit of a simpler formulation, similar to TSP, called the target visitation problem (Grundel and Jeffcoat, 2004) (each iceberg has a value and high values get prioritized for an earlier visit). The reason is that we expect to have communication link with all UAVs and to be covering a vast area, where dynamics like gimbal pose and non-holistic constraints will be small compared to the Euclidean distance between the icebergs. This enables us to solve the problem fast for a limited number of UAVs and icebergs, which then enables real-time implementation. We take into account the dynamics of the problem by implementing the solution similar to an MPC, meaning that we solve a static optimization problem often and update the dynamics of the UAVs and iceberg between each time we run the optimization.

1.1 Contribution

The contribution of this paper is a framework for monitoring moving targets. The planning is centralized and the optimization formulation allows for the use of multiple UAVs. The framework extends earlier work (Albert and Imsland, 2015). We compare the improved approach using three UAVs to one UAV. In addition, we demonstrate the first step towards practical experiments with a test performed in Ny-Ålesund at Svalbard.

1.2 Organization

This paper is arranged as follows. In Section 2, we introduce the problem formulation through a scenario. In Section 3 we explain the setup, the modeling of the icebergs and UAVs, and the observer we use. Then, we introduce some assumptions in order to use the target visitation formulation on our problem, and describe how we formulate the problem using mixed integer linear programming in Section 4.

To demonstrate the use of multiple UAVs and the possibilities of the algorithm, we introduce a scenario with a boat moving through an iceberg infested area. We perform a simulation of the scenario, compare it to a single UAV, and illustrate the results in Section 5. In Section 6, we present the experimental results from Ny-Ålesund at Svalbard. Finally, in Section 7 and 8 we discuss the results and conclude the work of this paper.

2 Problem Formulation

To motivate the problem formulation we consider a concrete case: A boat traveling in an arctic area, where there are drifting icebergs. The boat must avoid collisions with icebergs. Monitoring the surrounding icebergs is necessary for safe operation. One might consider using satellite images for this task. However, in arctic areas the update frequency is often too slow and image resolution is too low to provide sufficient warning for the operations (Eik, 2008). Another solution might be to use marine radar, but we assume that the range of a marine radar is insufficient for detecting icebergs early enough for this boat to be able to maneuver safely and efficiently. An iceberg on collision course requires time to take appropriate action. In addition, marine radars suffer from performance degradation due to poor weather conditions such as rain and high waves (Eik, 2008). An unmanned aerial vehicle (UAV) with a sensor capable of automatic (or manual) detection of icebergs can be flexible, cheap (compared to manned flights), efficient, and with better coverage than marine radar and has increased spatial and temporal resolution compared to satellites.

The UAV can, for example, be equipped with an optical camera to detect icebergs. This camera will have a limited field of view (FOV), which only makes it possible to observe icebergs in a limited area, at the same time. To monitor a larger area the UAV must move around. A fixed wing UAV is best suited for this purpose, since it can cover relatively large distances.

We want to use multiple UAVs, so we can increase the area coverage. Our task is to make a path planner for each UAV to do continuous tracking of all icebergs in a surrounding area.

The scenario is illustrated in Figure 1. In the scenario, the boat (the yellow polygon) needs to move through an area with icebergs (blue squares). It has three UAV available with a limited field of view, which is illustrated by a light yellow circle. In the drawing, two of the UAVs are currently observing an iceberg each.

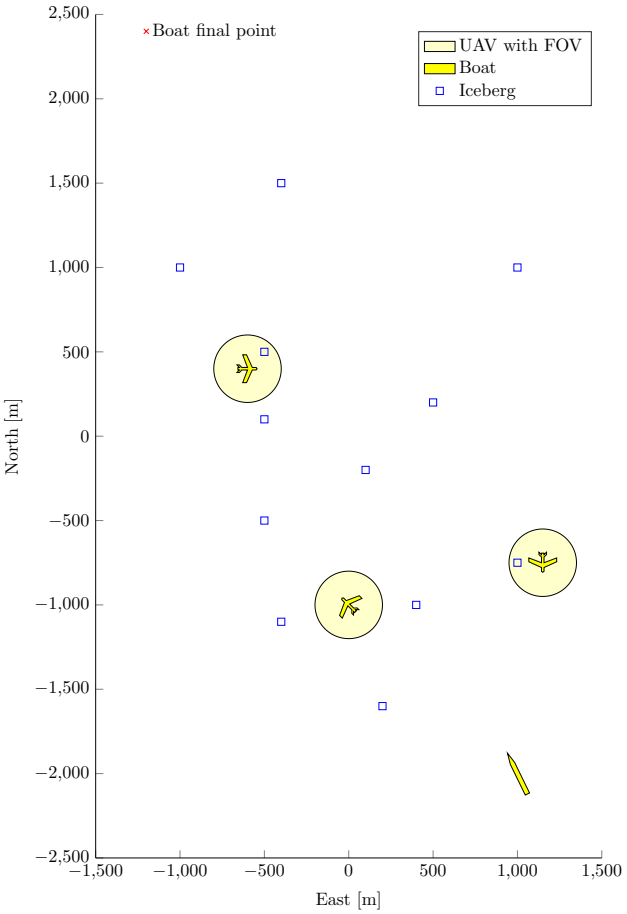

Figure 1: Illustrated scenario

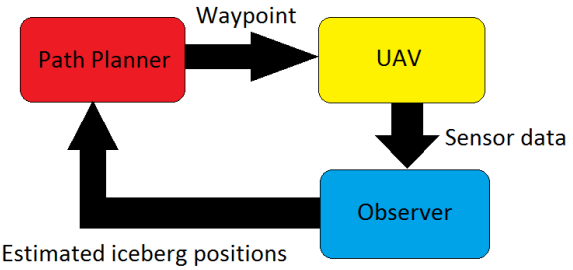

Figure 2: System components.

3 System Overview and Modeling

The system will consist of three components. The setup is illustrated in Figure 2. First, a set of physical UAVs, each with an autopilot capable of following waypoints. In addition, each UAV has a sensor able to detect icebergs. Second, the sensor data is sent to an observer to estimate iceberg position and velocity. Finally, the path planner uses the iceberg positions and velocities to find a path to track icebergs. The path planner then sends waypoints to each UAV. The focus of this paper is the path planner.

We model the UAVs as Dubins vehicles,

$$\dot{z}_i = \begin{bmatrix} \dot{x}_i \\ \dot{y}_i \\ \dot{\psi}_i \end{bmatrix} = \begin{bmatrix} U\cos(\psi_i) \\ U\sin(\psi_i) \\ u_i \end{bmatrix} \quad \forall i \in [1, \ldots, n_{\text{UAVs}}] \quad (1)$$

where ψ_i is the heading, u_i is the bank angle, x_i and y_i are the Cartesian position, and U is the velocity of each vehicle (all the vehicles are assumed to move with the

same velocity). Note that we assume constant altitude with this model.

We assume that the real position and velocity of each iceberg is governed by:

$$\dot{\xi}_i = \begin{bmatrix} \dot{s}_i \\ \dot{v}_i \end{bmatrix} = \begin{bmatrix} 0 & I \\ 0 & 0 \end{bmatrix} \xi_i + \begin{bmatrix} 0 \\ 1 \end{bmatrix} w_i(t) \quad \forall i \in [1, \ldots, n_{\text{icebergs}}] \tag{2}$$

where ξ_i is the four dimensional iceberg state consisting of position s_i and velocity v_i, both of dimension $\mathbb{R}^2$. The process noise is $w_i(t) \sim (0, q_i)$, which we assume has a Gaussian distribution with a mean of zero and variance of q_i.

To estimate the positions of the icebergs for the path planner we use a discrete Kalman filter. The discrete equation and measurement model for each iceberg using Euler integration, is (for clarity we lose the subscript i for each iceberg)

$$\hat{\xi}_{k+1} = \begin{bmatrix} I & \Delta T \\ 0 & I \end{bmatrix} \hat{\xi}_k + \begin{bmatrix} 0 \\ \Delta T \end{bmatrix} w_k = A\hat{\xi}_k + \begin{bmatrix} 0 \\ \Delta T \end{bmatrix} w_k \tag{3}$$

$$y_k = \begin{bmatrix} I & 0 \end{bmatrix} \hat{\xi}_k + v_k = C\hat{\xi}_k + v_k \tag{4}$$

here $\hat{\xi}_k$ and $\hat{\xi}_{k+1}$ are the estimated state for the current and the next time step, y_k is the measured position of the iceberg and ΔT is the time step. The measurement noise, v_k, is also assumed to have a Gaussian distribution with a mean of zero and variance of R.

A Kalman filter tracks the estimated state of the system and updates an associated error covariance matrix. The update process of the filter consists of two steps. The first step is known as the a priori step, where the state is estimated based on the model and the error covariance matrix is updated with the model and process uncertainty. In the second step, known as the posteriori step, the measurement is incorporated to the estimated state and the error covariance matrix is reduced. For our case, the posteriori step will only be included when a UAV is observing the relevant iceberg, while the priori step will be updated each time step.

The Kalman equation for the a priori step will be

$$\hat{\xi}_{k+1}^{\text{priori}} = A\hat{\xi}_k^{post} \qquad \hat{\xi}_0 = \hat{\xi}_0 \tag{5}$$

$$P_{k+1}^{\text{priori}} = AP_k^{post}A^T + Q \quad P_0 = Q \tag{6}$$

here P is the error covariance matrix.

When a measurement is available, the Kalman filter performs the posteriori step

$$K = P_{k+1}^{\text{priori}} C^T / (CP_{k+1}^{\text{priori}} C^T + R) \tag{7}$$

$$\xi_{k+1}^{\text{post}} = \hat{\xi}_{k+1}^{\text{priori}} + K(y_k - C\hat{\xi}_{k+1}^{\text{priori}}) \tag{8}$$

$$P_{k+1}^{\text{post}} = P_{k+1}^{\text{priori}} - KCP_{k+1}^{\text{priori}} \tag{9}$$

here K is known as the Kalman gain. If no measurement is available $K = 0$ (the posteriori value equals the priori value)

We can now define the position uncertainty using the error covariance matrix

$$\sigma = \text{tr}(p_{11}) \tag{10}$$

where

$$P = \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix} \tag{11}$$

Remark: If we select q_i in equation (2) different from the process noise, it will function as a heuristic to assign/manipulate priorities to the icebergs. Then, $\sigma(t)$ will represent priority instead of uncertainty. This heuristic can be set different for each iceberg, and can vary with time, $q(t)$. For example, the path planner can set the initial priority, σ_0, from satellite imagery and the rate of change, $q(t)$, based on distance from a moving boat.

4 Problem Setup and MILP Formulation

We assume we have multiple UAVs to track icebergs in an area. Each UAV is capable of following a given sequence of waypoints. This motivates us to exploit mixed integer linear programming (MILP) to find an optimal sequence of icebergs to visit for each UAV. MILP problems are optimization problems that can contain integer variables in objective function and/or constraints.

4.1 Assumptions

We want a path planner capable of real-time implementation and thus want to use a mixed-integer framework where the constraints and objective function are linear. To avoid nonlinearities, we make the following two assumptions:

1. The UAVs can follow any path.
2. Icebergs are stationary within the horizon considered in the optimization.

Assumption 1 means that we do not take the UAV equation (1) into consideration in the path planning, except for the distance from the initial position of the UAV to the first iceberg. This will inevitably lead to a suboptimal solution, since if we also consider the movement constraints of the UAV another visitation sequence might be better. However, the difference between the two visitation sequences will be small if the area is large compared to the turning radius of the

UAV. In addition, the autopilot will manage to pilot the UAV to any waypoint even though it is not selected according to its dynamics. The second assumption enables us to solve the problem more efficiently. We can make this assumption since UAVs in general move much faster than icebergs (typically 0.1 m/s for icebergs against 22-25 m/s for UAVs). To take into account the slow movement of the icebergs we plan to rerun the optimization either at fixed intervals (sample-based), or every time a UAV observes an iceberg (event-based). Before we rerun the optimization, we update the iceberg positions and position uncertainties with the model from equation (5) and (10), and if available UAV observations and satellite imagery.

We also make some additional assumptions. First, we assume a constant number of icebergs that are known a priori. Icebergs can easily be added or subtracted in between optimization runs. Second, we assume perfect communication between the UAVs and a ground station. The ground station can perform the optimization and instantly communicate the result to each UAV. Finally, we do not consider the size of the icebergs, as we are only concerned with their location.

We consider anti-collision or fuel constraints for the optimization to be out of scope for this article. Anti-collision can be solved by a lower level controller like the waypoint following controller on each UAV or the UAVs can just fly at different altitude. If we have fuel constraints this can be implemented through keeping track of the distance from base and compare it to remaining fuel. When the fuel gets low the UAV can be set to return to base and removed from the optimization problem.

4.2 Optimization variables

Now, we can formulate our problem using MILP. Our approach to formulating the optimization problem in a MILP framework uses the formulation for TSP in Miller et al. (1960) as a basis. Furthermore, we only use the subclass ILP (integer linear programming) of MILP, since we will only use integer and binary variables. The problem will have N nodes, which is the sum of UAVs (n_{UAV}) and icebergs (n_{iceberg}), $N = n_{\text{UAV}} + n_{\text{iceberg}}$. The optimization variables are organized in a matrix and a vector. First, we have a matrix of binary variables $y_{\text{path}} \in \mathbb{Z}_{\{0,1\}}^{N\times N}$. The entry $y_{\text{path}}(i,j)$ represents the path from node i to j. A node is an iceberg or a UAV. It is one if a UAV moves between the nodes and zero otherwise. Second, each UAV has an appurtenant sequence. In each sequence, the UAV is the first entry with the remaining part of the sequence consisting of icebergs. An integer vector, $t \in \mathbb{Z}^{N\times 1}$, represents the number each node has in its sequence. For example, if we have two UAVs, named UAV_1 and UAV_2, and five icebergs, named ice_1 to ice_5, then $t = [\text{UAV}_1, \text{UAV}_2, \text{ice}_1, \text{ice}_2, \text{ice}_3, \text{ice}_4, \text{ice}_5]^T$. Suppose the optimal solution is that the first UAV, UAV_1, visits iceberg ice_2, ice_5 and then ice_1, while the second UAV, UAV_2, visits ice_3 before ice_4. Then we will get the following $t = [1, 1, 4, 2, 2, 3, 3]^T$. Notice that the sequences for each UAV are mixed together in the vector t. We use the binary matrix y_{path} to assign icebergs to each UAV. The t-vector is used to avoid Hamiltonian subpaths, and to include position uncertainty in the objective function.

4.3 Constraints

Here, we will find a set of constraints that describe the set of feasible paths in terms of the optimization variables. First, each node cannot be visited and left more than once

$$\sum_{i}^{N} y_{\text{path}}(i,j) \leq 1 \quad \forall j \quad \text{and} \tag{12a}$$

$$\sum_{j}^{N} y_{\text{path}}(i,j) \leq 1 \quad \forall i. \tag{12b}$$

Next, the number of total paths between the nodes is equal to the number of nodes minus the number of UAVs,

$$N - n_{\text{UAV}} = \sum_{i=1}^{N}\sum_{j=1}^{N} y_{\text{path}}(i,j). \tag{13}$$

The t-variable is an integer vector that decides the sequence each UAV will visit icebergs, as explained above. The values of the vector must be within the number of nodes,

$$1 \leq t(i) \leq N \quad \forall i \in [1, 2, \ldots, n_{\text{UAV}}]. \tag{14}$$

The same integer vector t is used to represent the visitation sequence for each UAV. For example, if we have two UAVs in our problem, their sequence will be mixed together in the same vector. This is unproblematic since we use the binary matrix y_{path} to set the paths between UAVs and icebergs. We use the t-vector to prioritize high uncertainty icebergs in the objective function and to avoid subcycles. To make sure each UAV is the first of its sequence the first n_{UAV} elements in the t-vector representing the UAVs must be one,

$$t(i) = 1 \quad \forall i \in [1, 2, \ldots, n_{\text{UAV}}]. \tag{15}$$

Finally, each path must be connected. To avoid subcycles we need an additional constraint. The following constraint, called the Miller-Tucker-Zemlin constraint

(Chen et al., 2010), makes sure that all the paths start with a UAV and are connected:

$$t(j) - (t(i) + 1) \geq -N(1 - y_{\text{path}}(i, j)) \quad \forall i \neq j. \quad (16)$$

4.4 Optimization Problem

We can now formulate the following optimization problem:

$$\min F(y_{\text{path}}, t) = -(1 - \tau)F_1 + \tau F_2 \quad (17a)$$
$$\text{s.t. } (12), (13), (14), (15), \text{ and } (16)$$

where

$$F_1 = \sum_{i=1}^{N} \sigma(i)(N - t(i)) \quad (17b)$$

$$F_2 = \mu \sum_{i}^{N} \sum_{j}^{N} y_{\text{path}}(i, j) d(i, j). \quad (17c)$$

Here, μ is a scaling variable to make F_1 and F_2 of comparable size, see Section 4.5, and τ is a tuning constant used to weight between the two objectives. Setting $\tau = 1$ puts all weight on shortest distance and $\tau = 0$ puts all weight on position uncertainty. In the simulation and practical experiments of this paper we found through experience the value $\tau = 0.5$ to be appropriate. In the objective function, we have two competing objectives. First, minimizing $-F_1$ equals sorting the icebergs in sequences based on their position uncertainty. Here, $\sigma(i)$ is the position uncertainty of each iceberg and UAV. The UAVs will have a position uncertainty of zero. The position uncertainty is constant when running the optimization. In between optimization runs it is updated with equation (10), which means it increases linearly with time for unobserved icebergs and decreased towards zero for observed icebergs. Second, F_2 contains the distance traveled by the UAVs. The matrix d contains the distances between all the nodes, which is recalculated for each iteration of the optimization problem. The distances are Euclidean distances except the distance from the UAVs to the icebergs. To calculate the distance from a UAV with a given heading to each iceberg we use Dubins paths without fixed final heading. A Dubins path is a curve with a minimum turning radius and a fixed initial and final heading. This way we take the nonholonomic dynamics from equation (1) of the UAVs into consideration for the path from each UAV to the first iceberg in each sequence.

4.5 Scaling

To get a proper trade-off of the objectives F_1 and F_2 in equation (17a) we need to scale them by choosing an appropriate value for μ. If we knew the optimal values of F_1 and F_2 in advance of the optimization, a natural choice for μ would be the ratio between them. Instead we approximate this ratio by calculating the maximum value for F_1 and an estimated average value for F_2. The maximum value for F_1 is found by optimizing without constraints, which is easy and efficient to do. We cannot do the same for F_2, since the minimal F_2 will always be zero. Instead we take the d-matrix and calculate the average distance and multiply it by the number of paths we need in our solution, $F_{2,\text{avg}} = d_{\text{avg}}(N - n_{\text{UAV}})$. We can now choose:

$$\mu = \frac{F_{1,\max}}{F_{2,\text{avg}}}. \quad (18)$$

If we compare the μ from equation (18) with the ratio of F_1 and F_2 after we run the optimization in 1000 simulations, the guessed ratio from equation (18) has a mean of 89% and a standard deviation of 13%.

4.6 Implementation aspects

In the implementation of the optimization algorithm there are two adjustments. First, to avoid revisiting newly visited icebergs, the average position uncertainty of all the icebergs is calculated before the optimization. Then, only icebergs with a position uncertainty above 10 % of the average are included in the optimization. Second, the optimization can lead to a UAV not being assigned to track any iceberg. In this case, the UAV is set to track the closest iceberg (calculated in Dubins distance without the final heading).

5 Simulation

To implement the optimization algorithm from the previous section we use MATLAB R2014b with the toolbox YALMIP (Löfberg, 2004). YALMIP enables easy implementation of optimization problems. Furthermore, we use the CPLEX solver from IBM (ibm, 2015).

To illustrate the possibilities of the optimization algorithm we construct a simulation case with a boat moving through an area with icebergs. We compare using three UAVs to a single UAV for monitoring icebergs. When the boat enters the area we know the position and velocity of the 10 icebergs. This is an unrealistic assumption. However, it is necessary since we have chosen to set the iceberg velocities higher than normal (see next paragraph). The UAV(s) launch from the boat. In our case, the boat is only included for illustrative purposes, so its path is hard-coded.

In Table 1, the simulation parameters are given. The optimization of the UAV-paths is implemented sample-based, meaning that the optimizations are run using

fixed time intervals. As mentioned in section 4.1, icebergs typically move at a velocity of 0.1 m/s. In the simulation, we have chosen to set the iceberg velocity up till about 3 m/s. The reason is that we want to better illustrate the tracking capabilities of the algorithm, since it is also suitable to other tracking applications than icebergs.

Two snapshots from each simulation are shown in Figure 3. Here the boat is illustrated by a yellow polygon. The UAVs have a solid line for their recent movement and a dotted line for the sequence they plan to visit the icebergs. Each iceberg has a solid blue line for their recent movement and a blue "x" for their current position. The icebergs are also enumerated. A red circle indicates the observers estimated position of each iceberg.

In Figure 4, we see the average position uncertainty during the simulations compared for one and three UAVs. The reason the average position uncertainty falls somewhat at the end of the simulation for the single UAV, is that it stops tracking some of the icebergs it cannot find.

Table 1: Simulation Parameters

Parameter	Value	Unit
Boat	1 unit	
$(x_0,y_0$,heading)	(1500,0,2.0246)	(m,m,rad)
Velocity	10	m/s
Min turning radius	587	m
UAVs	3 units	
$(x_0,y_0$,heading)	(1500,0,0) $(1500,0,\frac{\pi}{2})$ $(1500,0,\pi)$	(m,m,rad)
Velocity	22	m/s
Minimum turning radius	105	m
FOV	150	m
Icebergs	10 units	
x_0	$\in$[0,1500]	m
y_0	$\in$[0,1500]	m
v_x	$\in$[-3,3]	m/s
v_y	$\in$[-3,3]	m/s
Observer		
Time step, ΔT	0.1	s
Process noise, Q	$\begin{bmatrix} 0_{2\times2} & 0_{2\times2} \\ 0_{2\times2} & 0.01I_{2\times2} \end{bmatrix}$	$\begin{bmatrix} m^2 \\ \frac{m}{s}2 \end{bmatrix}$
Measurement noise, R	$\begin{bmatrix} 5 & 0 \\ 0 & 5 \end{bmatrix}$	m^2
Measurement frequency	2	s^{-1}
Simulation		
Simulation length, T	200	s
Optimization sample time	5	s

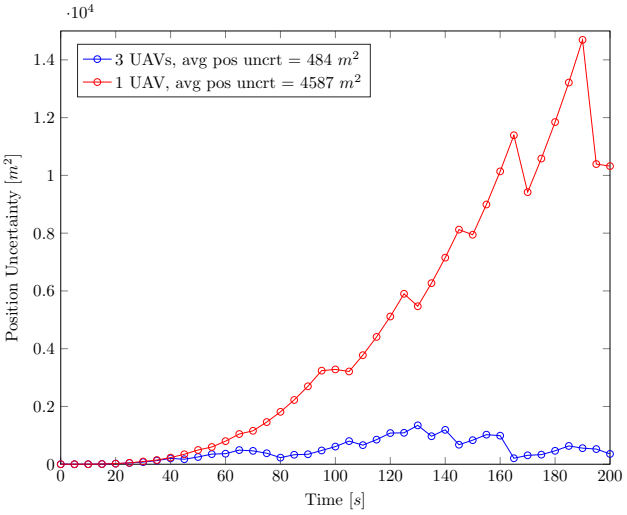

Figure 4: The average position uncertainty for the iceberg for the two simulations with 3 and 1 UAV(s).

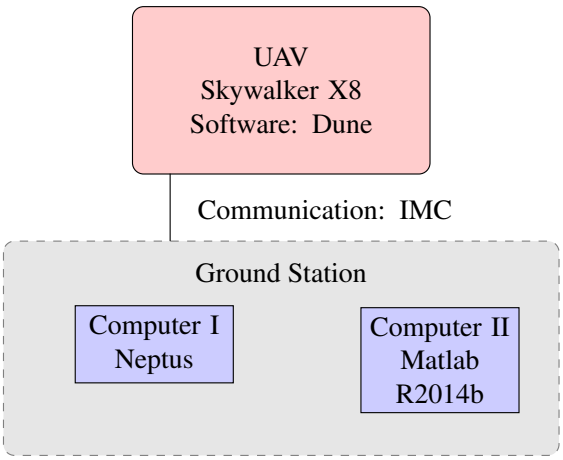

Figure 5: Practical Experiment Setup.

6 Towards practical experiments

To conduct the practical experiment in Ny-Ålesund at Svalbard there were multiple components involved. In this section, we first describe the components, including software involved in the experiments. Then, we give a description of the practical experiments conducted.

6.1 Setup

Figure 5 illustrates an overview of the components used in the experiment.

The UAV platform used for the practical experiments was an X8 from Sky Walker Technology (x8,

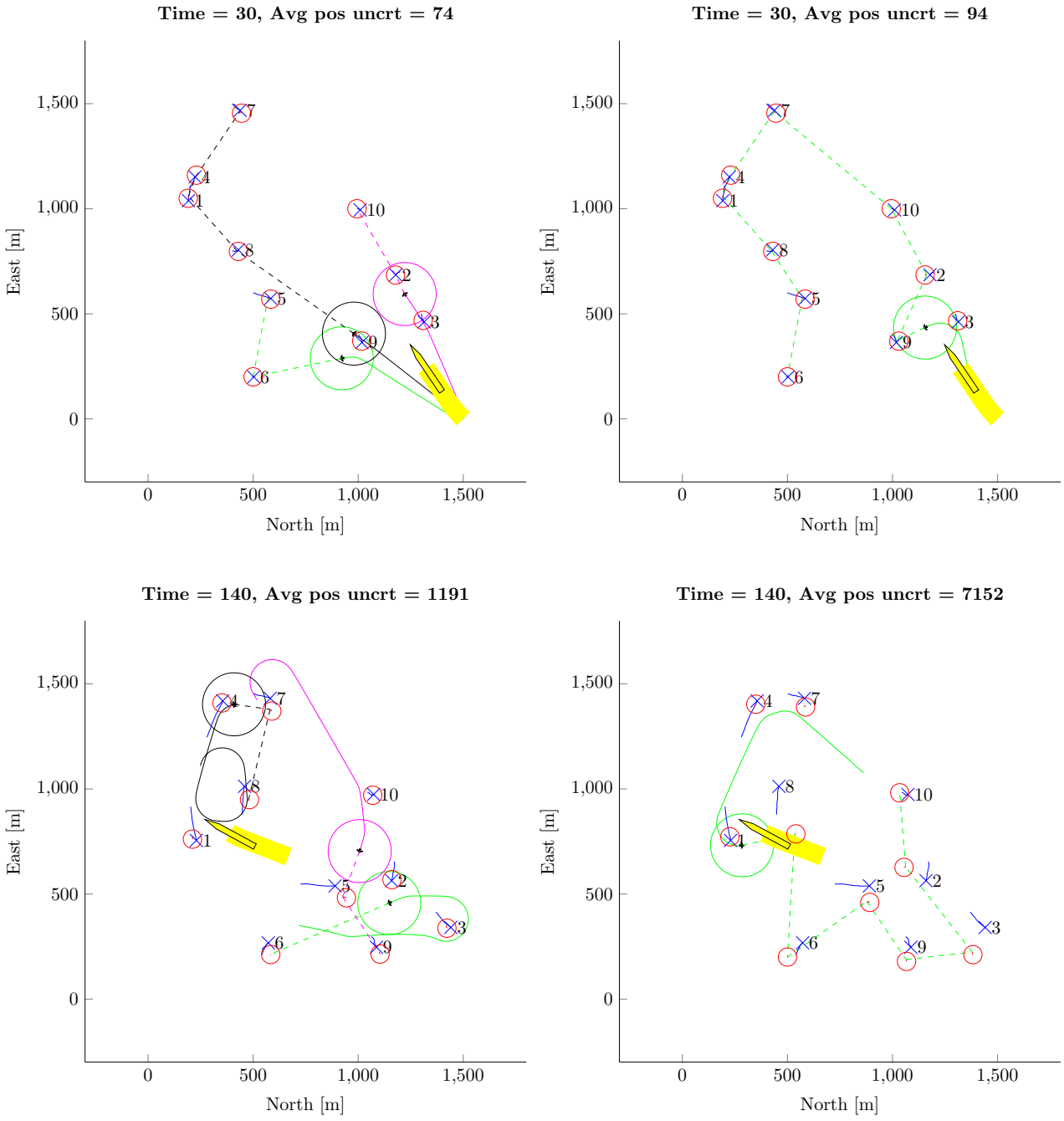

Figure 3: Simulations of a boat moving through an area with three or a single UAV(s) to track icebergs. The dotted lines illustrate the planned path for the UAVs (green, black and pink), and the solid lines show their recent movement. In addition, the FOV is marked with a solid circle around each UAV. The iceberg positions are plotted with a blue "x", and have their recent movement in a solid blue line. The observer indicates the current estimate of iceberg positions with a red circles. The boat is drawn as a yellow polygon.

Figure 6: Launch of X8 using a catapult.

2016). This is a small light-weight off-the-shelf platform with an electric motor. It can carry a light payload of 1-2 kg and is able to stay in the air for about 60 minutes. We used a catapult to launch it into the air, as illustrated in Figure 6.

The components onboard the X8 are a single board computer (odr, 2016) for processing, an autopilot (ard, 2016), a communication link (roc, 2016) and a switch which enables communication between all of the components.

The single board computer runs DUNE (Pinto et al., 2012). DUNE, short for DUNE Unified Navigational Environment, is an open-source "runtime environment for unmanned systems onboard software". This enables simple implementation of different tasks, such as reading sensor values or control of actuators, onboard the X8. In this experiment, we specifically use DUNE to receive waypoints from the ground station (i.e, the results from solving the MILP algorithm presented in Section 4), and translating and forwarding them to the onboard autopilot. The autpilot is then responsible for guiding the UAV using low level controllers to the given waypoints. Furthermore, DUNE is pulling telemetry data from the autopilot and forwarding it to the ground station, giving it the necessary information about the UAV to initialize the MILP algorithm and find the optimal visitation sequence for the simulated icebergs.

The ground station consists of two computers, each running a different software. One of the computers is running MATLAB R2014b, which is where the MILP algorithm is implemented and run. The second computer is running Neptus (Pinto et al., 2006). Neptus is an open-source Command & Control Center which can be used for a variety of tasks. Examples are world representation and modeling, mission planning, simulation, execution control and supervision, and post-mission analysis. For the work presented in this paper, Neptus is used to illustrate the location of the (simulated) icebergs, as well as the X8's position, telemetry data and current waypoint to the operator.

The communication between the UAV and the ground station is done using a protocol named Inter-Module-Communication (IMC), which is a set of predefined messages made for operations of (multiple) unmanned vehicles and real-time efficiency. The reader is referred to Martins et al. (2009) for a detailed description of the IMC protocol.

6.2 Experiment

On the way to full-scale practical experiments we did what can be considered a pilot experiment as a proof of concept. The experiment was conducted as follows:

1. Pilot launches the UAV in the air and stabilizes it at an altitude of about 120 meters.
2. The MILP-algorithm is initiated with four simulated icebergs and returns the optimal sequence for visiting them.
3. The UAV is sent a waypoint to the first iceberg from the resulting visiting sequence of the MILP-algorithm.
4. When the UAV reach the simulated iceberg it loiters around it for 30 seconds, while the ground station is running a new optimization.
5. The UAV is sent a waypoint to the first iceberg in the visiting sequence of the new optimization.
6. 100 seconds after the iceberg tracking mission is started, an additional simulated iceberg is added to the list, iceberg number 5.
7. The UAV continues to track the simulated icebergs by repeating step 3-5 for about 800 seconds of total loiter time.

The MILP-algorithm chooses a visiting sequence for the simulated icebergs based on position and an uncertainty value of each iceberg. The initial four icebergs where set with a random initial uncertainty value. After initialization, the position uncertainty increased constantly with a value of 1 for each second. In other words, $tr(p_{11})$ in equation (10) is 1. The fifth simulated iceberg was added with an uncertainty value of zero.

In Table 2 we see the coordinates of the simulated icebergs and their initial uncertainty value. Figure 7 illustrates the UAV's first visit of the five icebergs, with Figure 8 plots the position uncertainty of each simulated iceberg.

Unlike in Section 5, in the practical experiment we use an event-based approach for when to rerun the MILP algorithm. The image processing algorithm used

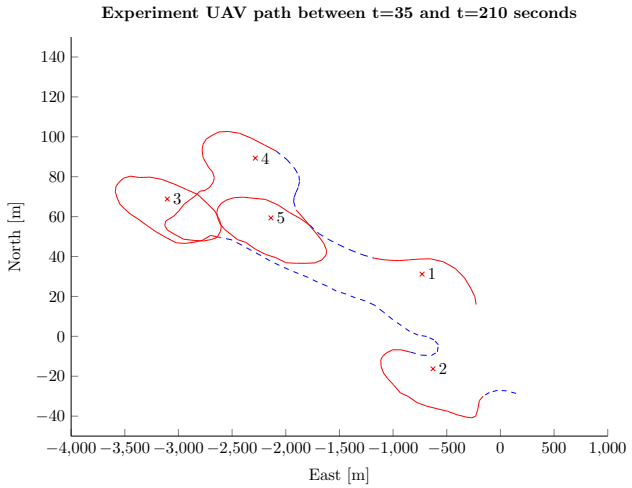

Figure 7: UAV path in practical experiments. Both the solid-red and dotted blue line illustrate the UAV path. The solid-red line indicates when the UAV is observing a point, while the dotted blue line is when the UAV is moving towards a new simulated iceberg.

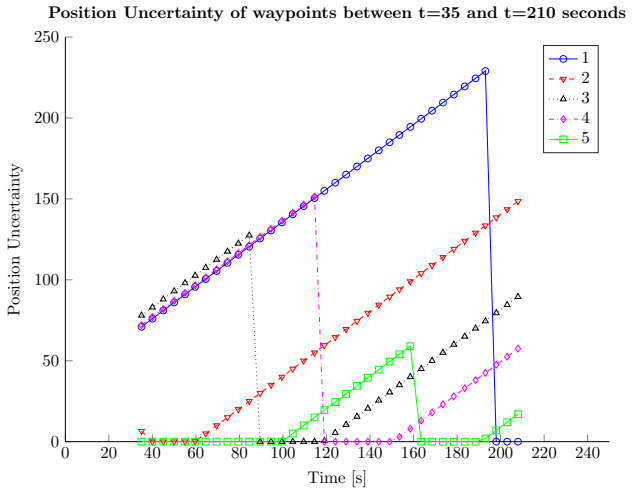

Figure 8: The position uncertainty value for each simulated iceberg during the experiment illustrated in Figure 7.

to obtain position and velocity estimate from an iceberg might require the UAV to circle around the iceberg to get more measurements than just a flyby. In our experiment, an event was the 30 seconds loitering around a simulated iceberg, which makes it natural to exploit this time to rerun the MILP-algorithm.

Table 2: Table with simulated iceberg coordinates and initial uncertainty

Nr.	Iceberg		Initial Value
1	(730	31)	1
2	(628	-16)	5
3	(3104	69)	8
4	(2284	89)	2
5	(2138	59)	1

7 Discussion

From the simulation in Section 5, a single UAV is not sufficient to track the ten icebergs. In the snapshot after 140 seconds the distance between iceberg number 8 and its estimated is larger than the FOV. In addition, we see that iceberg 2, 3, 5, 6 and 9 also are about to get outside FOV. This is not the case when in the simulation with three UAVs, which manage to keep track of the icebergs. When we look at Figure 4 the single UAV is unable to keep the position uncertainty within a limit, while three UAVs are sufficient.

The calculation of Dubins path reduces the sub-optimality originating from not taking UAV dynamics into account (cf. assumptions in Section 4.1). Especially, since we only use the first point of each sequence before we rerun the optimization. The way we reduce sub-optimality can best be explained by a simple example. Consider the case with one UAV and two icebergs, where the first iceberg is closer to the UAV in a metric distance. However, the UAV heading is turned towards the second iceberg, which makes the optimal sequence to visit the second before the first iceberg. When we use the Dubins instead of the metric distance, the algorithm manages to take the heading of the UAV into consideration. Notice that we are only able to use Dubins distance from the initial position of the UAV, since we do not know what heading the UAV will have at subsequent points.

The practical experiment serves as a proof of concept. It demonstrates how it is possible to implement the algorithm in practice. However, the practical experiment has several weaknesses. First, the simulated icebergs are too close to each other, for iceberg number 3, 4 and 5 the UAV goes directly from observing one iceberg to another. In addition, when the icebergs are so close to each other the UAV will inevitably visit each iceberg often independent of the sequence chosen by the MILP algorithm. Unfortunately, during the day of the experiment there was a lack of real icebergs. This lead to not getting image processing tested together with the MILP-algorithm.

In future experiments, a platform with greater reach

and greater durability is desired. Longer reach means that the UAV can track icebergs in a larger area, where selecting a more optimal visiting sequence for each UAV becomes essential. We believe the benefits of the proposed algorithm will be more substantial in such a case. Furthermore, in future experiments it will also be interesting to test tracking icebergs moving in open sea. The icebergs in Kongsfjorden close to Ny-Ålesund, where we did our experiments, do not move much and this makes them very easy to track.

Furthermore, for experiments in open sea the observer should be improved. It should handle false positive and false negative measurements. In addition, it should also be able to determine and reject outliers. For example, if two icebergs are close to each other at the time of observation, the observer might mistake them. This will lead to a wrong velocity estimate for both of them.

8 Conclusion

In this paper, we have studied the problem of tracking moving icebergs using a set of moving sensor platforms. To solve the problem we have used mathematical optimization, or more specifically, combinatorial optimization. To demonstrate the algorithm we have constructed a case with 10 icebergs and performed a simulation with one UAV and another with three UAVs. The simulations show that a single UAV is unable to track all ten icebergs, while 3 UAVs manage it.

We have also taken the first step towards practical implementation with a practical experiment conducted at Ny Ålesund in Svalbard during the fall of 2015. Here, we have made a successful first practical implementation of the algorithm with one UAV.

Future Work

Possibilities for future work include:

1. Integration of the of MILP-algorithm with an image processing algorithm like Leira et al. (2015).
2. Practical experiments with real icebergs and the use of image processing for iceberg detection.
3. Practical experiments with long distance UAV platform.
4. Use of multiple UAVs in practical experiments.
5. Extend observer to handle false positive and negative measurements, in addition to outliers.

Acknowledgment

This work was supported by the Research Council of Norway, Statoil, DNV GL and SINTEF through the Centers of Excellence funding scheme, grant 223254 - Centre for Autonomous Marine Operations and Systems (AMOS).

References

IBM Ilog CPLEX optimization studio. http://www.ibm.com, 2015. Accessed: 2015-09-12.

ArduPilot. http://ardupilot.com, 2016. Acc.: 2016-02-10.

Odroid U3. http://www.odroid.com, 2016. Acc.: 2016-03-02.

Sky Walker Technology (HK). http://www.skywalker-model.com, 2016. Accessed: 2016-02-10.

Ubiquiti rocket M5. https://www.ubnt.com/airmax/rocketm/, 2016. Accessed: 2016-02-10.

Albert, A. and Imsland, L. Mobile sensor path planning for iceberg monitoring using a MILP framework. In *12th Proc. Intl. Conf. Inf. Control, Automation, Robotics*, volume 1. pages 131–138, 2015.

Asnis, G. and Blackman, S. Optimal allocation of multi-platform sensor resources for multiple target tracking. In *Information Fusion, Proc. 14th Intl. Conf.* IEEE, pages 1–8, 2011.

Barton, K. and Kingston, D. Systematic surveillance for uavs: A feedforward iterative learning control approach. In *American Control Conf.* IEEE, pages 5917–5922, 2013. doi:10.1109/ACC.2013.6580766.

Bixby, R. E. Solving real-world linear programs: A decade and more of progress. *Operations research*, 2002. 50(1):3–15. doi:10.1287/opre.50.1.3.17780.

Chen, D.-S., Batson, R. G., and Dang, Y. *Applied integer programming: modeling and solution*. John Wiley & Sons, 2010. doi:10.1002/9781118166000.

Eik, K. Review of experiences within ice and iceberg management. *Journal of Navigation*, 2008. 61(4):557–572. doi:10.1017/S0373463308004839.

Enright, J., Frazzoli, E., Savla, K., and Bullo, F. On multiple uav routing with stochastic targets: Performance bounds and algorithms. In *Proc. AIAA Conf. on Guidance, Navigation, Control*. 2005. doi:10.2514/6.2005-5830.

Farmani, N., Sun, L., and Pack, D. Tracking multiple mobile targets using cooperative unmanned aerial vehicles. In *Intl. Conf. Unmanned Aircraft Systems*. IEEE, pages 395–400, 2015. doi:10.1109/ICUAS.2015.7152315.

Grundel, D. and Jeffcoat, D. Formulation and solution of the target visitation problem. *AIAA 1st Intell. Sys. Techn.*, 2004. 1:1–6. doi:10.2514/6.2004-6212.

Haugen, J. and Imsland, L. Optimization-based autonomous remote sensing of surface objects using an unmanned aerial vehicle. *2013 European Control Conference, ECC 2013*, 2013. pages 1242–1249.

Hirsch, M. J. and Schroeder, D. On the decentralized cooperative control of multiple autonomous vehicles. In *Handbook Unmanned Aerial Vehicles*, pages 1577–1600. 2015. doi:10.1007/978-90-481-9707-1_112.

Jünger, M. and et.al. *50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-art*. Springer Science & Business Media, 2009. doi:10.1007/978-3-540-68279-0.

Leira, F., Johansen, T. A., and Fossen, T. I. Automatic detection, classification and tracking of objects in the ocean surface from uavs using a thermal camera. In *IEEE Aerospace Conference*. 2015. doi:10.1109/AERO.2015.7119238.

Lesinskis, I. and Pavlovics, A. The aspects of implementation of unmanned aerial vehicles for ice situation awareness in maritime traffic. *Proc. Intl. Conf. Transport Means*, 2011. pages 65–68.

Löfberg, J. YALMIP : A toolbox for modeling and optimization in MATLAB. In *Proceedings of the CACSD Conference*. Taipei, Taiwan, 2004. doi:10.1109/CACSD.2004.1393890.

Ma, C. S. and Miller, R. H. Mixed integer linear programming trajectory generation for autonomous nap-of-the-earth flight in a threat environment. In *2005 IEEE Aerospace Conference*. IEEE, pages 1–9, 2005. doi:10.1109/AERO.2005.1559599.

Martins, R., Dias, P. S., Marques, E. R., Pinto, J., Sousa, J. B., and Pereira, F. L. IMC: A communication protocol for networked vehicles and sensors. In *Oceans 2009-Europe*. IEEE, pages 1–6, 2009. doi:10.1109/OCEANSE.2009.5278245.

Miller, C. E., Tucker, A. W., and Zemlin, R. A. Integer programming formulation of traveling salesman problems. *Journal of the ACM (JACM)*, 1960. 7(4):326–329. doi:10.1145/321043.321046.

Noon, C. E. and Bean, J. C. An efficient transformation of the generalized traveling salesman problem. *INFOR*, 1993. 31(1):39. doi:10.1080/03155986.1993.11732212.

Oberlin, P., Rathinam, S., and Darbha, S. Today's traveling salesman problem. *IEEE robotics & automation magazine*, 2010. 17(4):70–77. doi:10.1109/MRA.2010.938844.

Oh, H., Shin, H.-S., Kim, S., Tsourdos, A., and White, B. A. Cooperative mission and path planning for a team of uavs. In *Handbook of Unmanned Aerial Vehicles*, pages 1509–1545. Springer, 2015. doi:10.1007/978-90-481-9707-1_14.

Pinto, J., Calado, P., Braga, J., Dias, P., Martins, R., Marques, E., and Sousa, J. Implementation of a control architecture for networked vehicle systems. In *Proc. IFAC Navigation, Guidance, Control of Underwater Vehicles*. IFAC, 2012. doi:10.3182/20120410-3-PT-4028.00018.

Pinto, J., Dias, P. S., Gonçalves, R., Marques, E. R. B., Gonçalves, G. M., Sousa, J. B., and Pereira, F. L. Neptus: a framework to support a mission life cycle. In *Proc. IFAC Conf. Manoeuvring and Control of Marine Craft*. IFAC, 2006.

Schouwenaars, T., De Moor, B., Feron, E., and How, J. Mixed integer programming for multi-vehicle path planning. In *Control Conference (ECC), 2001 European*. IEEE, pages 2603–2608, 2001.

Shetty, V. K., Sudit, M., and Nagi, R. Priority-based assignment and routing of a fleet of unmanned combat aerial vehicles. *Computers & Operations Research*, 2008. 35(6):1813–1828. doi:10.1016/j.cor.2006.09.013.

Sinha, A., Kirubarajan, T., and Bar-Shalom, Y. Autonomous ground target tracking by multiple cooperative uavs. In *2005 IEEE Aerospace Conference*. IEEE, pages 1–9, 2005. doi:10.1109/AERO.2005.1559601.

Valavanis, K. and Vachtsevanos, G. UAV applications: Introduction. *Handbook of Unmanned Aerial Vehicles*, 2015. pages 2639–2641. doi:10.1007/978-90-481-9707-1_151.

Walton, C., Gong, Q., Kaminer, I., and Royset, J. Optimal motion planning for searching for uncertain targets. *IFAC Proc.*, 2014. 19:8977–8982. doi:10.3182/20140824-6-ZA-1003.01388.