REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	11 October 2017	Disseration

4. TITLE AND SUBTITLE

Decomposition Methods for Moving Target Search

5. FUNDING NUMBERS

6. AUTHOR(S)

Manon Raap

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

UNIBW

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES

Universität für der Bundeswehr München
Werner-Heisenberg-Weg 39
D-85577 Neubiberg Germany

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

Text in German.

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Public release. Copyrighted. (1 and 20)

12b. DISTRIBUTION CODE

ABSTRACT (Maximum 200 words)

Airborne search and rescue missions are of incredible importance to save the lives of missing persons. Such missions must be planned carefully in order to optimize the chances of survival. However, planning must be conducted within a small time frame so as not to waste time. The target of the search is often likely to move, which significantly complicates the problem. The reason for the increased complexity is that the search reward at time k does not only depend on the observation made at time k, but on all observations made up until time k. In other words, the rewards over time are inseparable and, hence, state-of-the –art shortest path planning algorithms become inapplicable.
Machine assisted translation.

14. SUBJECT TERMS

UNIBW, German, Moving Target

15. NUMBER OF PAGES

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Decomposition Methods for Moving Target Search

Manon Raap

Vollständiger Abdruck der von der Fakultät für Informatik der

Universität der Bundeswehr München

zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Promotionsausschuss:
Prof. Dr. Stefan W. Pickl
Prof. Dr.-Ing. Peter Stütz
Prof. Dr.-Ing. Markus Siegle
Prof. Dr.-Ing. Reiner K. Huber
Prof. Dr. Wolfgang Hommel

Die Dissertation wurde am 30.06.2017 bei der Universität der Bundeswehr München eingereicht und durch die Fakultät für Informatik am 10.11.2017 angenommen. Die mündliche Prüfung fand am 14.12.2017 statt.

Decomposition Methods for Moving Target Search

by

Manon Raap

Submitted to the Fakultät für Informatik
on 30-06-2017, in partial fulfillment of the
requirements for the degree of
Doktor der Naturwissenschaften (Dr. rer. nat.)

Abstract

Airborne search and rescue missions are of incredible importance to save the lives of missing persons. Such missions must be planned carefully in order to optimize the chances of survival. However, planning must be conducted within a small time frame so as not to waste time. The target of the search is often likely to move, which significantly complicates the problem. The reason for the increased complexity is that the search reward at time k does not only depend on the observation made at time k, but on all observations made up until time k. In other words, the rewards over time are inseparable and, hence, state-of-the-art shortest path planning algorithms become inapplicable. A pilot who is faced with such a complex task in a stressful situation is prone to planning a suboptimal search trajectory. Coordinating a team of cooperating aerial platforms is especially difficult. Automation of search trajectory optimization is therefore the aim of this dissertation. Three novel problems are considered in this thesis: single platform search under kinematical constraints, single platform search under kinematical and resource constraints and strategy optimization for a team of heterogeneous cooperating platforms with shared resources. A mixed integer linear problem (MILP) formulation is proposed as well as a decomposition method for solving each problem. Computational experiments and simulations show that each proposed model and algorithm is applicable and efficient for solving its considered problem. The first problem variation is solved much faster using the proposed generalization of a branch & bound algorithm compared to solving the MILP formulation using a commercial solver. To solve the second problem variation, a Benders' decomposition algorithm is developed for more efficient optimization of the proposed MILP formulation. This algorithm significantly reduces the computation times for solving the problem with scarce resources. The proposed linear upper bound for solving the third problem variation improves a previous linear upper bound by a very large margin. Furthermore, the proposed branch & price algorithm reduces computation times significantly. Finally, a first step towards a decision support tool is made by applying the proposed methods to a terrorist threat scenario of a search for a radiological dispersion device. The input parameters for search effectiveness here are obtained in field experiments.

Thesis Supervisor: Stefan W. Pickl
Title: Prof. Dr.

Thesis Supervisor: Peter Stütz
Title: Prof. Dr.-Ing.

Acknowlegdements

First and most importantly, I want to describe my gratitude towards my supervisor Prof. Dr. Stefan Pickl for including me in his research group COMTESSA, the Core Competence Center for Operations Research Tenacity - Management - Excellence Safety & Security Alliance. Besides many opportunities for personal growth, he also gave great guidance for this dissertation by giving insightful feedback after internal presentations and by advising on the global content during meetings on my progress. I furthermore thank my second supervisor Prof.-Ing. Dr. Peter Stütz for his advice on the practical side of my research.

I am very grateful to my advisor and friend Dr. Martin Zsifkovits. Martin helped me better understand the world of academics and my place in it. He is the perfect advisor that everyone should wish for. He was of great support both practically and mentally.

This dissertation would not have been established without the 3-year scholarship provided by Airbus Defence & Space in Manching, Germany. I am very grateful to Thomas Pic, head of the department for unmanned aerial systems, for assigning this scholarship to me. I am also very grateful for his understanding of my decision to proceed internally at the Universität der Bundeswehr after the first year and for the remained valuable contact.

I also thank Prof. Dr. Alexander Bordetsky from the U.S. Naval Postgraduate School Center for Network Innovation and Experimentation (CENETIX) for the opportunity to conduct the field experiments for this dissertation. Furthermore, I am grateful to Prof. Dr. Daniel Nussbaum for the invitation to present my work at the Operations Research Department at the U.S. Naval Postgraduate School for, among others, Prof. Dr. Alan R. Washburn, Prof. Dr. Robert F. Dell, and Prof. Dr. Moshe Kress. I am very grateful for the validation of my work and provided feedback by these founders of search path optimization.

Furthermore, I would like to thank my former colleague Dr. Silja Meyer-Nieberg for everything she learned me during my first year about scientific writing. My colleagues and fellow PhD students in alphabetical order Gonzalo Barbeitos, Elisa Canzani, Maximilian Moll, Marian Sorin Nistor, Truong Son Pham, Michael Preuß and Zhonglin Wang, who all became friends throughout the past years, have been a large part of my live in the past years. The mutual support within our group as well as the open atmosphere is unique in

my opinion. Not a single day has passed without a good laugh. I would also like to thank the Universität der Bundeswehr for all its excellent sport facilities. I gratefully used the swimming pool, the climbing hall, and the gym during lunch breaks, after work hours, or weekends to stay fit for 'mens sana in corpore sano' - a healthy mind in a healthy body.

Of course I am also very grateful to my parents Sybolt-Jan Raap and Anita Steggerda for everything. Last but not least, I thank my love Sander Kan for his patience and for all those skiing- and mountain biking trips that make me feel alive and happy.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The research presented in this thesis was initiated by Airbus Defense & Space in Manching, Germany. To be more precise, by its department for unmanned aerial vehicles (UAVs) and systems (UASs). There exists a demand for solutions such as the ones developed in this research, as many customers that purchase unmanned aerial vehicles and systems for search missions are interested in supporting software for search strategy optimization as well. The necessary requirements for the optimization methods have been determined in cooperation with in-house experienced pilots and UAV pilots.

The remainder of this introduction begins with the motivation for the topic in section 1.1, followed by the scope in section 1.2. The main contributions of this thesis are summarized in section 1.3, and finally, section 1.4 provides an outline of this thesis.

1.1 Motivation

Recent events have shown the enormous importance of search-and-rescue missions. The number of lives that were lost at sea after refugee ships sunk in the Mediterranean and Aegean seas exceeded 2500 in the first months of 2015 [1]. Operational decisions for a search mission using a fleet of aircraft are made by an assigned coordinator of a Maritime Rescue Coordination Center (MRCC). This coordinator allocates the search effort by assigning surveying assets to distinct subareas. This task is already supported by systems based on algorithmic search approaches, e.g., the search and rescue optimal planning system [2] is

currently used by the United States Coast Guard. Nevertheless, individual pilots are expected to plan their optimal trajectory by hand, which is tremendously complex; it is proven to be an $\mathcal{NP}$-complete optimization problem by [3] for a single platform searching for a single *stationary* target. Planning for *moving* target search is considerably more complex in general, and moreover, the kinematical constraints of the aircraft must also be taken into account. Pilots must be ready for take off within the prescribed time to preparedness, which is a maximum of 30 minutes by international agreement. Planning a search trajectory is even more complex when multiple vehicles are assigned for a cooperative search. Not to mention the additional complexity when factors such as constraints on resources (e.g. time exposed to risk) must be taken into account. Executing such a complex task in a stressful situation is susceptible to resulting in a sub-optimal search trajectory and rescue may come too late. The aim of this thesis is therefore to automate this task with an outlook towards autonomous search missions by unmanned aerial vehicles (platforms). A major benefit of exploiting unmanned vehicles is that the current 30-minute time to preparedness for the crew can be reduced, compared to the deployment of manned vehicles. The recent rise in technological development of unmanned aerial systems makes it very probable that these will be exploited in the future. The methods presented in this thesis are applicable for aerial sensor platforms in general, i.e. platforms that are either fixed-winged or rotary-winged, either manned or unmanned and either autonomous or non-autonomous. An aerial sensor platform is referred to by *platform* for short in the remainder of this thesis.

1.2 Scope

This thesis focuses on the problem of *search* for *moving* targets by fixed-wing and rotary-wing platforms. Methods for solving three variants of this problem are proposed:

- Search by a single platform under kinematical constraints.
- Search by a single platform under kinematical and resource constraints.
- Cooperative search by a team of heterogeneous platforms with shared resources.

Here, resource constraints is an abstract notion which can be specified to obstacle-, threat-, or collision avoidance and communication-, time-, or fuel constraints. These problems are considered in *discrete time* for computational purposes, whereas moving target search in continuous time has been addressed in literature [4–7] as well. Planning a search strategy can be conducted either implicitly or explicitly [8]. In implicit planning methods, the plan specifies how the platform interacts with the environment and how it responds to sensor observations. A plan in this context is a prescription of how to react. In explicit planning methods, a search-path is computed based on predictions of the environmental state and predictions of the sensor observations. This thesis focuses on *explicit* planning, because it allows for anticipation of estimated target movements. Anticipation leads to a higher probability of detection compared to merely reacting to an observation. The objective is to maximize the probability of detecting the target. Other types of objective functions can be considered, for example minimal expected time to detection or minimizing the latest time to detection. However, maximizing the probability of detection corresponds to the actual real-life objective in most search missions and is by far the most used objective in literature as well.

Many related problems have been studied in literature. In order to make a clear distinction, related important problems which are not included in the scope of the research in this thesis are the following. Problems concerning *terrain covering* or *static target search* are not considered. Neither are targets considered that may react in any way to the searcher, such as evasion or cooperation. Such problems are known as *two-sided* search problems. The focus in this thesis is on *one-sided* search problems. Other related but excluded problems concern imaging, recognition, classification, tracking, surveillance, target state prediction, target motion modeling, sensor effectiveness, information merging, sensor fusion and shortest path to moving target problems. Nevertheless, solutions to these problems are critical to ensure the effectiveness of a search strategy in real-life search missions.

The proposed methods are applicable to real-life search missions by a single platform or by a team of platforms. Therefore, the constraints of the individual platform given through flight physics need to be taken into account for each optimized trajectory. Physically feasible trajectories can be optimized for any type of aerial sensor platform. This is a unique aspect

of this work, as most approaches in literature consider a higher level of planning in which the platforms are assigned to search a subarea during a period of time. With such high level search plans, the actual flight trajectory within the subarea must still be decided on by the pilot. The proposed methods in this thesis are therefore much better suited for a more automated conduction of search missions. Finally, even though the search for a single target is considered throughout this thesis, the proposed methods are applicable to multiple targets by means of a simple and straightforward extension.

1.3 Contributions

The main contributions of this work are listed and elaborated upon in the following. Each of the contributions successfully addresses one of the research gaps as identified in section 3.2.

- **Discrete uniform model for platform kinematics with timeline synchronization.**

The first main contribution consists of a unified model for fixed-wing and rotary-wing platforms, taking kinematical constraints into account. A complicating aspect in moving target search is that the timelines of the target and platforms must be synchronized. When the distances between waypoints are varying while the time steps remain constant, the platform would have to constantly accelerate and decelerate to stay in sync. The constant arc lengths as proposed are therefore much better suited for time dependent aerial vehicle trajectory planning. Additional details are incorporated in the proposed model, e.g. relocation arcs on which search effort is traded off for quick relocation and feasible transitions between trajectories planned in consecutive planning stages. Furthermore, a set of linear constraints is proposed for physically feasible waypoint selection, resulting in a physically feasible search trajectory for the platform. This contribution has been published in the Journal of Optimization Theory and Applications [9] and is a continuation of ideas from the author's master thesis [10].

- **A mixed integer linear programming formulation for Markovian target search trajectory optimization on heterogeneous grids with resource constraints.**

The second main contribution consists of a novel model formulation for Markovian target search trajectory optimization, which generalizes and improves the author's previous work in [9, 10] and has been published in the journal Computers and Operations Research [11]. This is an initial approach that takes both kinematical and resource constraints into account, and furthermore, allows for modeling of the target and platform on heterogeneous grids. The use of heterogeneous grids is important when aiming for a constant speed and for a detailed map of the estimation of the position of the target. Finally, this is a first formulation of the considered problem that allows for being decomposed using decomposition techniques. Decomposition techniques are useful to solve complex problems by decomposing the problem into a set of smaller problems which are easier to solve.

- **Generalization of a state-of-the-art branch & bound algorithm for Markovian target search trajectory optimization incorporating kinematical constraints and heterogeneous grids.**

The third main contribution consists of a generalization of a state-of-the-art branch & bound algorithm for moving target search trajectory optimization. The proposed alternation results in an algorithm that yields an optimal search trajectory which is physically feasible for the considered platform. Furthermore, the generalization allows for modeling of the target and platform movement over heterogeneous grids. To the best of the author's knowledge, all developed K-step-lookahead methods in literature model both to move over one shared grid. The concept of heterogeneous grids has two advantages over one shared homogeneous grid; a target specific grid can be constructed as fine as necessary to keep a detailed probability map of the target position, whereas a hexagonal grid for the platform allows for modeling a more natural flight trajectory. All the proposed methods in the following contributions account for kinematical constraints.

- **A Benders' decomposition algorithm for Markovian target search trajectory optimization with resource constraints.**

The fourth main contribution consists of a first Benders' decomposition algorithm for search trajectory optimization, allowing to disconnect the target and platform networks and iteratively solve two much smaller problems until the desired optimality tolerance has been reached. First, the proposed mixed integer linear program is extended to account for resource constraints, which is then decomposed using a Benders' decomposition approach. This results in the first method for moving target search that takes both kinematical and resource constraints into account. In general, this is the first method utilizing a Benders' decomposition approach for solving a static or moving target search problem. An intermediate version of this method is published in the proceedings of the biannual Student Conference on Operations Research [12]. The proposed Benders' decomposition algorithm significantly reduces the computation times for solving the problem with scarce resources.

- **A branch & price algorithm for cooperative search by a team of heterogeneous platforms with shared resources.**

The fifth main contribution consists of an initial method for solving a cooperative search trajectory problem for a team of heterogeneous platforms with shared resources. In addition, this is the first branch & price algorithm for solving a moving target search optimization in general. In literature, branch & price algorithms have shown to be especially powerful when the underlying sub-systems have a special combinatorial structure, such as a trajectory. The method consists of a novel non-convex mixed integer programming formulation which is relaxed linearly. This proposed relaxation yields an upper bound to the problem that is significantly tighter compared to the previous tightest linear upper bound in literature [13]. Due to the increased complexity to the problem when considering multiple platforms the non-convex formulation is computationally intractable. Therefore, the proposed branch & price algorithm solves the relaxed problem to optimality yielding a tight upper bound to the original problem. In addition, a novel branching mechanism is proposed that yields better results compared to the usual branching mechanism used in literature in branch & bound

approaches for moving target search.

- **Field experiments for detection of a radiological dispersion device**

The final main contribution consists of a description of how the input parameters such as the search effectiveness can be determined, such that the proposed branch & price method can be used to provide decision support in real-life search missions. Field experiments are conducted in order to determine the maximum ranges of radiological sensors for the detection of hazardous material that can be used in a terrorist attack. The results of computational experiments in a realistic scenario show the applicability of a decision support tool with the proposed method. In addition to the theoretical search instances solved in the previous chapter, the results of these more realistic instances of search for a suspect carrying a radiological dispersion device show that the proposed method for cooperative search trajectory optimization for a team of heterogeneous platforms yields a significantly higher probability of detection when compared to existing methods. Furthermore, the computation times of the proposed method are shorter in almost all cases, such that the team of platforms can start the search sooner, which in turn leads to earlier detection and, hence, earlier threat relief.

1.4 Outline

The remainder of this thesis is structured as follows. First, in chapter 2, a formal description of the three considered problem variations is given. The overview of the related literature and derived research gaps are described in chapter 3. Then, the method for solving the first problem variation of search by a single platform under kinematical constraints is described in chapter 4, followed by the method for solving the second problem variation of search by a single platform under kinematical and resource constraints, described in chapter 5. The method for solving the third problem variation of cooperative search by a team of heterogeneous platforms with shared resources is described in chapter 6. Field experiments towards decision support in search for radiological material and the corresponding computational results are described in chapter 7. Finally, the conclusions are listed in chapter 8.

Chapter 2

The problem of moving target search

The three considered variations of the problem of moving target search are formally formulated in this chapter, starting with the first problem of search by a single platform under kinematical constraints in section 2.1. The next problem concerns search by a single platform under kinematical and resource constraints and is formulated in section 2.2. Finally, the problem concerning cooperative search by a team of heterogeneous platforms with shared resources is described in section 2.3.

2.1 Search by a single platform under kinematical constraints

This section describes the search trajectory problem for a single platform under kinematical constraints as introduced in [14]. The problem formulation consists of the following aspects: the probability map for the target position, the target model, the sensor model, the platform model and, finally, the search objective. These aspects are further described in the following, similarly to the author's previous works in [9, 11].

The search area is modeled by discretization into a finite set of cells $\mathcal{C}$ and the duration of the search mission is discretized as well into a sequence $\mathcal{K}$. The time allocated to a planning stage is defined by a sequence $\mathcal{K} = (1, \ldots, K)$ of K time steps. The target to search for is assumed to occupy one unknown cell $C_k \in \mathcal{C}$ at time step $k \in \mathcal{K}$. For the duration of the

search mission, a probability map pc_k is maintained, where the probability of containment $pc_{k,c}$ represents the probability of the target occupying cell c at time k. Although the exact initial position of the target is unknown, it is characterized by a known initial probability distribution pc_1. The target track is modeled by a stochastic process $(C_1, ..., C_K)$, which is assumed to be Markovian [15]. The probability map evolves due to the target motion according to

$$pc_{k+1,c} = \sum_{c' \in \mathcal{C}} d(c', c)\, pc_{k,c'}, \tag{2.1}$$

where the transition function $d(c', c) \in [0, 1]$ represents the probability that the target moves from cell c' to cell c and must be estimated for each cell $c' \in \mathcal{C}$. An example of an evolution of the probability map when no observations are made is shown in Figure 2-1.

A plane above the search area, at the height that is considered optimal for search by the considered platform, is also discretized into a finite set of cells. Each of the cell centers is represented by a node. The resulting set of nodes is denoted by $\mathcal{V}$. A search trajectory is a sequence of waypoints, in which a waypoint is defined by a time-node combination (k, v) with $k \in \mathcal{K}$ and $v \in \mathcal{V}$. The object to control in this problem is an aerial platform. Its motion model, adopted from [16], is given by

$$\begin{aligned} \theta_{k+1} &= \theta_k + dt \cdot \dot{\psi}_k \\ x_{k+1} &= x_k + dt \cdot s_k \cos(\dot{\psi}_k) \\ y_{k+1} &= y_k + dt \cdot s_k \sin(\dot{\psi}_k), \end{aligned} \tag{2.2}$$

where dt is a time increment, parameter s_k is the speed of the platform, parameter θ_k is its heading angle and $\dot{\psi}_k$ is its turn rate at time k, which are obviously restricted by the laws of physics. Variables x_k and y_k represent the coordinates of the platform on the plane above the search area at time k, i.e., waypoint $(v, k) = [x_k, y_k]$. Throughout this thesis, a trajectory is represented by the binary vector $\boldsymbol{z} = (z_{k,v})_{k \in \mathcal{K}, v \in \mathcal{V}}$, such that

$$z_{k,v} = \begin{cases} 1 & \text{if } (k, v) \text{ is a waypoint on the trajectory,} \\ 0 & \text{otherwise.} \end{cases} \tag{2.3}$$

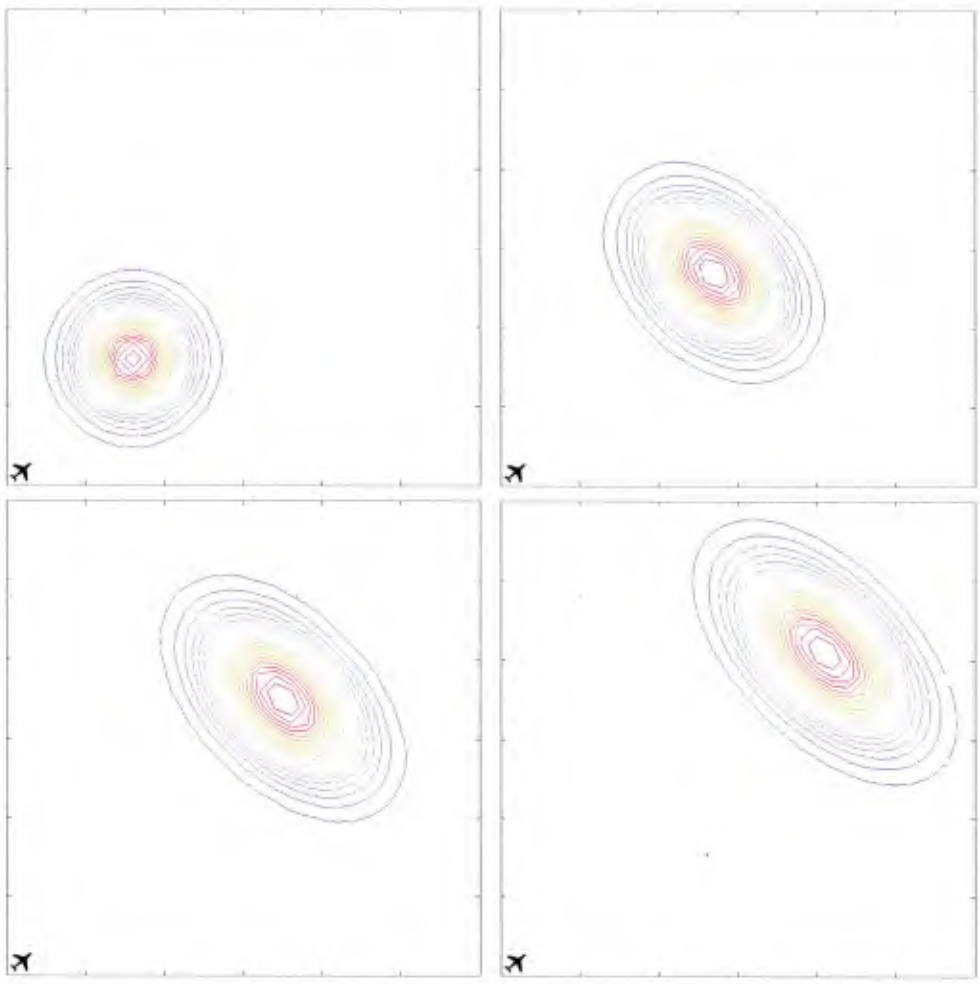

Figure 2-1: Example of an evolution of the probability map. The platform remains stationary in the bottom left corner of the search area causing no observations to be made. The probability map evolves according to formula (2.1) due to the probabilistic nature of the motion model of the target. It moves north and east with equal probability.

A trajectory is constrained by the motion model in (2.2). The set of trajectories on $\mathcal{V}$ that do not violate the motion model in (2.2) is denoted by $\boldsymbol{Z}$.

It is furthermore assumed that the considered aerial sensor platform has a stabilized sensor equipped to make observations. However, it is possible to overlook the target. The probability of not overlooking the target is referred to by *glimpse probability* [3]. The glimpse probability can be positive and variant for multiple cells at once, as visualized in Figure 2-2.

When observations are made, the probability map evolves according to the motion model

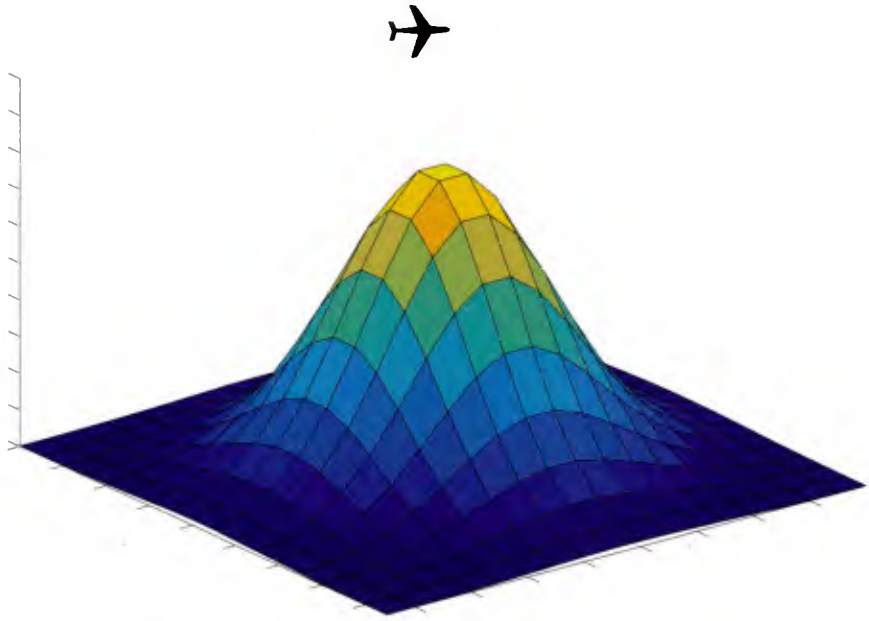

Figure 2-2: Variant glimpse probability over the target grid from one point of observation.

as in Equation (2.1) and, in addition, evolves according to the obtained glimpse probabilities for each cell as in Equation (5.29). Therefore, Equation (2.1) is extended to account for observation results as follows:

$$pc_{k+1,c} = B \sum_{c' \in \mathcal{C}} d(c', c)\, pc_{k,c'} \left(1 - pg_{k,v,c'}\right), \tag{2.4}$$

where the normalization coefficient B is given by

$$B = \left(\sum_{c \in \mathcal{C}} pc_{k,c} \left(1 - pg_{k,v,c}\right) \right)^{-1}. \tag{2.5}$$

After the first observation, the sum of the probabilities of containment over the cells do not sum up to unity. Therefore, the probability on containment pc_k is not a real probability for $k > 1$. It is merely an indication of the probability of containment compared to the other cells. Nevertheless, pc_k is not normalized because the difference in the expected probability of containment at time 1 and at time K yields the useful expected probability of detection. An example of an evolution of the probability map when observations are made is shown in

Figure 2-3.

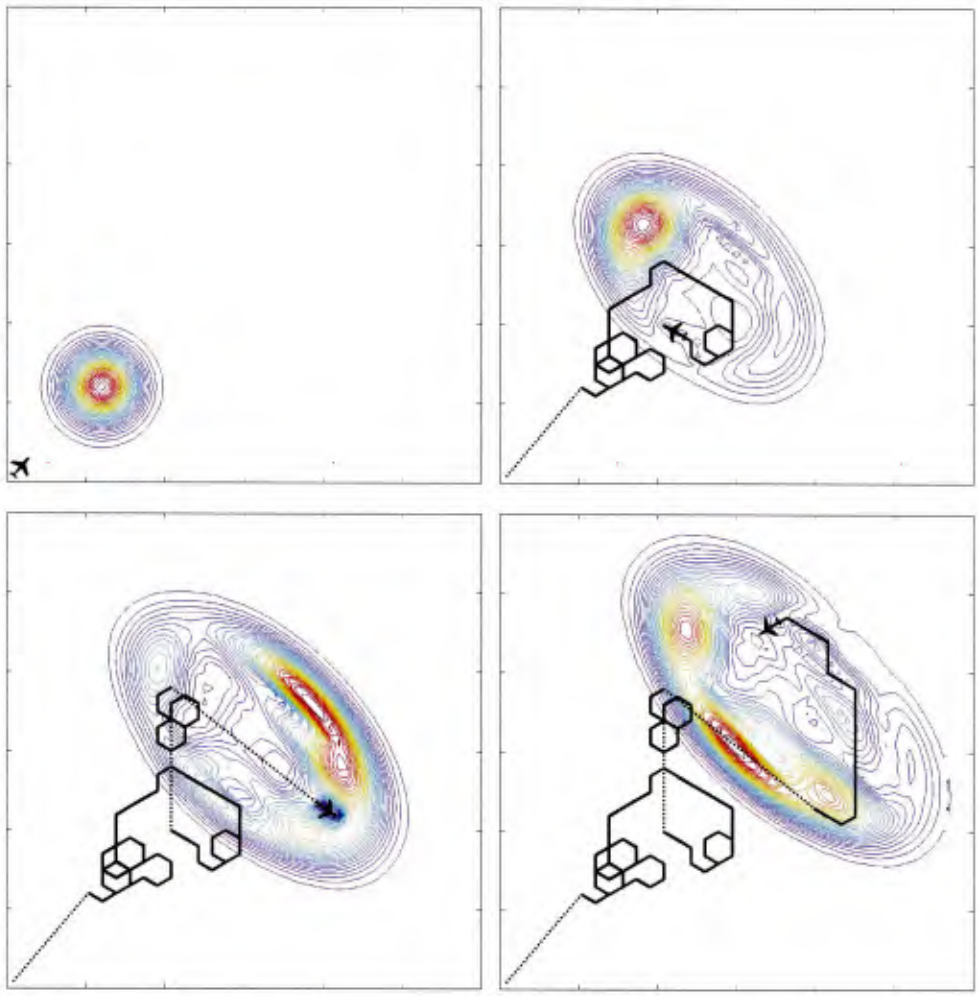

Figure 2-3: Example of an evolution of the probability map, when observations are made. The platform follows a flight trajectory and makes an observation in each time period. The probability map evolves according to formula (2.4) due to the probabilistic nature of the motion model of the target and the observations made by the platform. The target moves north and east with equal probability.

The objective is to determine a search trajectory $\boldsymbol{z} \in \boldsymbol{Z}$ maximizing the expected probability of detection over time period $\mathcal{K}$, i.e.,

$$\max_{\boldsymbol{z}\in\boldsymbol{Z}} \sum_{k=1}^{K} \sum_{c\in\mathcal{C}} pd_{k,c}, \tag{2.6}$$

where $pd_{k,c}$ is the probability of detecting the target at time k in cell c and is calculated by

$$pd_{k,c} = pc_{k,c} pg_{k,v,c} z_{k,v}. \tag{2.7}$$

The probability of containment $pc_{k,c}$ is calculated through Equation (2.4). Furthermore, the probability of containment $pc_{k,c}$ is calculated by Equation (2.4), but with $B = 1$. The normalization from Equation (2.5) is omitted, so that the probability map is not normalized. Consequently, the probability of containment $pc_{k,\cdot}$ does not represent an actual probability anymore, since it does not sum to unity over the grid cells. It does, however, sum to the probability that the target has not been found up until time k, despite the search effort. Therefore, the objective function in Equation (2.6) yields the expected probability of detection over $\mathcal{K}$.

2.2 Search by a single platform under kinematical and resource constraints

This section describes the search trajectory problem for a single platform under kinematical and resource constraints. It is an extension of the first problem that is described in the previous section 2.1. In addition to the kinematical constraints, this problem takes resource constraints into consideration as well. An abstract notion of resources is used in this thesis. In practice, resource constraints can be interpreted as e.g. communication, fuel, and linear risk constraints such as the total time exposed to risk. Furthermore, the same type of formulation can be applied to collision or obstacle avoidance.

Formally, the required resource at node v at time k is denoted by $pr_{k,v} \in \mathbb{R}^+$ and the limit on resource consumption by $T \in \mathbb{R}^+$. The total resource consumption on the trajectory is not allowed to exceed the limit T. Apart from these additional resource constraints, this problem is equivalent to the former.

2.3 Cooperative search by a team of heterogeneous platforms with shared resources

A formal formulation of the problem of cooperative search by a team of heterogeneous platforms with shared resources is given in this, final section of the chapter. It differs from the previous problem in two ways. The first being the cooperative search by a team of heterogeneous platforms, and the second being the general target motion as the Markov assumption is dropped.

The search area is still modeled by discretization into a finite set of cells $\mathcal{C}$ and the duration of the search mission is discretized as well into a sequence $\mathcal{K}$. The target to search for is assumed to travel over an unknown track $\psi = (C_1, C_2, ..., C_K) \in \Psi$ with probability pt_ψ, in which Ψ is the set of possible tracks for the target to take. Search for the target is conducted by a set $\mathcal{U}$ of heterogeneous platforms. For each platform $u \in \mathcal{U}$ an effectiveness of search is given by parameter $W_{u,k,v,c}$. Furthermore, let $pg_\psi \in [0, 1]$ be the conditional probability that the target is detected by at least one of the platforms, given that the target takes track ψ. The glimpse probability is given by the typical [13, 17–19] formula

$$pg_\psi = 1 - e^{-\sum_{u\in\mathcal{U}}\sum_{k\in\mathcal{K}}\sum_{v\in\mathcal{V}}\sum_{c\in\mathcal{C}} W_{u,k,v,c}x_{\psi,k,c}z_{u,k,v}}, \tag{2.8}$$

where indicator $z_{u,k,v} \in \{0, 1\}$ takes the value 1 when platform u visits node v at time k and indicator $x_{\psi,k,c} \in \{0, 1\}$ takes the value 1 when the target occupies cell c at time k given its track being ψ.

The aim is to optimize a search strategy that consists of one search trajectory for each platform $u \in \mathcal{U}$. A search trajectory is represented by the binary vector $\boldsymbol{z_u} = (z_{u,k,v})_{k\in\mathcal{K},v\in\mathcal{V}}$ and must be in the set of physically feasible search trajectories $\boldsymbol{Z}_u$ for platform u. The objective is to find a search strategy that maximizes the cumulative glimpse probabilities over the sum-product of all possible target tracks and the probability that track is taken, i.e.

$$\text{Maximize} \sum_{\psi\in\Psi} pt_\psi pg_\psi. \tag{2.9}$$

This objective maximizes the joint probability of detection. Finally, the set of search trajectories of the platforms is referred to by *search strategy*. Analogous to the previous problem the search strategy can be constrained by a shared resource limit. Resource consumption for one platform u at time k at node v is represented by parameter $pr_{u,k,v}$. The cumulative resource consumption may not exceed a given limit T.

Chapter 3

Literature research

An extensive research of the related literature is described in this chapter. A number of surveys precede this overview. A survey of the search theory literature up until 1991 is presented in [20]. A related survey in [21] offers a review of literature on search-path planning and sensor scheduling. Its main focus is on dynamic programming methods for stochastic optimal control. Besides planning for search, tracking of multiple ground targets and exploration is taken into consideration as well. The survey in [22] focuses on search-path planning methods for rotary-wing platforms. Another related survey, with a focus on mobile robotics, can be found in [23]. This survey contains all methods within the scope from section 1.2 up until April 2017.

The overview of related literature is presented in section 3.1, followed by the identification and description of the research gaps in section 3.2.

3.1 Literature overview

Related literature comes from two different directions of research: operations research and applied fields such as aviation technology and robotics. The first direction focuses mainly on a high level of search strategy optimization, in which the strategy consists of an allocation of search effort. It aims to manage one or a team of searchers in a top-down fashion. The search area is typically divided into subareas, in which search effort is allocated over time. A subarea is to be searched during a given number of time periods. The search effort is

typically expressed in number of searchers of a certain type. The sequence of subareas allocated to a platform is path-constrained, i.e. consequent subareas must be adjacent. These subareas are often larger than the field of view of a sensor platform, so the actual search trajectory within the subarea is not yet planned. Research from applied fields such as aviation technology and robotics focus mainly on optimizing search trajectory. Here, the resulting search trajectory must be physically feasible for the platform. The parameters are speed, turning radius, climb/dive angle, and acceleration, which are obviously restricted by the laws of physics.

In this overview, the methods from literature for search effort allocation are first described in subsection 3.1.1. The methods from literature for search trajectory optimization are described in subsection 3.1.2. Finally, all discussed methods are tabularly summarized in subsection 3.1.3.

3.1.1 Search effort allocation

The first approaches under review are those optimizing search effort allocation. Within this category, the methods are divided into exact methods for a single searcher, heuristics for a single searcher, and methods (both exact and approximate) for multiple searchers.

Exact methods for a single searcher

Eagle [24] was the first to consider the problem as a partially observable Markov decision process, assuming Markovian target motion. As was standard for problems exploiting a Markov assumption, the solution technique here is dynamic programming. Possible search paths are defined by a finite set of vectors. Eagle noted that some vectors are dominated and can be removed from consideration. He proposed a potentially large linear program to check for dominance, supplemented by two alternative heuristic reduction schemes. Stewart [25] introduced the path-constrained search effort allocation problem in 1979. He chose a depth-first branch & bound approach, because most approaches to linear binary problems were of this type. In this algorithm, lower bounds on the probability of non-detection are obtained by relaxing the searcher's path constraints. The resulting substitute problems

are being solved using a modification of Brown's algorithm [26]. However, Brown's algorithm does not guarantee optimality for discrete search effort, so obtained bounds are only approximate. This may result in mistakenly pruning an optimal branch. Experimental analysis with this method is presented by Stewart [27]. Eagle and Yee [17] continued the work by replacing the approximate bound with a true lower bound, which is obtained by relaxing the integrality constraints of the searcher's position. The relaxed problem is then solved using the Frank-Wolfe method [28].

Martins [18] reconsidered the Eagle-Yee procedure and proposed a new bound. Contrary to Stewart's and Eagle-Yee's approaches, no constraints are relaxed. Instead, the maximum expected probability of detection provides an upper bound for the maximum probability of detection. This objective function is linear in the decision variables, so maximizing the expected probability of detection is equivalent to finding the longest path through a directed acyclic network. The longest path algorithm is adapted from Cormen *et al.* [29], which has linear complexity. This bound is more easily evaluated than the Eagle-Yee bound, but is less tight. Despite more branching, run times are still roughly a factor of four smaller on all test problems.

In 1995, Washburn summarized the computational experiences of the branch & bound procedures mentioned above in [30]. He proposed and evaluated a type of hybrid approach where a backup bound is calculated if and only if the primary bound comes within δ of fathoming the segment, where $\delta \geq 0$. This arises from the idea that putting in some more effort of computing a tighter bound may prevent branching. In all examined test cases, a pure procedure is at least as good as any hybrid. A hybrid method is generally slightly slower, but Washburn suggests that the gain in robustness might be worth the sacrifice. Washburn's preferred hybrid procedure is Martins' bound backed up by the Eagle-Yee bound.

Washburn [19] published another paper on computational results on several bounds in the branch & bound procedure, a few years later in 1998. He proposed a new bound, which combines the relaxation of the integrality constraint on the searcher's position and a linear overestimation of the glimpse probability function. The need for solving a longest path problem is removed, but the bound is less tight on account of the additional relaxation. His iterative forward-and-backward algorithm [31] yields optimal results for this relaxed

problem.

Lau *et al.* [32] introduced an improvement of Martins' bound, called the discounted mean (DMEAN) bound. In the calculation of Martins' bound, the reward at time k is calculated while ignoring all observations made at time steps other than k. It is known that the true probability of detection at time step k depends on the cumulative probability of non-detection at all previous time steps. By additionally accounting for the probability of detection at time step $k-1$, the bound is tightened. They showed with the help of computational analyses that this bound leads to faster solution times than any of the existing bounds, because it is significantly tighter with almost no additional computation costs. The approach was furthermore extended to find solutions for the search effort allocation problem with non-uniform travel times.

All bounds mentioned so far are *dynamic* in the sense that they are optimized before each branching. Sato [33], on the other hand, introduced *static* (SB) versions of Martins' MEAN bound and Lau's DMEAN bound. These bounds are weaker, but have the advantage that the longest path is calculated only once prior to any branching, instead of before each branching attempt. Especially the latter *directional* static bound leads to shorter run times. He furthermore proposed a network reduction method. Relocation arcs are generated, leading towards nodes-of-first-contact where the searcher has the earliest possibility of detection. This procedure may reduce the amount of branching attempts significantly.

Morin *et al.* [34] extended the path-constrained search effort allocation problem by adding an inter-region visibility criterion. In this new formulation, a searcher is able to make one observation in any of the cells adjacent to its position. They proposed a novel MILP, instances of which are solved using a commercial solver.

Hohzaki and Lida [35] considered an extension to the search effort allocation problem, where the decision of whether or not to look has to be made for each time period additionally. A look does not only yield a reward by increasing the probability of detection, but it also comes with a cost. In their approach to obtain an upper bound, the path-constraints as well as the integrality constraints on the searcher's position are relaxed. Additionally, they relax the integrality constraints on the look decisions. The relaxed problem is then solved using Washburn's forward and backward algorithm [31] and incorporated in a branch & bound

procedure.

Sato and Royset [36] extended Sato's original approach [33] by constraining the resources of the platform. They considered multiple constraints on consumption of resources such as time, fuel, and risk. In this problem, the glimpse probability is assumed to be history dependent. They propose a branch & bound method, with a Lagrangean directional static bound on the optimal probability of detection.

Heuristics for a single searcher

Besides exact methods, many researchers consider at least one heuristic method due to the intractability of the overall problem for large instances. Using a receding horizon approach was first proposed by Eagle [37] where he applied it to the dynamic programming method. Martins [18] introduced a heuristic method besides his exact approach. For each time period k, the search-path is extended by maximizing the expected probability of detection (ED) in times $k, ..., K$. The probability map, given that no detection has occurred, is updated after each extension. He uses this heuristic to effectively find a good initial solution for his branch & bound procedure. Washburn [19] also proposed a heuristic to find a nearly optimal solution, rather than exactly optimal. Instead of branching when the lower bound q is smaller than the best (lowest) non-detection probability $q*$ found so far, branching is carried out when $q + d < q*$. The last saved path will be within d of optimality. Increasing d from 0.01 to 0.02 already resulted in an order of magnitude decrease in runtime. Washburn considered this heuristic as perhaps being the best hope for solving search problems of a practical size. Thomas and Eagle [38] considered approximating bounds by procedures that are more easy to calculate. For their stationary target approximation all target states are assumed to be recurrent and for their no-learning approximation the transition matrix must have identical rows.

Sato [33] proposed a static bound heuristic (SBH) similar to Martins' ED heuristic, but adjusted with his own static bound. As a result, it performs only one longest path calculation. However, SBH is not available for huge problems because the time-expanded network becomes too large for computations.

Hong *et al.* [39] introduced the depth-l approximation, which is similar to Sato's SBH.

A time-expanded network is constructed and named depth-l network, because the cost of an arc is computed from the probability of non-detection over the previous l time steps. The second step of the algorithm is to solve the shortest path on this acyclic network, which requires linear time. Sato's SBH would be a depth-2 approximation in this setting. The computation costs increase exponentially with l.

Lanillos *et al.* [40] applied cross entropy optimization, in which samples are generated using multinomial sampling. They evaluated search effort allocation by several metrics: mean time to detection and time-discounted probability of detection. The resulting probability of detection of the search paths when optimizing those metrics is compared, where the latter dominates on almost all test instances.

Morin *et al.* [41] presented a constraint programming model. The proposed value selection heuristic simplifies the probability system by ignoring past non-detection events. At each time step, the accessible vertex is chosen that maximizes the total probability of detection in the remaining time, not conditioned on non-detection in previous time steps. The authors pointed out that the main advantage of this approach is expressiveness, because the model stays close to the natural problem formulation. Morin *et al.* [42] furthermore proposed a heuristic ant colony optimization method to solve their MILP with an inter-region visibility criterion. Simulations showed promising results for relatively large instances.

Stewart [43] considered the realistic case that a target may leave behind some evidence of its former presence. He proposed to optimize search by actively searching for a detectable trail left behind by a target. A moving-horizon rule and a heuristic simplification thereof were introduced.

Hong *et al.* [44] expanded their depth-l network [39] so that it accounts for a multiple-search-speed option, whereby a trade-off between speed and glimpse probability is made. The set of reachable cells within one time period is extended. This causes the number of arcs in each layer of the time-expanded network to increase significantly with the searcher's speed. They again used a longest path computation on the depth-l network.

Approaches for multiple searchers

Dell *et al.* [45] concentrated on solving the path-constrained search effort allocation problem with multiple searchers. They revisited Martins' procedure, which was found to exceed acceptable time for computation when applied to the multi-searcher problem. Several heuristics are presented, of which two exhibit reasonable run times and quality solutions. These heuristics are both centralized methods for cooperation. The first one extends Martins' ED heuristic for multiple searchers. It obtains solutions within 2% of the best known solution for each of the one- and two-searcher test problems considered. The second heuristic is a genetic algorithm, which was efficient on the three searcher problems, but no guarantee on the solution quality can be given.

To optimize the search effort allocation problem with multiple searchers, Sato [33] introduced a time-expanded configuration network. A configuration represents the combined locations of all searchers. He then applied his SB method to this network, which is intractable only for very small instances due to the large number of arcs. Therefore two heuristics were introduced as alternatives. The first is to apply the SBH on the time-expanded configuration network. Sato pointed out that this heuristic is not applicable to large problems because the time-expanded configuration network cannot be generated due to computational limits. The second heuristic is a cross entropy optimization method, which performs well on instances with as many as 30 searchers.

A method called the search and rescue optimal planning system is actually being used by the United States Coast Guard and is described by Kratzke *et al.* [2]. The technology behind it stands out compared to all other approaches with respect to the planning method. Here, a search-path is a pattern of equally spaced parallel paths within a rectangle. Instead of allocating a platform to a cell, the size, shape, and position of a rectangular subarea are optimized. The algorithm plans heuristically by placing a standard shaped rectangle on a cell with the highest probability, performs an accordion search [2] to find the proper size and location, and then fine-tunes it while minimizing overlap. A generalization of the multi-searcher problem was presented by Royset and Sato [46]. They introduced a novel MINLP for the problem with multiple targets, searcher deconfliction and target and location

dependent search effectiveness. The objective is to minimize the largest non-detection probability. The authors suggested using a cutting plane method, which results in a practical exact algorithm and is applicable for both conditionally deterministic and Markovian target models. This approach performs strongly compared to a branch & bound method, which fails on all test instances examined. A main advantage is the scalability as the number of searchers grows.

A MILP approach in which the planning of search jobs, rescue jobs, and recharging at stations is optimized for a team of UAVs, is proposed in [47].

3.1.2 Search trajectory optimization

The following approaches all aim for optimized search trajectories taking the flight kinematical properties of the aerial platform into account. The overview starts by describing the four publications considering trajectory planning for a single aerial platform, followed by approaches for a team of aerial platforms.

Approaches for a single aerial platform

The first approach explicitly considering flight kinematics was presented by Bourgault *et al.* [14]. The authors used a discrete-time non-linear velocity model, which takes a limited turn rate and a limited speed as input. Simulations show the effectiveness of a myopic and of a piecewise-constant control implementation. Furukawa *et al.* [16] propose a new method for a continuous representation of the probability map: the element-based method. For planning, they used a similar myopic approach to [14], however uniquely using the Kullback-Leibler divergence as the objective function. Collins *et al.* [48] were the first and, to the authors' knowledge, only ones to propose a planning method that plans for both a search trajectory and sensor schedule within the scope of moving target search. A unique dual probability map representation is used, combining the particle filter and grid-based representations. They furthermore considered a dynamic graph whose nodes are placed at high-reward locations and are dynamically updated at each time step. For a combined search trajectory and sensor schedule planning, they suggested two implementations. The

first of these is by evaluating search trajectories based on the cumulative probability of containment within the field of regard. The second is by evaluating search trajectories based on optimization of the sensor schedule on the path.

Approaches for a team of aerial platforms

The following approaches consider a cooperative search team, where historical and planned observations are exchanged and adapted to achieve a non-redundant search strategy.

Bourgault *et al.* [49] extended their work from [14] and suggested a coordinated approach for multiple heterogeneous searchers, which are considered to behave in a completely decentralized manner. They found that a myopic form of coordinated search strategy provides very sensible control solutions at low computational costs. Hu *et al.* [50] proposed a gradient based control law, where the optimal next-time sampling position is chosen to maximize the information gathered at time k. This is a myopic heuristic with a simple best response type of cooperation. Another contribution is their distributed probability map updating model, assuming that each platform only communicates with its neighbors within communication range. In this model, the glimpse probability varies as a function of sensor altitude. Xiao *et al.* [51] applied a hybrid method. Its main idea is that a virtual force type method helps the receding horizon method to improve searching efficiency and reduce the computational costs. This method avoids the problem of distinguishing between paths in low uncertainty areas arising in receding horizon methods.

An optimal team reward can be achieved either in a centralized manner, or in a decentralized manner using negotiation. Wong *et al.* [52] propose a centralized method for cooperation, where the cumulative team reward is maximized. To overcome the problem of distinguishing between paths in low uncertainty areas, they introduced a switching objective function to direct the searcher towards uncertainty areas. In this case, the objective is to minimize the distance to the nearest uncertainty area. Aiming for a team-optimal reward, while avoiding centralized computation, Bourgault *et al.* [53] propose a decentralized negotiation scheme. They use a block-iterative non-linear algorithm. This consists of iteratively fixing all paths, except for the ith path which is then optimized with respect to the fixed paths. At termination, the search plan satisfies Nash's equilibrium. A first method aiming

for optimal cooperation in a decentralized setting was presented by Delle Fave *et al.* [54]. They suggested using a max-sum algorithm to optimize the joint probability of detection, which operates over a factor graph. This approach scales linearly with the numbers of searchers. Cole [55] presented an entire cooperative unmanned aerial system architecture for information theoretic searching and tracking. In a search for an unknown number of targets, which is Poisson distributed, the expected number of detected targets is maximized.

The objective of minimizing Shannon's entropy is applied in simulations in [56]. A method based on enumeration of all possible strategies is used and compared against, amongst other strategies, the Zamboni strategy of fully covering the search area in rectangular search strips. A combined strategy for autonomous take off, search, and landing for a team of UAVs is presented in [57]. Here, a simple myopic algorithm is used for waypoint planning. When a UAV arrives at a crossroad, the next waypoint is set to the end of a road which is currently not searched by other UAVs and has the highest probability of containment.

3.1.3 Summary of the related literature

All discussed approaches are summarized in tables 3.1 and 3.2. In these tables, the most important properties of the approaches with respect to assumptions and solutions are presented in the columns. Each property is abbreviated to fit in a table cell. All properties are elucidated first, starting with the motion model of the target. When a certain property has not been specified in the associated article, the property is denoted by the entry (-).

The first property concerns the motion model of the **target**. In each model, targets are assumed to be independent. The generic (G) target model consists of a set of possible target tracks, where each possible track has a given probability of the target taking that track. The number of possible tracks grows exponentially in time and space. Monte Carlo sampling methods can be used to generate a finite set of possible tracks and their corresponding probabilities. The Markovian (M) target model is more restricted. It assumes that the process has the Markov property [15], so that the position of the target at time $k+1$ only depends on its current position at time k and is independent of those at any previous point in time. The random (R) target model is a special case of the Markovian target model. Here,

Reference	Year	Target Motion	Sensor Model	Composition	Cooperation	Objective	Additional Constraints	Additional Output	Solution Quality	Formulation	Algorithm Type	Horizon	Speed Trade-off	Platform Space
Search effort allocation														
Exact methods for single searcher														
Eagle [24]	1984	M	I/1/St	Si	-	PND	-	-	O	POMDP	DP	K	-	FD
Stewart [25]	1979	G	I/1/St	Si	-	PD	-	-	A	MINLP	BB	K	-	FD
Eagle and Yee [17]	1990	M	I/1/St	Si	-	PD	-	-	O	MINLP	BB	K	-	FD
Martins [18]	1993	M	I/1/St	Si	-	PD	-	-	O	MINLP	BB	K	-	FD
Washburn [19]	1998	M	I/1/St	Si	-	PND	-	-	O	MINLP	BB	K	-	FD
Lau *et al.* [32]	2008	M	I/1/St	Si	-	PD	-	-	O	MINLP	BB	K	-	FD
Sato [33]	2008	M	I/1/St	Si	-	PD	-	-	A	MINLP	BB	K	-	FD
Morin *et al.* [34]	2009	M	I/1/Gi	Si	-	PD	-	Or	O	MILP	CS	K	-	FD
Hohzaki and Lida [35]	1997	G	I/1/St	Si	-	PD	LR	Lo	O	MINLP	BB	K	-	FD
Sato and Royset [36]	2010	M	I/1/St	Si	-	PD	LR	Alt	O	MINLP	BB	K	RA	FD
Heuristics for a single searcher														
Eagle [37]	1984	M	I/1/St	Si	-	PND	-	-	A	POMDP	DP	K	-	FD
Martins [18]	1993	M	I/1/St	Si	-	PD	-	-	A	MINLP	LP	Myopic	-	FD
Washburn [19]	1998	M	I/1/St	Si	-	PND	-	-	A	MINLP	BB	K	-	FD
Thomas and Eagle [38]	1995	M	I/1/St	Si	-	PD	-	-	A	MINLP	BB	K	-	FD
Sato [33]	2008	M	I/1/St	Si	-	PD	-	-	A	MINLP	LP	Myopic	-	FD
Hong *et al.* [39]	2009	M	I/1/St	Si	-	PND	-	-	A	-	LP	K	-	FD
Lanillos *et al.* [40]	2012	M	I/1/St	Si	-	DTR	-	-	A	OC	CE	K	-	FD
Morin *et al.* [41]	2012	M	I/1/Gi	Si	-	PD	-	Or	A	CP	CS	K	-	FD
Morin *et al.* [42]	2010	M	I/1/Gi	Si	-	PD	-	Or	A	-	ACO	K	-	FD
Stewart [43]	1985	G	I/1/St	Si	-	PD	-		A	POMDP	BB	K	-	FD
Hong *et al.* [44]	2009	M	I/1/St	Si	-	PD	-	Sp	A	-	LP	K	RE	FD
Approaches for multiple searchers														
Dell *et al.* [45]	1996	G	I/1/St	He	Ce	PD	-	-	A	MINLP	Enum	Myopic	-	FD
Dell *et al.* [45]	1996	G	I/1/St	He	Ce	PD	-	-	O	MINLP	BB	K	-	FD
Sato [33]	2008	M	I/1/St	He	Ce	PD	-	Alt	A	MINLP	BB	K	-	FD
Sato [33]	2008	M	I/1/St	He	Ce	PD	-	Alt	A	MINLP	CE	K	-	FD
Sato [33]	2008	M	I/1/St	He	Ce	PD	-	Alt	O	MINLP	LP	K	-	FD
Kratzke *et al.* [2]	2010	G	I/N/St	He	Ce	PD	-	-	A	-	-	K	-	FD
Royset and Sato [46]	2010	G	I/1/St	He	Ce	PND	CA	-	O	MINLP	OA	K	-	FD
Royset and Sato [46]	2010	G	I/1/St	Ho	Ce	PND	CA	-	O	MILP	CS	K	-	FD
Royset and Sato [46]	2010	M	I/1/St	Ho	Ce	PND	CA	-	O	MILP	CS	K	-	FD
Lee and Morrison [47]	2015	D	I/1/St	Ho	Ce	PD	LR	AT	O	MILP	CS	K	-	FD

Table 3.1: Overview of search strategy planning approaches for search effort allocation.

Reference	Year	Target Motion	Sensor Model	Composition	Cooperation	Objective	Additional Constraints	Additional Output	Solution Quality	Formulation	Algorithm Type	Horizon	Speed Trade-off	Platform Space
Search trajectory planning														
Approaches for a single platform														
Bourgault *et al.* [14]	2006	M	I/N/St	Si	-	MTD	-	-	A	OC	Enum	Myopic	-	IC
Furukawa *et al.* [16]	2007	M	I/N/St	Si	-	KLD	-	-	A	OC	-	Myopic	-	IC
Collins *et al.* [48]	2007	M	I/N/Gi	Si	-	PD	-	Or	A	OC	-	K	RA	FD
Approaches for a team of platforms														
Bourgault *et al.* [49]	2003	M	I/N/St	He	Co	MTD	-	-	A	OC	-	Myopic	-	IC
Hu *et al.* [50]	2012	R	I/N/St	He	Co	PD	-	Alt	O	OC	GM	Myopic	-	IC
Xiao *et al.* [51]	2012	R	I/N/St	He	Co	PD	CA	-	A	OC	VF	Myopic	OF	IC
Wong *et al.* [52]	2005	M	I/N/St	He	Ce	PPD	-	-	A	OC	-	Myopic	OF	IC
Bourgault *et al.* [53]	2004	M	I/N/St	He	Ne	PD	-	-	A	OC	-	Myopic	-	IC
Fave *et al.* [54]	2010	M	I/N/St	He	Ne	PD	-	-	A	OC	-	Myopic	-	ID
Cole [55]	2009	M	I/N/St	He	Co	EN	-	-	A	OC	-	Myopic	-	IC
Peng *et al.* [56]	2015	R	-	He	Co	SE	-	-	O	OC	Enum	K	-	FD
Meng *et al.* [57]	2016	DC	I/1/St	He	Co	PD	-	-	O	-	-	Myopic	-	FD

Table 3.2: Overview of search strategy planning approaches for search trajectory planning.

the target moves to one of the cells that are adjacent to its current cell with equal probability. Such a target is also said to be *diffusing*. In the conditionally deterministic (CD) target model, a track merely depends on a stochastic variable such as the initial position or velocity. If this variable were known, the position of the target would also be known at any time in the future. Thus, the track of the target is deterministic, conditional on this stochastic variable. In case no assumption on the target motion can be made, the model of decaying certainty (DC) can be used, in which the probability of containment of a cell increases over time after it has been observed. A deterministic (D) model can be used when the target track is assumed to be known.

Each conducting searcher is equipped with a **sensor**. Sensors for detection are e.g. electro-optics/infra-red sensors. Several aspects of such sensors are important. These are sensor range, agility, and performance, and are therefore considered in detail within literature. These properties are presented in a model of the form range/performance/agility. The range of a sensor is either assumed to be restricted to the cell (1) which is currently

observed by its carrying platform, or restricted to a predefined range (N). We distinguish between perfect (P) and imperfect (I) sensors with respect to their performance. A perfect sensor detects a target within its field of view with probability one, whereas an imperfect sensor can miss the target. For sensor agility, we distinguish between stabilized (St) and gimballed (Gi) sensors.

The team **composition** can be made up of a single (Si) searching platform or a team of multiple platforms searching cooperatively. These platforms may differ in general characteristics concerning speed, turn radii, and sensor quality, in which case the team is heterogeneous (He) and it is homogeneous (Ho) when the team is uniform.

Cooperation within a team can be achieved in several ways. Using a centralized (Ce) control, an optimal search strategy is found by solving one overall optimization problem for the entire team. Computation is unfortunately often only tractable for very small instances. A decentralized approach is often more tractable and is also less prone to loss of communication. Cooperation means that observations and planned trajectories are exchanged and adapted to aim for optimal team search results. Two types of cooperation exist, where the difference lies in the balance between optimality of team results and computational costs. The cheapest type of cooperation from a computational point of view is coordination (Co). The predicted observations on the optimized search trajectories are exchanged between the platforms. The historical and predicted measurements are fused into the (shared or individual) probability map and the platform specific problems are optimized accordingly in sequence. Additional negotiation (Ne) between platforms improves the objective value, but increases computational costs and communication effort in general.

The quality of a search strategy must be defined in terms of an **objective**. Several types of objectives have been subject to studies. Which type to use depends on the goal of the search mission, and therefore they are incomparable in terms of effectiveness. The objective value is typically called the *search reward* within the context of target search. Maximizing the probability of detection (PD) is effective when a fixed time frame for the search mission is prescribed. The decision maker wishes to maximize the probability of detection cumulative over all time periods as in Equation (2.6). Minimizing the probability of non-detection (PND) (or *overlook* probability) is the inverse of maximizing the probability of detection.

Maximizing the time-discounted reward (TDR) discounts the reward of probability of detection in each time step. Alternatively, the mean time to detection (MTD) can be considered, or the expected number of detections (EN) in the case of a large number of targets. The information entropy (or Shannon's entropy) (SE) [58] is a measure of uncertainty in which higher values indicate higher uncertainty of the whereabouts of the target. The Kullback-Leibler divergence (KLD) [59] is another measure based on information, described as the relative entropy between the start and the end of the search mission.

Additional constraints with respect to the search strategy can exist. The trajectory of strategy can be subject to limited resources (LR), such that the search objective must be optimized while not exceeding the allowed resource consumption. Examples of resources accounted for in literature are: fuel, observations, and risk exposure. Some approaches explicitly account for collision avoidance (CA).

Additional outputs to the waypoints or allocations can be e.g. altitude (Alt) of the platform at a time step, whether or not to make an observation (Lo), the speed of the platform (Sp), its assignment to alternative tasks (AT) such as refueling or rescue, or the orientation of the sensor (Or).

A **speed trade-off** can be modeled in case the speed of the platform can be increased for quicker relocation at the cost of decreased glimpse probability. A few approaches propose methods to balance search reward and relocation speed. Three types of search-relocation trade-off methods can be distinguished. A switch between objective functions or a switching objective function (OF) can be used, switching between minimizing the distance to the nearest uncertainty area and maximizing the search reward. The decision between these objectives is based on an adaptive switching parameter, which may be platform specific. Another type of method is to extend the range (RE) of reachable cells or waypoints within a single time period to directly adjacent cells (direct neighbors). By increasing the speed, the neighborhood is expanded. This enlarges the number of possible actions at each time period, and thereby the number of possible search trajectories significantly. Relocation arcs (RA) can be used when the platform state space is modeled as a (time-expanded) network. It is then possible to determine a set of nodes-of-first-contact. These are the nodes at which the searcher can detect the target the earliest. The arcs leading from the searcher's

current position towards these nodes are called relocation arcs and take multiple time periods to traverse. To enable solution approaches, the platform state space is either an infinite continuous (IC) state space or a finite discrete (FD) state space in the form of a grid. Based upon the platform state space, the search problem is formulated in literature as either a partially observable Markov decision process (POMDP), a mixed integer linear program (MILP), a mixed integer non-linear program (MINLP), a constraint program (CP), or an optimal control (OC).

There are different **types of algorithms** to solve a search strategy planning problem formulation. The following exact algorithm types and metaheuristics have been applied in the reviewed literature. The exact algorithm types are dynamic programming (DP), branch & bound (BB), outer approximation (OA), and constraint programming (CP). Metaheuristics applied to the search effort allocation problem are genetic algorithms (GA), ant colony optimization (ACO), and cross entropy (CE). A virtual force field algorithm (VF) is a typical robot guiding algorithm. Other algorithm types are the gradient method (GM) and the longest path on a directed acyclic graph (LP) [55]. A few approaches do not propose a new algorithm, but deploy a commercial solver (CS) to solve the problem formulation. The least efficient algorithm type is to enumerate (Enum) over all feasible search strategies.

Finally, optimal (O) or approximate (A) **solution qualities** are proposed for a certain **planning horizon**. Planning for the full duration (K) of a predefined number of time periods K leads to (approximation of) globally optimal search strategies, when computation is tractable. A myopic (Myopic) approach chooses the platform state that is optimal for the next time period, possibly with a cost-to-go approximation [60].

3.2 Research gaps

The need of additional new methods to solve the three problems from chapter 2 is argued in the following. An overview of the properties of the methods as discussed in the previous section is summarized in Table 3.3. Four important properties are listed in this table. The first property is *anticipation*. A method allows for anticipation to the expected movement of the target when it yields an optimal solution over K time steps. Other methods are myopic

methods and merely react to the current state of the system. Another important property is accountability for the *flight kinematics* of the searching platforms. Furthermore, it may occur in certain situations that the platforms are subject to *limited resources*. In that case, the optimization algorithm must find an optimal solution subject to that constraint. Finally, a method with the *multi platform* property optimizes a search strategy consisting of one search trajectory for each platform. A row in this table suggests that methods exist in literature with a combination of the checked properties. The rows representing the research gaps show the combination of properties for methods that are necessary for solving the three problems from chapter 2.

Table 3.3: Research Gaps

	Anticipation	**Flight Kinematics**	**Limited Resources**	**Multi Platform**
Existing	✓			
Existing		✓		
Existing	✓		✓	
Existing	✓			✓
Research gap 1	✓	✓		
Research gap 2	✓	✓	✓	
Research gap 3	✓	✓	✓	✓

For the first problem of *search by a single platform under kinematical constraints*, any of the approaches for a single platform for search trajectory planning [14, 16, 48] is essentially applicable. However, only myopic algorithms have been proposed and myopic algorithms often lead to suboptimal search strategies. Research gap 1 from Table 3.3 shows the necessary properties for solving this problem. Therefore, the first aim of this dissertation is to propose an exact K-step-lookahead method for search by a single platform under kinematical constraints.

In the second problem of *search by a single platform under kinematical- and resource constraints*, resource constraints must be taken into account as well. The literature overview in tables 3.1 and 3.2 shows that a number of methods account for resource constraints when considering the search effort allocation problem for one searcher [35, 36], and for

multiple searchers [47] as well. However, the problem of optimizing a search trajectory under kinematical and resource constraints has, to the best of the author's knowledge, not been solved yet in literature. This research gap has number 2 in Table 3.3. Therefore, the second aim of this dissertation is to propose an exact K-step-lookahead method for search by a single platform under kinematical and resource constraints.

The final problem of *cooperative search by a team of heterogeneous platforms with shared resources* is significantly harder to solve from a computational point of view. A few search strategy optimization approaches for multiple aerial platforms under kinematical constraints have been proposed in literature. Due to the difficulty, mainly myopic algorithms have been proposed. Furthermore, numerous methods have been proposed for the search effort allocation problem with multiple searchers, both exact algorithms as well as heuristics. One of these approaches [47] takes resource constraints into account, but makes the simplifying assumption that the target movement is deterministic, which results in a trivial search problem. Research gap 3 from Table 3.3 shows the necessary properties for solving this problem. The third aim of this dissertation is therefore to propose an exact K-step-lookahead method for cooperative search by a team of heterogeneous platforms with shared resources and kinematical constraints.

Chapter 4

A method for search by a single platform under kinematical constraints

A prerequisite when optimizing search by an aerial platform is that the trajectory is physically feasible. This search trajectory optimization problem, as formally described in section 2.1, is solved in this chapter. A trajectory is considered to be physically feasible when the platform can actually reach a next waypoint in time. Therefore, properties such as its speed, altitude, maximum thrust, weight, aspect ratio, parasite drag coefficient, and its maximum lift must be considered. A constant search speed is a large advantage when compared with accelerating and decelerating during a search mission. In this chapter, a unified model for both fixed-wing and rotary-wing platforms is presented, as well as an MILP formulation for solving the problem under consideration. A generalization of an existing branch & bound algorithm is used to reduce computation times. In literature, branch & bound has been the most promising exact approach for solving unconstrained *path* optimization problems for moving target search [17–19, 25, 32, 33]. Here, a path is considered to be a sequence of cells to which the searcher is assigned for a period of time. The movements of the searcher within a cell are not prescribed. A trajectory, on the other hand, is a more detailed search plan. It consists of a sequence of waypoints and can be used for autonomous search. Therefore, the kinematical properties of the platform must be taken into account. The nature of both problems are, however, much alike. It appears therefore promising to generalize the branch & bound procedure for solving the problem under consideration in this chapter. The

proposed generalization allows to account for the kinematical properties of the platform.

The contributions of this chapter are threefold. By means of the proposed unified model for both fixed-wing and rotary-wing platforms it is possible to plan physically feasible trajectories for both types of platforms. An MILP is formulated to solve the moving target search problem under kinematical constraints and an existing branch & bound algorithm is generalized for efficiency. The applicability of this method is shown in computational experiments. The first two sections of this chapter have been published in [9, 11].

The remainder of this chapter is structured as follows. First, a uniform discrete model for flight kinematics is presented in section 4.1. The MILP formulation for solving the considered problem is described in section 4.2, followed by the branch & bound procedure for solving the MILP more efficiently in section 4.3. Finally, the computational experiments are described in section 4.4.

4.1 Uniform discrete model for flight kinematics

A search trajectory is described by a sequence of waypoints. A waypoint on a search trajectory prescribes a node v and a time k at which the platform is supposed to be at node v. In order to select a sequence of waypoints to obtain an optimal search trajectory, a predefined and finite set of waypoints must be provided. The possible set of waypoints is a part of the platform network, which preferably yields trajectories with natural flight kinematics.

In this chapter, a method to discretize the model is described in subsection 4.1.1 and how to create such a platform network is presented in subsection 4.1.2. Finally, a description of how to select the possible waypoints in such a way that the sequence describes a physically feasible trajectory is given in subsection 4.1.3.

4.1.1 Hexagonal grid sizing

The nodes of the platform network are proposed to be placed on the cell centers of a hexagonal grid. The considerations behind using a hexagonal grid to model the movements of the platform are twofold. The main advantage over e.g. a square grid is that the travel distances between neighboring nodes in turning flight and straight flight are closer together.

Usually in the case of time-dependent routing, the vehicle is at a node at a certain time step. Now if the distances between neighboring nodes are longer, the platform would need to accelerate and decelerate more in order to be in sync with the timeline of the target. The hexagonal grid therefore allows a more natural movement of the platform. The second advantage is that the number of nodes is finite on a finite search area. In summary, the hexagonal grid allows for a finite and discrete state space of the platform, which moreover has computational advantages over an infinite and continuous platform state space as in (2.2). Figure 4-1 depicts the possible continuous movements within the center cell, when coming from left below. Figure 4-1 depicts possible segments of continuous trajectories on a hexagonal grid. In the last figure, it is visible that it is not mandatory to fly over the cell centers, which will be used as waypoints for the trajectory. Figures 4-3 and 4-4 show a segment of a hexagonal grid including the options for consecutive waypoints.

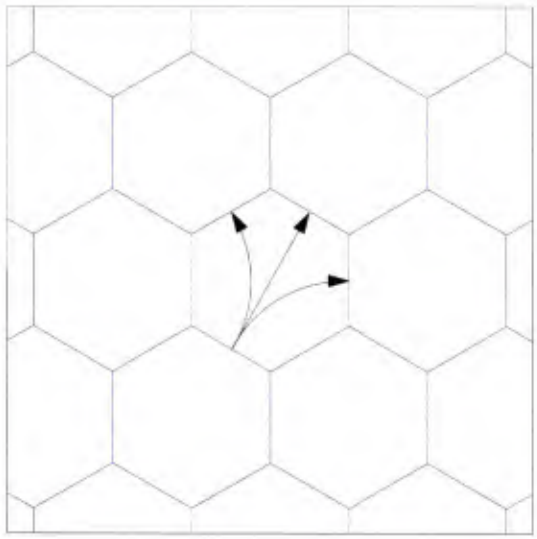

Figure 4-1: Possible continuous movements within a hexagonal cell, when coming from left below.

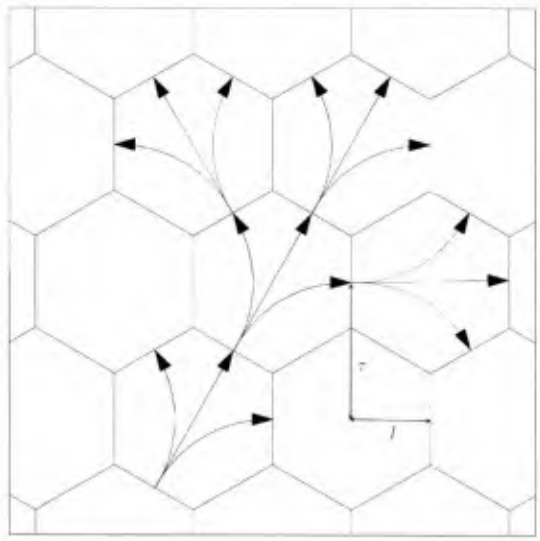

Figure 4-2: Possible segments of continuous trajectories on a hexagonal grid.

When constructing the hexagonal grid, the size of the hexagons must be determined based on the turning radius τ of the platform. The size of a hexagon is described by its inradius l. The inradius l of the hexagon as a function of the turning radius τ is given by:

$$l = \frac{\tau}{\sqrt{3}}. \tag{4.1}$$

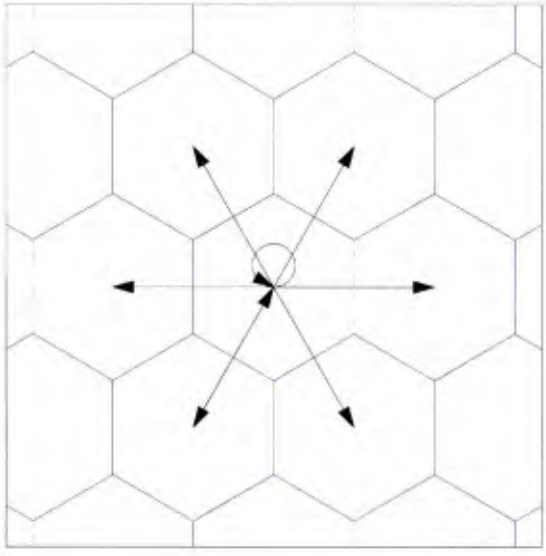

Figure 4-3: Seven options for a rotary-wing platform for consecutive waypoints on a hexagonal grid.

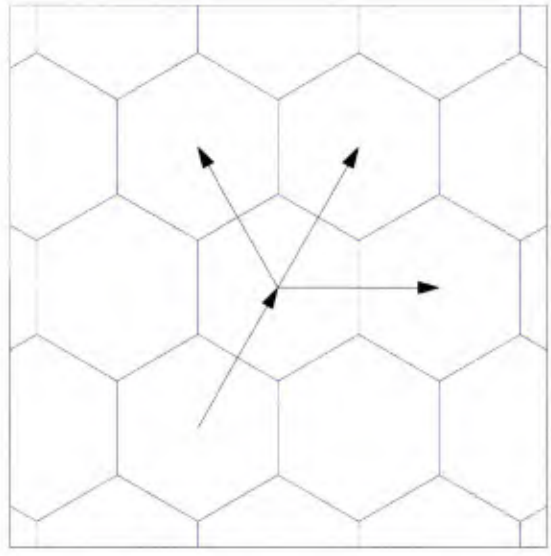

Figure 4-4: Three options for a fixed-wing platform for consecutive waypoints on a hexagonal grid.

This formula can be derived using basic geometry. The next question is how to derive the turning radius suitable for a search mission, which is answered in the following. First of all, the *turning radius* in meters is given as a function of the speed of the platform V and its *turn rate* $\dot{\psi} = (d\psi/dt)$ in radians per second [61]:

$$r = \frac{V}{\dot{\psi}}, \tag{4.2}$$

where the turn rate $\dot{\psi}$ is given as a function of the gravitational acceleration g in m/s^2, the load factor n and the speed of the platform V in m/s, as follows:

$$\dot{\psi} = \frac{g\sqrt{n^2 - 1}}{V}. \tag{4.3}$$

The load factor n is defined by the lift divided by the weight of the platform and equals the value 1 during straight flight and is larger than 1 during a turn. More details on the load factor and the derivation of the turn rate can be found in [62] on p. $467 - 470$. As a result, the turning radius can be given directly as a function of the gravitational acceleration g, the

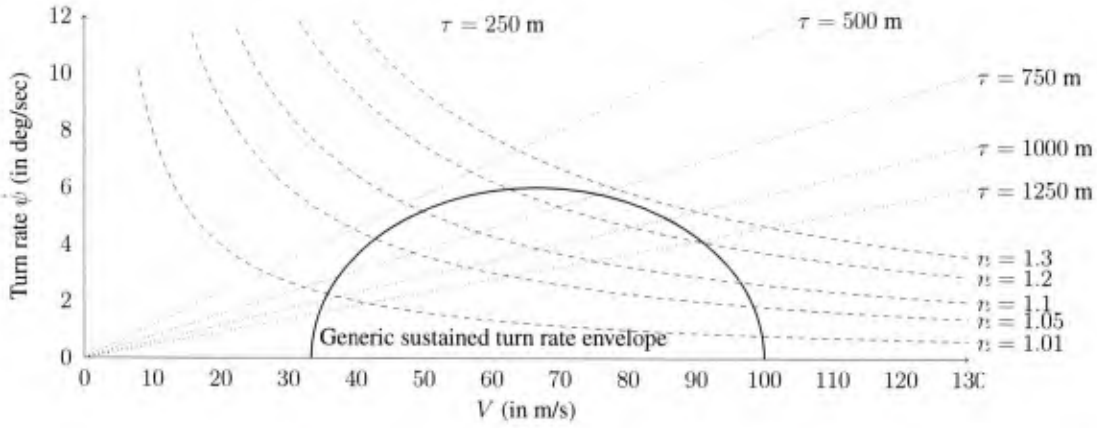

Figure 4-5: Turn rate and generic sustained turn rate envelope similar to [62].

load factor n and the speed of the platform V, as follows:

$$r = \frac{V^2}{g\sqrt{n^2 - 1}}. \tag{4.4}$$

From these formulas, it becomes clear that the turning radius results from a combination of the turn rate and speed. However, not all such combinations are possible. Only a range of combinations allows for maintenance of total energy. Possible combinations are shown in a sustained turn rate envelope. A *sustained turn* is a turn in which the platform maintains its speed and altitude. Figure 4-5 shows a *generic* sustained turn rate envelope. Actual envelopes must be derived specifically for a type of platform as described in [61, 62].

Figure 4-5 depicts a large amount of information. It shows the turn rate as a function of velocity for a fixed load factor using dashed lines by Equation (4.3). Furthermore it shows the relation between the velocity and the turn rate for a given turning radius using Equation (4.4) using dotted lines. By means of this figure, one can read the associating turning radius for a given combination of velocity and load factor. For example, with a speed of 70 m/s and a load factor of 1.2, the associating turning radius is approximately 750 m. More details can be found in [61] on p. 137. Finally, from this envelope, an optimal speed for searching needs to be determined, e.g. 180 knots (= 92.6 m/s) for the Airbus C295. In general, it can be determined by choosing a pareto optimal speed that maximizes the range while keeping the search effectiveness high [63] p. 6-21. Especially with visual search the search effectiveness decreases with the speed of the platform. The speed for maximum

range can be optimized using Brequets range equations [61, 62] and depends in general on aerodynamic and propulsion characteristics as well as on the weight of the aircraft. Optimal speed for range maximization are typically higher than when maximizing flight time. The finally chosen search speed can be used as a constant value for V. The speed and altitude of the platform are assumed to be constant during a search mission.

4.1.2 Platform network

In the following, a restricted network is constructed on the hexagonal grid on which the trajectory will be optimized. The complexity of the problem, however, grows exponentially in, among other things, the number of nodes on the network. Therefore, a fixed number of nodes are placed which are most likely to be contained in the optimal search trajectory, i.e., a set of connected nodes yielding the highest potential probability of detection. Selecting this subset is recognized to be equivalent to the maximum weight connected subgraph problem [64]. A greedy heuristic can be used for approximating this subgraph. The heuristic procedure basically selects feasible nodes in a greedy manner, until the desired number of nodes on the restricted network is reached. An example is shown in Figure 4-7. The resulting restricted network $G = (\mathcal{V}, A, R)$ is used for search trajectory planning and is defined by its nodes $\mathcal{V}$, adjacency matrix A, and reachability matrix R. These elements are described next.

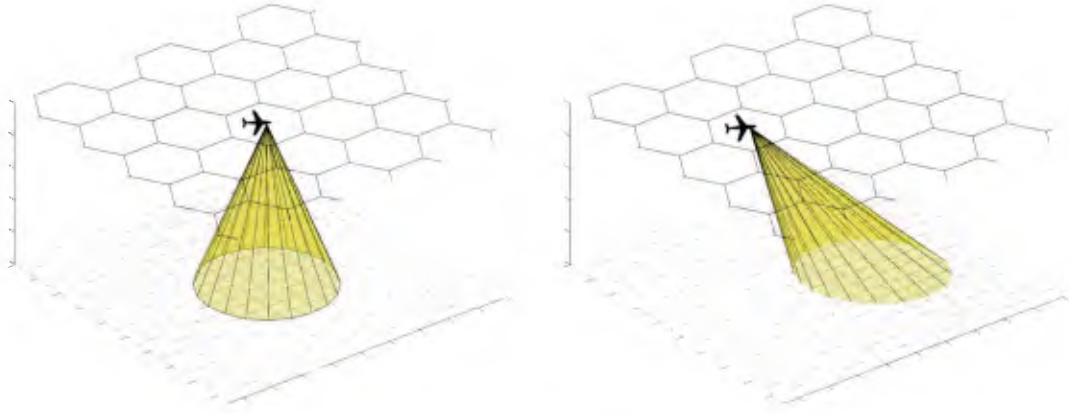

Figure 4-6: The heterogeneous grids; a hexagonal grid for the platform and a square grid for the target. The platform is able to observe multiple target grid cells at once.

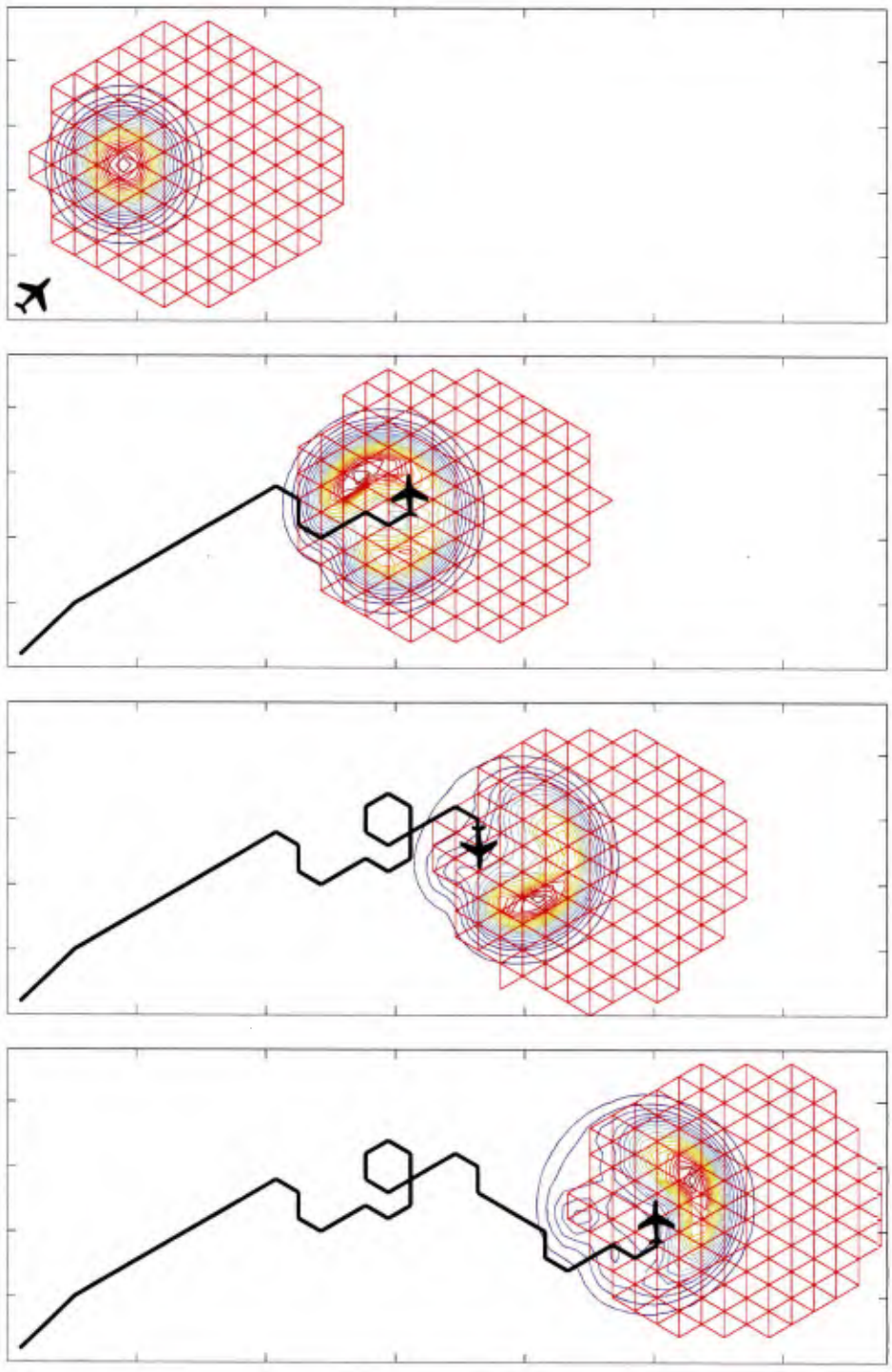

Figure 4-7: Examples of restricted networks. These figures show snapshots of a simulated search mission. Each snapshot is taken at the beginning of a planning stage. The network is placed only at those parts of the search area where the probability of containment at some point within the next K time steps is the highest. The depicted trajectory is already executed. The search maps have been updated according to the target's motion model, as well to the observations made along the search trajectory.

The binary adjacency matrix A holds information on the adjacency of the nodes. An entry $a_{v,v'}$ is 1 if node v is adjacent to node v' and 0 otherwise, where a pair of nodes is defined to be adjacent if and only if they are direct neighbors.

At the start of a search mission, the platform is usually at a position that is not represented by a node on the restricted network. Therefore, relocation arcs are added between the current position of the platform and each of the nodes. These arcs are represented by the binary reachability matrix R. An entry $r_{v,k}$ is 1 if node v is reachable in k time steps and 0 otherwise. If k' is the minimal time needed to reach node v, then $r_{v,k} = 1$ for all $k \geq k'$ and 0 otherwise. The use of these relocation arcs has three major benefits. First, it significantly reduces one of the dimensions of the model and thereby its complexity. Second, a drifting uncertainty area can be intercepted by a linear flight approach, due to the ability of anticipation of the target's predicted movements. The advantage of this aspect, when compared with a myopic one-step-lookahead method, is emphasized in a simulation shown in Figure 4-8. Third, this overcomes a typical problem faced by approaches with a static lookahead horizon of K_{static}; if all feasible trajectories of length K_{static} yield zero reward, no decision on the preceding flight trajectory can be made. By using these relocation arcs, there is at least one feasible search trajectory yielding positive reward.

4.1.3 A physically feasible trajectory on the network

The problem considered in this chapter is now restricted to finding a physically feasible trajectory on network G, that maximizes the probability of detection of the target. In this section, a physically feasible trajectory on network G is defined for rotary-wing and fixed-wing platforms. Let $\mathcal{V}(v_k) = \{v' \in \mathcal{V} : a_{v_k,v'} = 1\}$ be the set of nodes adjacent to node v_k and let $\mathcal{R}(k) = \{v \in \mathcal{V} : r_{v,k} = 1\}$ be the set of nodes reachable within k time steps. A trajectory can be specified by a sequence of nodes [65]. In this sequence, consecutive nodes must be adjacent and reachable in time, formally:

Definition 1 (Trajectory). *A trajectory is a sequence of nodes $(v_k)_{k\in\mathcal{K}}$, in which*

a) consecutive nodes in the sequence are adjacent nodes in the network, i.e. node v_{k+1} is in set $\mathcal{V}(v_k)$, for all $k \in \mathcal{K}$, and

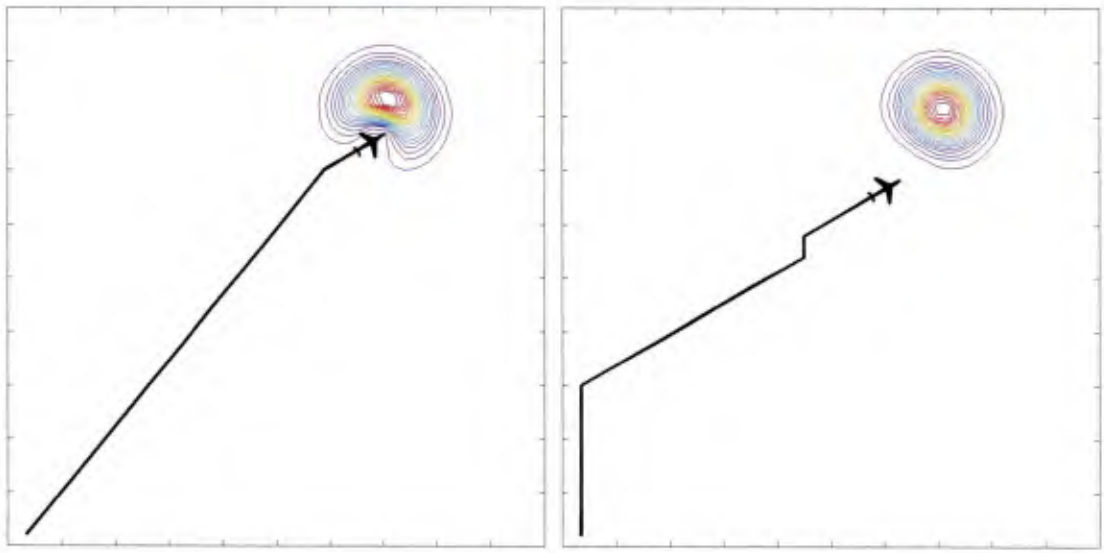

Figure 4-8: The K-step-lookahead aspect of the proposed method (left) allows for anticipation of the target's movement and finds a direct (shorter) approach towards the eastbound target. On the other side, a myopic method used in (right) acts greedy, resulting in a suboptimal trajectory. As a result, the platform in (left) has a higher probability of detecting the target in K time steps. This aspect intensifies as the approach distance increases.

b) each node v_k is reachable within k time steps, i.e. $v_k \in \mathcal{R}(k)$.

For a rotary-wing platform, a trajectory is inherently physical feasible. For a fixed-wing platform, however, an additional constraint is required for its trajectory to be physically feasible, because a fixed-wing platform can not hover, make sharp turns, or fly backwards. This is assured by prohibiting the next node from being adjacent to the previous node. Formally:

Definition 2 (Physically Feasible Trajectory)**.** *A trajectory $(v_k)_{k\in\mathcal{K}}$ is inherently physically feasible for rotary-wing platforms. A trajectory $(v_k)_{k\in\mathcal{K}}$ is physically feasible for fixed-wing platforms if and only if node v_{k+1} is in set $\mathcal{V}(v_k) \setminus \mathcal{V}(v_{k-1})$, for all $k \in \mathcal{K}$.*

Figures 4-9 and 4-10 show a visualization of the sets $\mathcal{V}(v_k)$ and $\mathcal{V}(v_k) \setminus \mathcal{V}(v_{k-1})$ from these definitions.

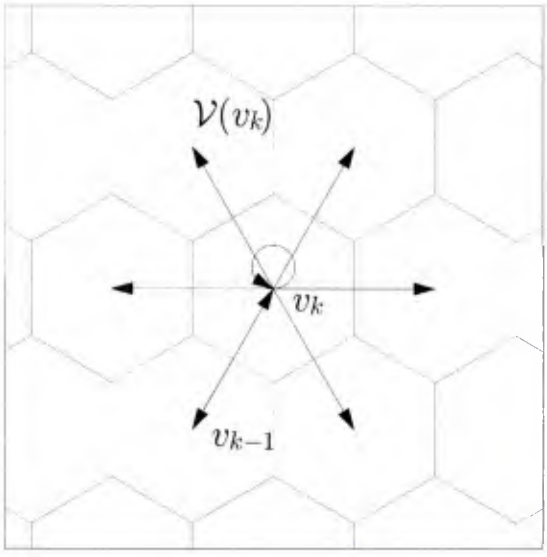

Figure 4-9: Physical feasibility for rotary-wing platforms. For rotary-wing platforms, the next node v_{k+1} must be adjacent to the current node v_k, i.e., $v_{k+1} \in \mathcal{V}(v_k)$ for the trajectory to be physically feasible.

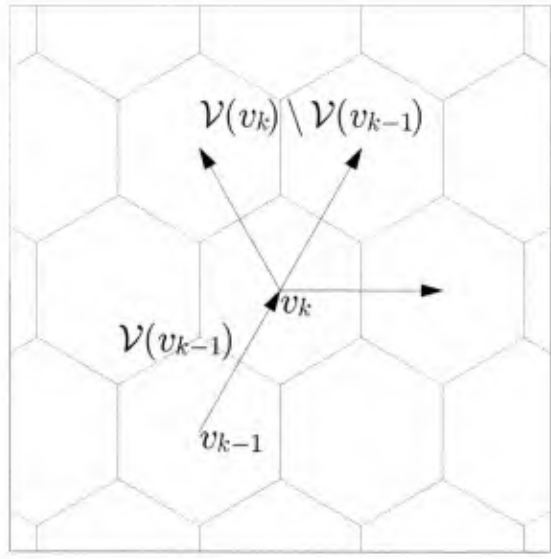

Figure 4-10: Physical feasibility for fixed-wing platforms. For fixed-wing platforms, the next node v_{k+1} must be adjacent to the current node v_k but not to the previous node v_{k-1}, i.e., $v_{k+1} \in \mathcal{V}(v_k) \setminus \mathcal{V}(v_{k-1})$ for the trajectory to be physically feasible.

4.2 Mixed integer linear programming formulation

An optimal physically feasible trajectory on the restricted network can be found by means of a mixed integer linear programming formulation (MILP). The proposed MILP contains three types of decision variables. The decision variable $z_v^k \in \{0,1\}$ is 1 if (k,v) represents a waypoint on the trajectory such that the platform is at node v at time k and 0 otherwise. The binary vector $\boldsymbol{z} = \{z_v^k \in \{0,1\} : k \in \mathcal{K}, v \in \mathcal{V}\}$ represents a trajectory on the reduced network. These main decision variables cause the combinatorial character of the problem. Additionally, two types of auxiliary decision variables are used for an accurate calculation of the objective function. Auxiliary decision variable $pd_c^k \in [0,1]$ represents the probability of detection in cell c at time k and auxiliary decision variable $pc_c^k \in [0,1]$ represents the probability of containment in cell c at time k. In the remainder of this section, decision variables are written in *italics*, whereas variables for input parameters are written in normal font. The input parameters in this problem are the glimpse probability $\mathrm{pg}_{k,v,c}$, the target

motion probability $\mathrm{d}_{c',c}$, the reachability of a node $\mathrm{r}_{v,k}$, the known initial probability of containment pc_c^1, and the node adjacency $\mathrm{a}_{v,v'}$. Finally, the input parameter ψ is introduced with the value

$$\psi = \begin{cases} 1 & \text{if the platform is } \textit{rotary}\text{-winged,} \\ 0 & \text{if the platform is } \textit{fixed}\text{-winged.} \end{cases}$$

A mixed integer linear program that yields a physically feasible trajectory on the restricted network that maximized the probability of detection can be formulated as follows:

$$\text{Maximize} \sum_{k=1}^{K} \sum_{c \in \mathcal{C}} pd_{k,c} \tag{4.5}$$

subject to

$$pd_{k,c} - \mathrm{pg}_{k,v,c} pc_{k,c} \leq 1 - z_{k,v} \qquad \forall k \in \mathcal{K}, \forall v \in \mathcal{V}, \forall c \in \mathcal{C} \tag{4.6}$$

$$pd_{k,c} - \sum_{v \in \mathcal{V}} \mathrm{pg}_{k,v,c} z_{k,v} \leq 0 \qquad \forall k \in \mathcal{K}, \forall c \in \mathcal{C} \tag{4.7}$$

$$pc_{k,c} - \sum_{c' \in \mathcal{C}} \mathrm{d}_{c',c} pc_{k-1,c'} + \sum_{c' \in \mathcal{C}} \mathrm{d}_{c',c} pd_{k-1,c'} = 0 \qquad \forall k \in \{2, \ldots, K\}, \forall c \in \mathcal{C} \tag{4.8}$$

$$pc_{1,c} = \mathrm{pc}_{1,c} \qquad \forall c \in \mathcal{C} \tag{4.9}$$

$$\sum_{v \in \mathcal{V}} z_v^k \leq 1 \qquad \forall k \in \mathcal{K} \tag{4.10}$$

$$\sum_{v \in \mathcal{V}} \sum_{k=1}^{K} z_v^k \left(1 - \mathrm{r}_{v,k}\right) = 0 \tag{4.11}$$

$$\sum_{v \in \mathcal{V}} z_v^k - \sum_{v \in \mathcal{V}} z_v^{k+1} \leq 0 \qquad \forall k \in \{1, ..., K-1\} \tag{4.12}$$

$$\sum_{v' \in \mathcal{V}} \left(1 - \mathrm{a}_{v,v'}\right) z_{v'}^{k+1} + z_v^k \leq 1 \qquad \forall k \in \{1, ..., K-1\}, \forall v \in \mathcal{V} \tag{4.13}$$

$$\sum_{v' \in \mathcal{V}} \mathrm{a}_{v,v'} z_{v'}^{k+1} + z_v^{k-1} \leq 1 + \psi \qquad \forall k \in \{2, ..., K-1\}, \forall v \in \mathcal{V} \tag{4.14}$$

$$z_{v_0}^0 + z_{v_{-1}}^{-1} = 2 \tag{4.15}$$

$$z_{k,v} \in \{0, 1\} \qquad \forall k \in \mathcal{K}, \forall v \in \mathcal{V} \tag{4.16}$$

$$pd_{k,c}, pc_{k,c} \in [0, 1] \qquad \forall k \in \mathcal{K}, \forall c \in \mathcal{C} \tag{4.17}$$

Here, the objective (4.5) is to maximize the expected probability of detection. The probability of detection $pd_{k,c}$ at time k and cell c must be calculated according to the formula (2.7). This is ensured by means of both constraints (4.6) and (4.7). The first constraint ensures accurate calculation of $pd_{k,c}$ depending on the visited node. The constraint (4.6) for the node that is visited at time k, say node v', is the most restrictive in its set. Because the value of $pd_{k,c}$ must be smaller or equal to each of the constraints in (4.6) it takes the value $\text{pg}_{k,v',c}pc_{k,c}$. However, in the case that none of the nodes is visited at a time k, all constraints in (4.6) are too lax. Constraint (4.7) ensures that in this case $pd_{k,c}$ is zero for that time k. Furthermore, the probability of containment in a cell at time k depends on all observations made up until time k as well as on the motion model of the target. Constraint (4.8) ensures that the probability of containment is updated according to formula (2.4) with $B = 1$. Finally, the decision variables $pc_{1,c}$ must be set to the initial known probability of containment, as ensured by constraint (4.9). The constraints so far ensure the accurate calculation of the probabilities for a given trajectory. The following constraints ensure that the binary vector $\boldsymbol{z}$ represents a physically feasible trajectory. First, a trajectory can contain at most one waypoint per time step, which is ensured by constraint (4.10) and it can only contain reachable waypoints, as ensured by constraint (4.11). In case the first waypoint is not immediately at time $k = 1$, then the platform is relocating. However, as soon as the search starts at arrival at the first waypoint at time k', the trajectory must contain waypoints at each time $k \geq k'$ in order to obtain a physically feasible trajectory, as ensured by constraint (4.12). The next constraint (4.13) ensures that the next node is adjacent to the current node according to definition (1). Constraint (4.14) ensures that the next node is *not* adjacent to the previous node according to definition (2) for fixed-wing platforms. This constraint is relaxed for rotary-wing platforms using the value 1 for input parameter ψ. Finally, in case the platform is not relocating but keeps searching, the constraint (4.15) ensures physical feasibility on the transition between consecutive trajectories. Here, two artificial waypoints $(0, v_0)$ and $(-1, v_{-1})$ are added at the start of the trajectory and match these to the last two waypoints of the previous trajectory by means of constraint (4.15). The artificial waypoint $(0, v_0)$ must be subject to the trajectory constraints (4.10), (4.12), and (4.13) and both artificial waypoints must be subject to the kinematical constraints (4.14) as well. These constraints are therefore additionally added

and ensure the physical feasibility of a trajectory that is planned in iterations.

The total number of decision variables, including the auxiliary decision variables, is of order $\mathcal{O}(K|\mathcal{V}||\mathcal{C}|)$. The number of possible physically feasible trajectories on the network G is of order $\mathcal{O}(|\mathcal{V}|3^{K-1})$ for fixed-wing platforms and $\mathcal{O}(|\mathcal{V}|7^{K-1})$ for rotary-wing platforms.

4.3 Branch & bound algorithm

In order to solve the problem more efficiently compared to solving the MILP (4.5)-(4.17) using a commercial branch & bound solver, an existing branch & bound algorithm is generalized in this section. The main idea of the branch & bound algorithm is described here. More details can be found in [17–19, 25, 32, 33]. The generalization with respect to flight kinematics occurs in the decomposition into subproblems, which is initially described in subsection 4.3.1. The generalization with respect to heterogeneous grids occurs in solving an upper bound to the original problem, which is described in subsection 4.3.2.

4.3.1 Decomposition into subproblems

The branch & bound procedure starts with initiation of the lower bound LB. The LB represents the actual probability of detection for the so far best trajectory found throughout the procedure. In order to avoid confusion between nodes on the platform network G and nodes on the branch & bound tree, the latter is referred to by *bb-node* in this chapter. Each layer on the tree corresponds to a time step $k \in \mathcal{K}$ and each bb-node on layer k' is associated with a unique partial trajectory with length $k' \leq K$. A priority queue $\mathcal{Q}$ of bb-nodes is initiated with a root-node representing an empty partial trajectory with length zero. Then, the bb-nodes in the priority queue are processed sequentially in order of priority, until the priority queue $\mathcal{Q}$ is empty. Priority is determined by the highest actual probability of detection of the partial path associated with its bb-node. Processing a bb-node first starts with solving a relaxation of the problem P_{UB} that is significantly easier to solve. Due to the fact that this problem is solved at every node in the branch & bound tree, it is shown in literature [19, 30] that lower computation time results in better performance than the tightness of the bound. Two upper bounds are described in the next subsection. If the upper

bound is smaller than the lower bound, no better solutions are possible further down the tree and the node can be pruned. Otherwise, there is a possibility that an improved solution exists further down in the tree. If the bb-node n is on the K^{th} layer, i.e. $n.k = K$, it is a leaf node and the corresponding partial trajectory $n.\boldsymbol{z}$ has length K. Hence, it is a full trajectory for which the actual probability of detection $f(n.\boldsymbol{z})$ can be determined in closed form. If $f(n.\boldsymbol{z})$ is larger than the best solution for far LB, then it is the new best trajectory $\boldsymbol{z}^*$ and the LB is updated to the corresponding actual probability of detection $f(n.\boldsymbol{z})$. If on the other hand the bb-node is not a leaf node, i.e. $n.k < K$, child bb-nodes are created through branching. Similar to the branching strategy in literature, branching is performed on the next possible waypoints. Because in [17–19, 25, 32, 33] the kinematical constraints of a fixed-wing platform are not taken into account, all adjacent waypoints are next possible waypoints. The proposed generalization is to create a child bb-node n' by extending the partial trajectory with node $v \in \mathcal{V}$ if and only if node v is adjacent to current node $n.v$ and not adjacent to the node $n.parent.v$ in which parent corresponds to the parent bb-node of bb-node n. Each created child bb-node is then added to the priority queue $\mathcal{Q}$. At termination of the branch & bound procedure, the optimal search trajectory is $\boldsymbol{z}^*$. This algorithm is presented in pseudo code in Algorithm 1.

Computational experiments using this algorithm are presented in section 4.4. Furthermore, the output of this algorithm while solving one specific instance of the search problem is presented in Appendix A.

4.3.2 Upper bounds

Two existing upper bounds are described and generalized for application to heterogeneous grids, the MEAN bound [18] and the STAT bound [36]. These bounds have been improved in [32] and [36] respectively by tightening the bound with very small increases in computational costs. The generalization is applied to the MEAN and STAT bound because of their simplicity. However, the generalizations made in this chapter can be applied analogously to the improved bounds for even shorter computation times. The MEAN bound was proposed for a moving target search problem in which both the platform and the target were modeled

Algorithm 1 Branch & bound for moving target search

Initiate lower bound $LB \leftarrow 0$
Initiate priority queue $\mathcal{Q} \leftarrow$ root bb-node
while $\mathcal{Q} \neq \emptyset$ **do**
  $n \leftarrow$ pull bb-node with highest priority from queue $\mathcal{Q}$
  $UB \leftarrow$ solve problem P_{UB} for upper bound
  if $UB \leq LB$ **then**
    Prune bp-node n
  else
    if $n.k = K$ **then**
      if $f(n.\boldsymbol{z}) > LB$ **then**
        $\boldsymbol{z}^* \leftarrow n.\boldsymbol{z}$
        $LB \leftarrow f(n.\boldsymbol{z})$
      end if
    else
      for each $v \in \mathcal{V} : a_{v,n.v} = 1$ **and** $\tilde{a}_{v',n.parent.v} = 1$ **do**
        $n' \leftarrow$ new node with $n'.v = v$ and $n'.k = n.k + 1$
        $\mathcal{Q} \leftarrow \mathcal{Q} \cup n'$
      end for
    end if
  end if
end while

to move over one shared grid. In the problem considered in this chapter, however, the platform observes multiple cells at the same time with variant glimpse probabilities as shown in Figure 2-2. The MEAN bound is generalized to account for such heterogeneity. This bound yields an upper bound of the actual expected probability of detection on the trajectory by summing over the probabilities of detection at each time step, neglecting observations made at other time steps. I.e., the probability map is updated according to formula (2.1) instead of to formula (2.4). The neglecting probability of detection $NPD_{k,v}$ at time k at node v is calculated according to the formula

$$NPD_{k,v} = \sum_{c \in \mathcal{C}} pc_{k,c} pg_{k,v,c}, \tag{4.18}$$

in which $pc_{k,c}$ is determined by formula (2.1). Then, a directed acyclic graph (DAG) $(\mathcal{N}, \mathcal{E})$ is created as described in [18, 32, 36]. A node on the DAG is referred to by *dag-node*. Each dag-node corresponds to a waypoint (k, v) and has the value $NPD_{k,v}$ assigned. The upper bound can now be solved by solving a longest path problem on the DAG [29], which runs in $\mathcal{O}(|\mathcal{N}| + |\mathcal{E}|)$ time. The STAT bound is essentially the same as the MEAN bound, with the sole difference that the STAT bound is calculated once for each bb-node on the branch & bound tree prior to any branching. Whereas the MEAN bound is calculated in each bb-node, the STAT bound remains unchanged (static) throughout the branch & bound procedure. The advantage of the MEAN bound is that it is tighter, because the observations made on the partial trajectory are taken into account when calculating $NPD_{k,v}$. However, recalculating $NPD_{k,v}$, constructing the DAG, and solving a longest path problem in each bb-node increases computation time. This is exactly the advantage of the STAT bound, as calculating $NPD_{k,v}$, constructing the DAG, and solving a longest path problem need to be executed only once. The disadvantage though, is that the upper bound is significantly less tight compared to the MEAN bound. Finally, computation of the upper bound in a leaf bb-node is done by computing the actual probability of detection of the partial trajectory, because the partial trajectory associated with a leaf bb-node is already a full trajectory. Hence, the actual probability of detection is the UB as well as the LB in that bb-node.

4.4 Computational experiments

By the means of an extensive computational test environment, the proposed model and corresponding branch & bound algorithm are tested for both effectiveness and efficiency in terms of computation time. The branch & bound algorithm with two existing upper bounds is benchmarked against solving the MILP formulation using IBM Cplex.

All tests were performed on an Intel(R) CoreTM i7-4810MQ CPU processor with 2.80 GHz and a usable memory of 15.6 GB. The test instances were generated by means of the author's instance generator developed in Matlab. The MILP is solved using IBM Cplex with default parameter settings and the branch & bound procedure as well as the algorithm for solving the longest path problem on the directed acyclic graph are implemented in Java.

The remainder of this section is structured as follows. First, the test-bed is described in subsection 4.4.1, followed by the simulation results in subsection 4.4.2.

4.4.1 Testbed

The testbed consists of a set of representative test instances as well as the algorithms for solving the instances. First, the search environment and the target properties are described, followed by the properties of the platform conducting the search. The algorithms are listed last.

The search area in all tests was modeled as a 30×30 square grid. Four target types have been taken under consideration; each target regardless of type is assumed to move to the north, east, south, or west with equal probability in each time step. The distinction between types comes from the probability that the target stays in its current cell. We consider targets that are assumed to stay in their current cell with $0.0, 0.2, 0.4$, and 0.6 probability. As a result, the probabilities to move to the north, east, south, or west are $0.25, 0.20, 0.15$, and 0.10 for the four types respectively.

For all targets regardless of type, the start position C_0 was bivariate normally distributed ($C_0 \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$), with $\boldsymbol{\mu} = [15, 15]$ and $\boldsymbol{\Sigma} = [5, 0; 0, 5]$. The glimpse probability $pg_{k,v,c}$ given platform position at node v at time k was calculated using the following formula:

$$pg_{k,v,c} = \begin{cases} 1 - e^{-W_{k,v,c} z_{k,v}} & \text{if the target is in cell } c \text{ at time } k, \\ 0 & \text{otherwise.} \end{cases} \tag{4.19}$$

with $W_{k,v,c} \geq 0$ being a measure of search effectiveness for cell c. The search effectiveness decreases with the Euclidean distance $||v - c||$ between cell c and the platform at node v as follows:

$$W_{k,v,c} = Q\left(||v - c||\right)^{-1}, \tag{4.20}$$

where Q is a sensor quality indicator.

Search is conducted by a single platform with turn radius $l = 1$ and each instance is solved three times, once using each of the sensor qualities $Q = 0.5, 1.0, 1.5$. The number of

nodes placed on the search area to create the network G is $|\mathcal{V}| = 45$ nodes. Instances are generated for each of the durations $K = 4, 6, 8, 10, 12, 14$. As a result, 72 test instances are generated. These instances are each solved using the proposed algorithms:

- **MILP**: Solving the MILP formulation (4.5)-(4.17) using IBM Cplex.
- **BB STAT**: The branch & bound algorithm in Algorithm 1 with the STAT bound.
- **BB MEAN**: The branch & bound algorithm in Algorithm 1 with the MEAN bound.

All algorithms find an optimal solution in terms of expected probability of detection. Therefore, the algorithms are compared based on computation time. The number of nodes processed in the branch & bound trees is the second performance measure for the branch & bound procedures. The results and an analysis thereof are presented in the next subsection.

4.4.2 Results and analysis

The results show that the proposed generalization of the branch & bound algorithm for moving target search trajectory optimization is indeed effective and efficient in solving the problem under consideration. The computational results are summarized in Table 4.1 for the target with 0.0 stay probability, in Table 4.2 for the target with 0.2 stay probability, in Table 4.3 for the target with 0.4 stay probability, and in Table 4.4 for the target with 0.6 stay probability. In these tables, the leftmost column contains the input parameters: stay probability, search duration K, and sensor quality Q. The computation time in milliseconds of the **MILP** algorithm is shown in the first column on the right, followed by the number of processed nodes and the computation times of the **BB STAT** and **BB MEAN** algorithms, respectively. In the rightmost column, the achieved maximal probability of detection is shown, which is equal for each algorithm. Since the possibility of multiple optimal trajectories exists, the resulting trajectories did differ in several instances. In the case that the computation time of the **MILP** algorithm exceeded 1.25×10^6 milliseconds, the computation was aborted and the entry "-" was listed as a result. The **BB STAT** and **BB MEAN** never reached this computation time, because the system ran out of memory long before. In the case of an out of memory exception, the result "-" was listed as well.

Table 4.1: Computational results for search for a target with 0.0 stay probability by a single platform under kinematical constraints.

Input			MILP	BB STAT		BB MEAN		
Stay prob.	K	Q	Time(ms)	#Nodes	Time(ms)	#Nodes	Time(ms)	PD
0.0	4	0.5	2527	1739	77	510	103	0.4217
0.0	4	1.0	6334	1836	70	699	101	0.5621
0.0	4	1.5	9918	1836	80	732	111	0.6241
0.0	6	0.5	61367	10060	525	2875	245	0.4783
0.0	6	1.0	164981	10126	530	3558	271	0.5946
0.0	6	1.5	191419	10126	490	3803	291	0.6402
0.0	8	0.5	708902	46928	3041	10186	961	0.5079
0.0	8	1.0	1188396	46952	3213	11960	1052	0.6054
0.0	8	1.5	1217087	46952	3294	11194	1026	0.6439
0.0	10	0.5	-	196316	15764	10186	3599	0.5215
0.0	10	1.0	-	196318	15418	32485	3654	0.6084
0.0	10	1.5	-	196318	15640	25558	3063	0.6445
0.0	12	0.5	-	-	-	90453	10990	0.5278
0.0	12	1.0	-	-	-	77574	9570	0.6093
0.0	12	1.5	-	-	-	47224	6446	0.6446
0.0	14	0.5	-	-	-	234486	35024	0.5309
0.0	14	1.0	-	-	-	155395	23319	0.6096
0.0	14	1.5	-	-	-	76859	11582	0.6447

When comparing the results of the **BB STAT** and **BB MEAN** algorithms, one sees that for the smallest instances with the random target the **BB STAT** algorithm converges quicker, even though more nodes are processed. For all other instances, the **BB MEAN** algorithm needs less computation time and requires less nodes to be processed as well. On instances with a longer search duration, i.e. $K \geq 12$ the **BB STAT** algorithm threw out of memory exceptions, whereas this happened fewer times for the **BB MEAN** algorithm and only when the target was of the conditionally deterministic type. When comparing the computation times of the **MILP** and **BB MEAN** algorithms, one can clearly see the advantage of **BB MEAN**. The computation times are much shorter for each of the instances and for a number of instances the **BB MEAN** algorithm converged in time whereas the **MILP** algorithm did not. An interesting observation, however, is that the **MILP** algorithm performs better when

Table 4.2: Computational results for search for a target with 0.2 stay probability by a single platform under kinematical constraints.

Input			MILP	BB STAT		BB MEAN		
Stay prob.	K	Q	**Time(ms)**	**#Nodes**	**Time(ms)**	**#Nodes**	**Time(ms)**	PD
0.2	4	0.5	5067	1962	105	683	60	0.4443
0.2	4	1.0	10392	2107	68	793	53	0.6134
0.2	4	1.5	23778	2107	66	898	58	0.6886
0.2	6	0.5	74590	11926	805	3099	308	0.5245
0.2	6	1.0	218456	12089	755	3898	394	0.6645
0.2	6	1.5	248507	12089	648	4396	488	0.7161
0.2	8	0.5	-	59764	4876	12218	1440	0.5669
0.2	8	1.0	-	59845	4975	16292	1875	0.6817
0.2	8	1.5	-	59845	4959	15662	1918	0.7222
0.2	10	0.5	-	263843	27017	44024	6570	0.5898
0.2	10	1.0	-	263853	25747	54805	8349	0.6882
0.2	10	1.5	-	263853	26305	43797	6570	0.7242
0.2	12	0.5	-	-	-	147210	26892	0.602
0.2	12	1.0	-	-	-	150418	27079	0.6907
0.2	12	1.5	-	-	-	99669	17773	0.7247
0.2	14	0.5	-	-	-	461450	96741	0.6085
0.2	14	1.0	-	-	-	364716	77060	0.6915
0.2	14	1.5	-	-	-	188569	39799	0.7248

the target is of the conditionally deterministic type, whereas the **BB STAT** and **BB MEAN** algorithm perform better when the target is of the diffusing type. An intuitive explanation of this is the following. It is known that both bounds base their reward on the expected probability of detection at node v at time k while ignoring the complicating observations made up until time k. In the case of a target randomly moving in four possible directions, the maximum probability of redundant observations in two consecutive time steps is only one-fourth. In the case of a deterministic moving target, on the other hand, the maximum probability of redundant observations in two consecutive time steps is one. Therefore, in the latter case more redundancy is likely to occur and hence the bound is weaker for this type of target. Furthermore, computational results in [32] and [36] show that the bounds perform less well on targets with a higher probability of staying in a cell. The same conclusion

Table 4.3: Computational results for search for a target with 0.4 stay probability by a single platform under kinematical constraints.

Input			MILP	BB STAT		BB MEAN		
Stay prob.	K	Q	**Time(ms)**	**#Nodes**	**Time(ms)**	**#Nodes**	**Time(ms)**	PD
0.4	4	0.5	4510	1970	117	698	70	0.4572
0.4	4	1.0	11135	2107	73	803	51	0.6293
0.4	4	1.5	21002	2107	65	901	58	0.7048
0.4	6	0.5	124000	11970	738	3219	385	0.5445
0.4	6	1.0	256021	12089	688	4033	538	0.6358
0.4	6	1.5	282480	12089	733	4528	547	0.7358
0.4	8	0.5	-	59809	5440	13140	1653	0.5924
0.4	8	1.0	-	59845	5184	18146	2139	0.7059
0.4	8	1.5	-	59845	5102	17768	2135	0.7436
0.4	10	0.5	-	263853	27026	52095	7924	0.6193
0.4	10	1.0	-	263853	26581	63673	9631	0.7142
0.4	10	1.5	-	263853	26218	56099	8307	0.7459
0.4	12	0.5	-	-	-	179000	33097	0.6343
0.4	12	1.0	-	-	-	187063	34670	0.7174
0.4	12	1.5	-	-	-	132335	23911	0.7466
0.4	14	0.5	-	-	-	-	-	-
0.4	14	1.0	-	-	-	-	-	-
0.4	14	1.5	-	-	-	-	-	-

can be drawn from the results in Tables 4.1-4.4. With increasing stay probability, the **BB MEAN** algorithm processes a higher number of nodes and requires a longer runtime before an optimal solution is found on all instances. When the stay probability exceeded 0.4, **BB MEAN** algorithm ran out of memory for $K = 14$, whereas this was no issue on the instances with a lower stay probability. Interestingly, the **MILP** algorithm needed increasing runtimes with increasing stay probabilities as well. Apparently, the general upper bound used by IBM Cplex is weaker on instances with increasing stay probabilities as well. Increasing stay probabilities did not only increase the runtimes, but also resulted in a higher expected probability of detection.

Finally, there appears to be no correlation between the sensor quality Q and the computation times of any of the algorithms. On the other hand, of course, a clear correlation exists

Table 4.4: Computational results for search for a target with 0.6 stay probability by a single platform under kinematical constraints.

Input			MILP	BB STAT		BB MEAN		
Stay prob.	K	Q	**Time(ms)**	**#Nodes**	**Time(ms)**	**#Nodes**	**Time(ms)**	PD
0.6	4	0.5	2947	1976	99	698	63	0.4706
0.6	4	1.0	8844	2107	69	813	51	0.6458
0.6	4	1.5	14894	2107	67	898	58	0.7214
0.6	6	0.5	84666	12002	798	3326	333	0.5657
0.6	6	1.0	202714	12089	823	4093	376	0.7085
0.6	6	1.5	298017	12089	791	4594	430	0.7567
0.6	8	0.5	-	59834	5278	14472	1773	0.6197
0.6	8	1.0	-	59845	5043	20120	2347	0.7325
0.6	8	1.5	-	59845	4985	21114	2526	0.7664
0.6	10	0.5	-	263853	26533	58858	8702	0.6515
0.6	10	1.0	-	263853	25488	70941	10584	0.7426
0.6	10	1.5	-	263853	25736	65904	9486	0.7693
0.6	12	0.5	-	-	-	223045	42979	0.6703
0.6	12	1.0	-	-	-	248601	50194	0.7469
0.6	12	1.5	-	-	-	183715	32525	0.7702
0.6	14	0.5	-	-	-	-	-	-
0.6	14	1.0	-	-	-	-	-	-
0.6	14	1.5	-	-	-	-	-	-

between the sensor quality Q and the probability of detection; a sensor of higher quality results in a higher probability of detection.

Chapter 5

A method for search by a single platform under kinematical- and resource constraints

In certain situations, the trajectory of the platform is subject to constraints on certain resources, as described formally in section 2.2. This chapter describes a proposed solution to this problem. The branch & bound procedure from the previous chapter is very efficient, however, when considering resource constraints, the upper bounds can not be calculated by solving a longest path problem on a directed acyclic graph. A non-polynomial time procedure is necessary to solve in each bb-node and the branch & bound procedure becomes intractable. Therefore, a novel method for solving this problem is proposed using Benders' decomposition [66]. The application of Benders' decomposition has shown to be very promising for various other vehicle routing problems, e.g. multidepot salesmen problems [67] and the traveling salesman problem with time windows [68]. Algorithms of this type have also been developed for *combined* vehicle routing problems with allocation [69, 70], assignment [71, 72], and scheduling [73]. Aircraft specific routing combined with crew scheduling is solved using Benders' decomposition in [74]. Furthermore, within the context of vehicle routing, several *hybrid* methods have been proposed where Benders' decomposition is exploited in combination with Lagrangean relaxation [75], constraint programming [76], and a genetic algorithm [77].

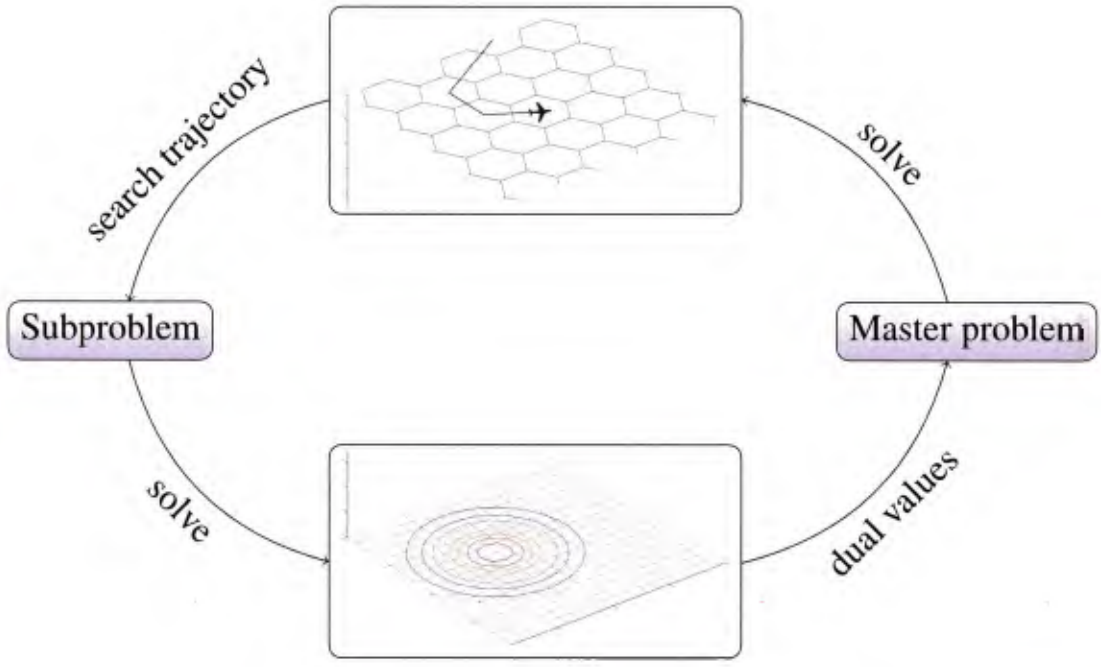

Figure 5-1: The Benders' decomposition approach for moving target search strategy optimization.

The main idea of the proposed method to find an optimal search trajectory is to disconnect the networks of the platform and the target. Figure 4-6 shows the connectivity of these networks due to the glimpse probabilities. When disconnecting the networks using the Benders' decomposition approach, two much smaller problems are obtained which are solved in iterations. The master problem is solved to obtain incumbent physically feasible search trajectories, and the subproblem is solved to obtain dual values which are used to generate a cut. The cut is added to the master problem, which is solved in order to obtain an incumbent physically feasible search trajectory. This process runs in iterations until convergence at which point an optimal solution has been found. This idea is depicted in Figure 5-1.

The contribution of this chapter consists of a Benders' decomposition algorithm to solve a problem of Markovian target search trajectory optimization under limited resources. First, the proposed novel mixed integer linear program (MILP) is described, which accounts for a heterogeneous state space for the target and platform, as well as for the resource constraints. Furthermore, a novel Benders' decomposition algorithm is proposed that solves an extensive formulation of the MILP more efficiently compared to solving the MILP in its compact

form using IBM Cplex. To the best of the author's knowledge, this is the first Benders' decomposition approach to solving a target search problem in general, for either *static* or *moving* targets, and with *finite* or *infinite* resources. We published an intermediate version of this approach in [12].

The remainder of this chapter is organized as follows. First, a compact MILP formulation is proposed in section 5.1, followed by its Benders' decomposition in section 5.2. An algorithm to solve the extensive formulation is presented in section 5.3. Finally, the computational experiments show the applicability and effectiveness of the proposed method in section 5.4.

5.1 Mixed integer linear programming formulation

The problem for moving target search trajectory optimization under kinematical and resource constraints can be formulated as the following MILP. It is largely equal to the MILP in (4.5)-(4.17) for moving target search trajectory optimization under kinematical constraints, with the additional resource constraints additionally included. The decision variables remain the same; the decision variable $z_{k,v} \in \{0,1\}$ takes the value 1 if the platform visits node v at time k and 0 otherwise. The auxiliary decision variable $pd_{k,c} \geq 0$ represents the probability of detection in cell c at time k. The auxiliary decision variable $pc_{k,c} \geq 0$ represents the probability of containment in cell c at time k. All decision variables are written in *italics* and input parameters are written in normal font.

$$\text{Maximize} \sum_{k=1}^{K} \sum_{c \in \mathcal{C}} pd_{k,c} \tag{5.1}$$

subject to

$$pd_{k,c} - \text{pg}_{k,v,c} pc_{k,c} \leq 1 - z_{k,v} \qquad \forall k \in \mathcal{K}, \forall v \in \mathcal{V}, \forall c \in \mathcal{C} \tag{5.2}$$

$$pd_{k,c} - \sum_{v \in \mathcal{V}} \text{pg}_{k,v,c} z_{k,v} \leq 0 \qquad \forall k \in \mathcal{K}, \forall c \in \mathcal{C} \tag{5.3}$$

$$pc_{k,c} - \sum_{c' \in \mathcal{C}} \text{d}_{c',c} pc_{k-1,c'} + \sum_{c' \in \mathcal{C}} \text{d}_{c',c} pd_{k-1,c'} = 0 \qquad \forall k \in \{2,\ldots,K\}, \forall c \in \mathcal{C} \tag{5.4}$$

$$pc_{1,c} = \mathrm{pc}_{1,c} \qquad \forall c \in \mathcal{C} \tag{5.5}$$

$$\boldsymbol{z} \in \boldsymbol{Z} \tag{5.6}$$

$$\sum_{k\in\mathcal{K}}\sum_{v\in\mathcal{V}} \mathrm{pr}_{k,v} z_{k,v} \leq \mathrm{T} \tag{5.7}$$

$$z_{k,v} \in \{0,1\} \qquad \forall k \in \mathcal{K}, \forall v \in \mathcal{V} \tag{5.8}$$

$$pd_{k,c}, pc_{k,c} \geq 0 \qquad \forall k \in \mathcal{K}, \forall c \in \mathcal{C} \tag{5.9}$$

The objective function (5.1) maximizes the expected probability of detection. The sets of constraints in (5.2) and (5.3) ensure calculation of $pd_{k,c}$ according to the formula (2.7). The set of constraints in (5.4) ensure calculation of $pc_{k,c}$ according to the formula (2.4) and the set of constraints in (5.5) ensure that the initial values of $pc_{1,c}$ is set according to the known initial probability of containment $pc_{1,c}$. Let $\boldsymbol{Z}$ be the set of binary vectors for the $z_{k,v}$ variables that yield physically feasible trajectories. The set of constraints (5.6) ensures the binary vector $\boldsymbol{z}$ to be a physically feasible trajectory. This abstract notation is used here instead of the extensive formulation in (4.10)-(4.15) in section 4.2. Finally, the constraint (5.7) ensures that the used resources on the trajectory described by $\boldsymbol{z}$ does not exceed the limit T. The total number of decision variables, including the auxiliary decision variables $pd_{k,c}$ and $pc_{k,c}$, is of order $\mathcal{O}(K|\mathcal{V}||\mathcal{C}|)$. The number of possible feasible trajectories on network G is of order $\mathcal{O}(|\mathcal{V}|3^{K-1})$ for fixed-wing platforms and of order $\mathcal{O}(|\mathcal{V}|7^{K-1})$ for rotary-wing platforms.

Figure 5-2 shows a search scenario in which one platform is modeled to search for one east moving target in an environment containing two risk areas. Here, the platform is resource constrained in the sense that its time exposed to risk is limited by 16.66% of the total mission time, i.e. $T = K/6$. The optimal strategy is obtained by solving the mixed integer linear program in (5.1)-(5.9).

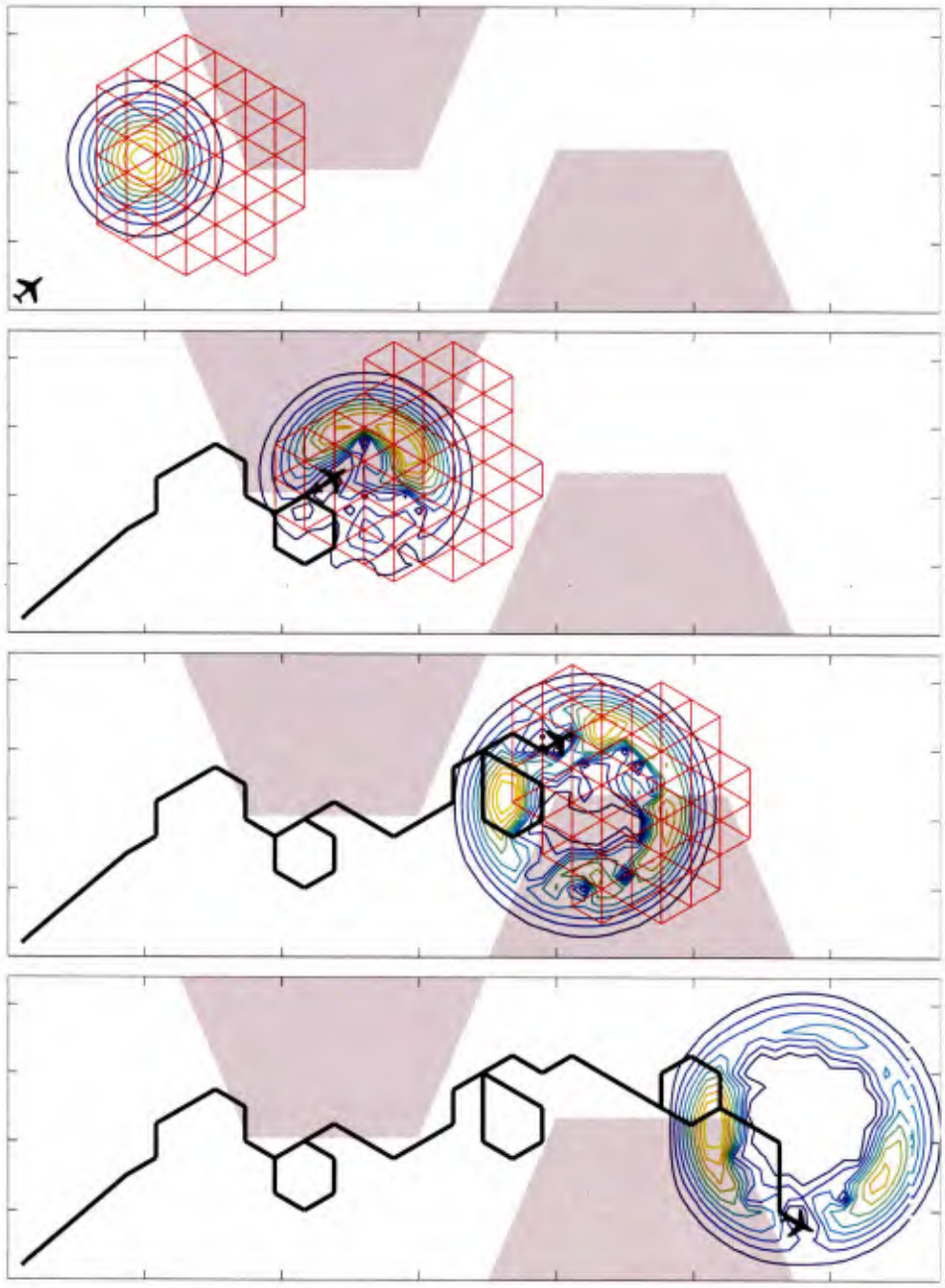

Figure 5-2: A search scenario with exposure to risk. The gray polygons represent areas in which the platform is exposed to risk. The platform is constrained to be exposed to risk at most for 16.66% of the total mission time.

5.2 Benders' decomposition

As the number of time steps or the number of nodes increases, the MILP becomes more difficult to solve. In order to decrease the computation time for solving this problem, the compact formulation (5.1)-(5.9) is decomposed into a pair of problems that can be solved more easily by applying Benders' decomposition. The resulting extensive formulation consisting of a subproblem is described in subsection 5.2.1, and a Benders' master problem is described in subsection 5.2.2.

5.2.1 Primal subproblem

The subproblem can be formulated as follows. First, recall Z to be the set of feasible trajectories on the platform network G. When fixing any binary vector $\bar{z} \in Z$, the trajectory constraints (5.6) and the resource constraint (5.7) become obsolete and can be dropped. The original problem reduces to the following *primal subproblem* in the $pc_{k,c}$ and $pd_{k,c}$ variables:

$$\text{Maximize} \sum_{k=1}^{K} \sum_{c \in \mathcal{C}} pd_{k,c} \tag{5.10}$$

Subject to

$$pd_{k,c} - \mathrm{pg}_{k,v,c} pc_{k,c} \leq 1 - \bar{z}_{k,v} \qquad \forall k \in \mathcal{K}, \forall v \in \mathcal{V}, \forall c \in \mathcal{C} \tag{5.11}$$

$$pd_{k,c} - \sum_{v \in \mathcal{V}} \mathrm{pg}_{k,v,c} \bar{z}_{k,v} \leq 0 \qquad \forall k \in \mathcal{K}, \forall c \in \mathcal{C} \tag{5.12}$$

$$pc_{k,c} - \sum_{c' \in \mathcal{C}} \mathrm{d}_{c',c} pc_{k-1,c'} + \sum_{c' \in \mathcal{C}} \mathrm{d}_{c',c} pd_{k-1,c'} = 0 \qquad \forall k \in \{2, ..., K\}, \forall c \in \mathcal{C} \tag{5.13}$$

$$pc_{1,c} = \mathrm{pc}_{1,c} \qquad \forall c \in \mathcal{C} \tag{5.14}$$

$$pd_{k,c}, pc_{k,c} \geq 0 \qquad \forall k \in \mathcal{K}, \forall c \in \mathcal{C} \tag{5.15}$$

The resulting program (5.10)-(5.15) contains no integer decision variables. Such a linear program is very easy to solve in general. This is of great importance due to the many times this subproblem is solved in an iterative procedure such as Benders'.

5.2.2 Dual subproblem

The dual of the subproblem must be formulated in order to define the cuts that need to be added to the master problem, as well as to define the set of extreme points for which it makes sense to add cuts. The dual of the primal subproblem (5.10)-(5.15) is derived over its Lagrangian. Let $\boldsymbol{\pi} = (\pi_{k,v,c} \geq 0 : k \in \mathcal{K}, v \in \mathcal{V}, c \in \mathcal{C})$, $\boldsymbol{\rho} = (\rho_{k,c} \geq 0 : k \in \mathcal{K}, c \in \mathcal{C})$, $\boldsymbol{\sigma} = (\sigma_{k,c} \in \mathbb{R} : k \in \{2, ..., K\}, c \in \mathcal{C})$, and $\boldsymbol{\tau} = (\tau_{1,c} \in \mathbb{R} : c \in \mathcal{C})$ be the dual variables associated with constraints (5.11), (5.12), (5.13), and (5.14) respectively. The Lagrangian L_{PS} of the primal subproblem (5.10)-(5.15) then amounts to:

$$
\begin{aligned}
L_{PS} = \sum_{k=1}^{K} \sum_{c \in \mathcal{C}} & pd_{k,c} \\
& - \sum_{k=1}^{K} \sum_{v \in \mathcal{V}} \sum_{c \in \mathcal{C}} \pi_{k,v,c} \left(pd_{k,c} - \mathrm{pg}_{k,v,c} pc_{k,c} - 1 + \bar{z}_{k,v} \right) \\
& - \sum_{k=1}^{K} \sum_{c \in \mathcal{C}} \rho_{k,c} \left(pd_{k,c} - \sum_{v \in \mathcal{V}} \mathrm{pg}_{k,v,c} \bar{z}_{k,v} \right) \\
& - \sum_{k=1}^{K} \sum_{c \in \mathcal{C}} \sigma_{k,c} \left(pc_{k,c} - \sum_{c' \in \mathcal{C}} \mathrm{d}_{c',c} pc_{k-1,c'} + \sum_{c' \in \mathcal{C}} \mathrm{d}_{c',c} pd_{k-1,c'} \right) \\
& - \sum_{c \in \mathcal{C}} \tau_{1,c} \left(pc_{1,c} - \mathrm{pc}_{1,c} \right)
\end{aligned} \tag{5.16}
$$

For a reformulation hereof, the indicator function $\mathbb{1}$ is used such that $\mathbb{1}_{\{x\}} = 1$ if x is true and 0 otherwise. The reformulation then leads to:

$$
\begin{aligned}
L_{PS} = & \sum_{k=1}^{K} \sum_{c \in \mathcal{C}} pd_{k,c} \left(\sum_{v \in \mathcal{V}} \pi_{k,v,c} + \rho_{k,c} + \mathbb{1}_{\{k<K\}} \sum_{c' \in \mathcal{C}} \sigma_{c'}^{k+1} d_{c,c'} - 1 \right) \\
& + \sum_{k=1}^{K} \sum_{c \in \mathcal{C}} pc_{k,c} \left(\sum_{v \in \mathcal{V}} \mathrm{pg}_{k,v,c} \pi_{k,v,c} - \mathbb{1}_{\{k \neq 1\}} \sigma_{k,c} + \mathbb{1}_{\{k<K\}} \sum_{c' \in \mathcal{C}} \sigma_{c'}^{k+1} d_{c,c'} - \tau_{1,c} \right) \\
& + \sum_{k=1}^{K} \sum_{c \in \mathcal{C}} \sum_{v \in \mathcal{V}} \pi_{k,v,c} \left(1 - \bar{z}_{k,v} \right) + \sum_{k=1}^{K} \sum_{c \in \mathcal{C}} \rho_{k,c} \left(\sum_{v \in \mathcal{V}} \mathrm{pg}_{k,v,c} \bar{z}_{k,v} \right) \\
& + \sum_{c \in \mathcal{C}} \tau_{1,c} \mathrm{pc}_{c}^{1}
\end{aligned} \tag{5.17}
$$

From which the dual of the primal subproblem (5.10)-(5.15) is derived, resulting in the following *dual subproblem*:

$$\text{Minimize} \sum_{k=1}^{K} \sum_{c\in\mathcal{C}} \sum_{v\in\mathcal{V}} \pi_{k,v,c}\left(1-\bar{z}_{k,v}\right) + \sum_{k=1}^{K} \sum_{c\in\mathcal{C}} \rho_{k,c} \left(\sum_{v\in\mathcal{V}} \text{pg}_{k,v,c}\bar{z}_{k,v} \right) + \sum_{c\in\mathcal{C}} \tau_{1,c}\text{pc}_c^1 \quad (5.18)$$

subject to

$$\sum_{v\in\mathcal{V}} \pi_{k,v,c} + \rho_{k,c} + \mathbb{1}_{\{k<K\}} \sum_{c'\in\mathcal{C}} \sigma_{c'}^{k+1} d_{c,c'} \geq 1 \qquad \forall k \in \mathcal{K}, \forall c \in \mathcal{C} \quad (5.19)$$

$$\sum_{v\in\mathcal{V}} \text{pg}_{k,v,c}\pi_{k,v,c} - \mathbb{1}_{\{k\neq 1\}}\sigma_{k,c} + \mathbb{1}_{\{k<K\}} \sum_{c'\in\mathcal{C}} \sigma_{c'}^{k+1} d_{c,c'} - \tau_{1,c} \leq 0 \qquad \forall k \in \mathcal{K}, \forall c \in \mathcal{C} \quad (5.20)$$

$$\pi_{k,v,c}, \rho_{k,c} \geq 0 \qquad \forall k \in \mathcal{K}, \forall v \in \mathcal{V}, \forall c \in \mathcal{C} \quad (5.21)$$

$$\sigma_{k,c}, \tau_{1,c} \in \mathbb{R} \qquad \forall k \in \{2,\ldots,K\}, \forall c \in \mathcal{C} \quad (5.22)$$

This dual subproblem is an alternative formulation of the primal subproblem (5.10)-(5.15). The sets of constraints (5.19)-(5.22) define a polyhedron. The extreme points of this polyhedron, as well as the objective function in (5.18), are used to formulate the Benders master problem as described in the next subsection.

5.2.3 Benders' master problem

The master problem is solved to obtain an incumbent physically feasible trajectory. Therefore, the trajectory constraints (5.6) as well as the resource constraints (5.7) are included. However, the constraints (5.2)-(5.5) for accurate calculation of the corresponding expected probability of detection are obviated here, because the accurate expected probability of detection is determined by solving the subproblem (5.10)-(5.15). The original model (5.1)-(5.9)

is reformulated as the following *Benders master problem*:

$$\text{Maximize } y_0 \tag{5.23}$$

subject to

$$\sum_{k=1}^{K}\sum_{c\in\mathcal{C}}\sum_{v\in\mathcal{V}}\pi_{k,v,c}\left(1-z_{k,v}\right)+\sum_{k=1}^{K}\sum_{c\in\mathcal{C}}\rho_{k,c}\left(\sum_{v\in\mathcal{V}}\mathrm{pg}_{k,v,c}z_{k,v}\right)$$
$$+\sum_{c\in\mathcal{C}}\tau_{1,c}\mathrm{pc}_c^1\geq y_0 \quad \forall(\boldsymbol{\pi},\boldsymbol{\rho},\boldsymbol{\tau})\in P_\Delta \tag{5.24}$$

$$\boldsymbol{z}\in\boldsymbol{Z} \tag{5.25}$$

$$\sum_{k\in\mathcal{K}}\sum_{v\in\mathcal{V}}\mathrm{pr}_{k,v}z_{k,v}\leq \mathrm{T} \tag{5.26}$$

$$z_{k,v}\in\{0,1\} \quad \forall k\in\mathcal{K},\forall v\in\mathcal{V} \tag{5.27}$$

$$y_0\in\mathbb{R} \tag{5.28}$$

Here, let $\boldsymbol{\Delta}$ be the polyhedron defined by constraints (5.19)-(5.22), and let P_Δ denote the set of extreme points of $\boldsymbol{\Delta}$. Each extreme point represents a physically feasible trajectory on the network G. Polyhedron $\boldsymbol{\Delta}$ contains no rays because the subproblem is feasible for each $\boldsymbol{z}\in\boldsymbol{Z}$. Instead of maximizing the expected probability of detection, the free decision variable $y_0\in\mathbb{R}$ is introduced to be maximized. The decision variable y_0 is restricted by a cut for each extreme point in P_Δ in (5.24). The cardinality of P_Δ becomes very large with increasing size of the search area and search duration. The large corresponding set of constraints in (5.24) results in the intractability of the problem. However, because not all constraints are active in an optimal solution, it suffices to work with a restricted subset of constraints. A procedure for efficiently selecting such a restricted subset of constraints is known as a Benders' algorithm and is described in the next section for this specific problem.

5.3 Benders' algorithm

The Benders' algorithm is used to find and add meaningful constraints to the RMP in iterations. Such a process is also known as *row generation*. The algorithm starts with an empty set $P_\Delta = \emptyset$. An optimal physically feasible trajectory $\boldsymbol{z}^*$ that maximizes the free variable y_0^* is found by solving the master problem (5.23)-(5.28). Physically feasible trajectory $\boldsymbol{z}^*$ is fixed in the subproblem (5.10)-(5.15), which is solved with two results; the actual expected probability of detection PD corresponding to physically feasible trajectory $\boldsymbol{z}^*$ and the values of dual variables $(\boldsymbol{\pi}, \boldsymbol{\rho}, \boldsymbol{\tau})$ associated with constraints (5.11), (5.12), and (5.14) respectively. Because the primal subproblem is feasible for each $\boldsymbol{z}$, the values of the dual variables $(\boldsymbol{\pi}, \boldsymbol{\rho}, \boldsymbol{\tau})$ determine an extreme point of P_Δ. The dual values are used to generate a cut which is added to the master problem in (5.24). The master problem is again solved to find an incumbent physically feasible trajectory $\boldsymbol{z}^*$. This iterative procedure continues until its optimal value equals the optimal value of the primal subproblem, i.e. until $y_0^* = PD$. The value of y_0^* represents an upper bound and PD represents a lower bound. Therefore, when $y_0^* = PD$ the procedure converges and hence an optimal solution is found. This algorithm is presented in pseudo code in Algorithm 2.

Algorithm 2 Benders' decomposition algorithm for moving target search

1: Initiate $P_\Delta = \emptyset$
2: **while** $PD < y_0^*$ **do**
3: $\quad (\boldsymbol{z}^*, y_0^*) \leftarrow$ solve master problem
4: $\quad$ Fix $\boldsymbol{z}^*$ in subproblem
5: $\quad PD, (\boldsymbol{\pi}, \boldsymbol{\rho}, \boldsymbol{\tau}) \leftarrow$ solve subproblem
6: $\quad P_\Delta \leftarrow P_\Delta \cup (\boldsymbol{\pi}, \boldsymbol{\rho}, \boldsymbol{\tau})$
7: **end while**

This algorithm can be implemented in two ways when the master problem is solved using a generic branch & bound solver. In the *classical* implementation, the branch & bound tree is set up each time the master problem is solved. In the *modern* implementation, the branch & bound tree is set up only once [78]. Each time an integer solution is found in a node on the branch & bound tree, a callback function initiates the execution of the subproblem to find the dual values $(\boldsymbol{\pi}, \boldsymbol{\rho}, \boldsymbol{\tau})$ and a cut is added dynamically to the branch & bound tree. The

modern type of implementation reduces the overall computation time significantly, because all nodes on the branch & bound tree are processed once at most, whereas in the classical implementation parts of the tree are solved redundantly. In order to benchmark the classical and modern implementation of the proposed Benders' decomposition algorithm, both have been implemented and used in the computational experiments as described in the following section. Furthermore, the output of this algorithm while solving one specific instance of the search problem is presented in Appendix B.

5.4 Computational experiments

In this section, computational experiments are presented to show the applicability of the proposed model and the reduced computation times of the modern Benders' decomposition algorithm. Both the classic and modern version of the Benders' decomposition algorithm are benchmarked against solving the compact formulation using IBM Cplex.

The experiments were performed on an Intel(R) CoreTM i7-4810MQ CPU processor with 2.80 GHz and a usable memory of 15.6 GB. The instances were generated using the author's instance generator written in Matlab. The original MILP formulation and both the classic and modern version of the Benders' decomposition algorithm were coded in Java and solved using IBM Cplex with default parameters.

In the following, the test instances of the search problem as proposed in subsection 5.4.1 are presented first, followed by the results and a concise analysis thereof in subsection 5.4.2.

5.4.1 Testbed

The testbed exists as a selection of representative test instances as well as the three algorithms to solve each of these instances. First, a description is given of the target properties of the two target types under consideration, followed by a description of the properties of the platform. Finally, a summarizing overview of the algorithms is listed.

The general properties of the instances are similar to those from the previous chapter and are repeated for a complete description: The search area in all tests was modeled as a 30×30 square grid.

Analogous to the tests in the previous chapter, four target types have been taken under consideration; each target regardless of type is assumed to move to the north, east, south, or west with equal probability in each time step. The distinction between types comes from the probability that the target stays in its current cell. We consider targets that are assumed to stay in their current cell with $0.0, 0.2, 0.4$, and 0.6 probability. As a result, the probabilities to move to the north, east, south, or west are $0.25, 0.20, 0.15$, and 0.10 for the four types respectively.

For all targets regardless of type, the start position C_0 was bivariate normally distributed ($C_0 \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$), with $\boldsymbol{\mu} = [15, 15]$ and $\boldsymbol{\Sigma} = [5, 0; 0, 5]$. Analogous to the test instances in the previous chapter, the glimpse probability $pg_{k,v,c}$ given platform position at node v at time k was calculated using the following formula:

$$pg_{k,v,c} = \begin{cases} 1 - e^{-W_{k,v,c} z_{k,v}} & \text{if the target is in cell } c \text{ at time } k, \\ 0 & \text{otherwise.} \end{cases} \tag{5.29}$$

with $W_{k,v,c} \geq 0$ being a measure of search effectiveness for cell c. The search effectiveness decreases with the Euclidean distance $||v - c||$ between cell c and the platform at node v as follows:

$$W_{k,v,c} = Q \left(||v - c|| \right)^{-1}, \tag{5.30}$$

where Q is a sensor quality indicator. Instead of comparing the results for various sensor qualities, the results for various resource limits T are compared in this chapter. In order to obtain a clear view on the effect that limited resources have on a search problem, a representative subset of the instances from the previous chapter is considered as proposed in section 4.4 of the previous chapter. To be precise, all instances are considered for which the sensor quality was $Q = 1.0$. The instances for which the sensor quality was $Q = 0.5$ or $Q = 1.5$ were omitted to prevent redundancy in the tests.

Search is conducted by a single platform with turn radius $l = 1$. Each instance is solved four times, each time with a varying resource limit T. The number of nodes placed on the search area to create the network G is $|\mathcal{V}| = 45$ nodes. Instances are generated for each of the search durations $K = 4, 6, 8, 10, 12$. As a result, 80 test instances are generated and

solved using the following three algorithms:

- **MILP**: Solving the compact formulation (5.1)-(5.9) using IBM Cplex.
- **Benders' Classic**: Solving the extensive formulation using the proposed Benders' algorithm in the classic implementation.
- **Benders' Modern**: Solving the extensive formulation using the proposed Benders' algorithm in the modern implementation.

All three algorithms find optimal solutions to the problem. Therefore, the main performance measure is computation time. The number of cuts necessary to reach convergence in the Benders' algorithms are compared as well. Finally, the maximal expected probability of detection is denoted for each instance once, because each algorithm finds a trajectory with the optimal (maximal) expected probability of detection. The results are presented in the next subsection.

5.4.2 Results and analysis

The computational results are summarized in Table 5.1 for the target with 0.0 stay probability, in Table 5.2 for the target with 0.2 stay probability, in Table 5.3 for the target with 0.4 stay probability, and in Table 5.4 for the target with 0.6 stay probability. The results suggest that the proposed Benders' decomposition algorithm yields reduced computation times on most of the test instances. In these tables, the input parameters for the stay probability, search duration K, and resource limit T are listed in the leftmost columns. On the right hand side, the first column shows the computation time in milliseconds of the **MILP** algorithm, followed by the number of added cuts and computation time of the **Benders' Classic** algorithm and those of the **Benders' Modern** algorithm. Finally, the resulting expected probability of detection in shown. In some cases, the computation time of the **Benders' Classic** algorithm exceeded 0.65×10^7 ms, in which case the execution was aborted and the symbol "$-$" was denoted as result in Tables 5.1 - 5.4.

First, the computation times of the **Benders' Modern** algorithm are compared with the **Benders' Classic** algorithm. It is obvious that the modern implementation is a huge

Table 5.1: Computational results for search for a target with 0.0 stay probability by a single platform under kinematical- and resource constraints.

Input			**MILP**	**Benders' Classic**		**Benders' Modern**		
Stay prob.	K	T	**Time(ms)**	**# Cuts**	**Time(ms)**	**# Cuts**	**Time(ms)**	PD
0.0	4	0.75	1661	5	842	8	270	0.4775
0.0	4	0.80	2454	47	9869	53	1167	0.4890
0.0	4	0.85	7288	191	98836	194	3102	0.5080
0.0	4	0.90	5894	432	337852	439	6825	0.5249
0.0	6	1.05	4077	5	1829	8	671	0.5378
0.0	6	1.10	17271	61	64928	67	3654	0.5469
0.0	6	1.15	30695	315	752372	321	14493	0.5579
0.0	6	1.30	65453	-	-	3316	138142	0.5850
0.0	8	1.30	11344	0	1719	2	980	0.5646
0.0	8	1.35	51417	98	511269	106	13072	0.5733
0.0	8	1.40	126650	485	5098603	493	47977	0.5802
0.0	8	1.55	829045	-	-	9495	956172	0.5976
0.0	10	1.55	90481	56	807349	63	16080	0.5818
0.0	10	1.60	303200	-	-	526	120086	0.5879
0.0	10	1.65	916971	-	-	2363	453850	0.5936
0.0	10	1.75	3917211	-	-	21354	4487834	0.6015
0.0	12	1.70	88941	9	68129	13	17856	0.5849
0.0	12	1.75	364062	-	-	246	144223	0.5900
0.0	12	1.80	5195084	-	-	1849	589425	0.5946
0.0	12	1.85	6220395	-	-	8852	2804249	0.5988

improvement compared to the classic implementation, even though slightly more cuts were necessary for convergence of the upper and lower bound. The **Benders' Classic** algorithm became intractable with increasing K and T, whereas the **Benders' Modern** algorithm converged within the given time on all instances. For all algorithms, it is evident that the computation times increased with an increasing resource limit T. A straightforward explanation is the increased number of feasible trajectories, increasing the cardinality of the problem. When comparing the proposed **Benders' Modern** algorithm with the **MILP** algorithm, the strength of the **Benders' Modern** algorithm is clear. On most instances the computation time of the proposed algorithm is much shorter. However, for less restricted

Table 5.2: Computational results for search for a target with 0.2 stay probability by a single platform under kinematical- and resource constraints.

Input			MILP	Benders' Classic		Benders' Modern		
Stay prob.	K	T	Time(ms)	# Cuts	Time(ms)	# Cuts	Time(ms)	PD
0.2	4	0.80	2070	7	1216	9	519	0.5096
0.2	4	0.85	8163	69	28212	76	3366	0.5342
0.2	4	0.90	10119	205	143351	211	8328	0.5527
0.2	4	0.95	10135	530	552901	541	22259	0.5703
0.2	6	1.15	17580	21	13985	27	3526	0.5965
0.2	6	1.20	55304	115	179593	120	16260	0.6085
0.2	6	1.25	87969	399	1318123	411	51030	0.6190
0.2	6	1.30	106815	999	5507074	1048	135266	0.6294
0.2	8	1.45	37018	12	21748	16	6278	0.6337
0.2	8	1.50	145067	126	789058	135	39921	0.6437
0.2	8	1.55	324794	-	-	576	165040	0.6508
0.2	8	1.60	642882	-	-	1706	524948	0.6576
0.2	10	1.75	206984	56	965857	61	42089	0.6592
0.2	10	1.80	1388995	65	1520575	447	309359	0.6652
0.2	10	1.85	1596988	-	-	1892	1209457	0.6699
0.2	10	1.90	4608400	-	-	5990	3771354	0.6745
0.2	12	1.95	135830	0	20717	2	11965	0.6608
0.2	12	2.00	530450	64	5018647	103	140433	0.6715
0.2	12	2.05	2255540	-	-	904	933109	0.6742
0.2	12	2.10	-	-	-	4235	4434052	0.6790

problems with a larger resource limit T the **MILP** algorithm solves faster.

An interesting observation is that the **Benders' Classic** and the **Benders' Modern** algorithms do not obviously perform better on the instances with a target with a low stay probability compared to their performance on instances with a target with a high stay probability, whereas the branch & bound procedures in the previous chapter actually do perform better on the instances with such a target. The reason is that the **Benders' Classic** and the **Benders' Modern** algorithms do not use a relaxation of the problem that depends on ignoring observations made in the calculation of an upper bound on the expected probability of detection, in contrast to the STAT and MEAN bounds used in the branch & bound

Table 5.3: Computational results for search for a target with 0.4 stay probability by a single platform under kinematical- and resource constraints.

Input			MILP	Benders' Classic		Benders' Modern		
Stay prob.	K	T	Time(ms)	# Cuts	Time(ms)	# Cuts	Time(ms)	PD
0.4	4	0.85	3419	31	6506	36	1647	0.5370
0.4	4	0.90	7789	119	61169	129	5597	0.5560
0.4	4	0.95	10600	312	259776	318	12949	0.5733
0.4	4	1.00	9082	630	769217	635	25793	0.5909
0.4	6	1.20	17973	15	10238	20	2745	0.6137
0.4	6	1.25	47626	95	149593	102	14940	0.6263
0.4	6	1.30	70332	322	940948	329	47288	0.6371
0.4	6	1.35	98751	883	4378370	891	123429	0.6488
0.4	8	1.55	50086	34	103284	40	12819	0.6624
0.4	8	1.60	144215	157	1166013	166	51151	0.6683
0.4	8	1.65	389128	-	-	682	217630	0.6768
0.4	8	1.70	786664	-	-	1769	553280	0.6830
0.4	10	1.85	92277	11	65791	16	16914	0.6817
0.4	10	1.90	224963	128	4428205	135	90470	0.6894
0.4	10	1.95	1142410	-	-	745	498350	0.6932
0.4	10	2.00	4943160	-	-	2461	1666534	0.6978
0.4	12	2.15	387696	18	438647	23	54757	0.6959
0.4	12	2.20	1185154	-	-	305	360446	0.7009
0.4	12	2.25	5902910	-	-	1639	1733159	0.7042
0.4	12	2.30	-	-	-	6109	6401503	0.7074

procedure.

Another observation drawn from the results is that the expected probability of detection increases with an increasing resource limit T. It is at this point possible as well to compare these results with the expected probability of detection in the case with unlimited resource in Tables 4.1-4.4 in the previous chapter. Furthermore, analogous to the results in the previous chapter, the expected probability of detection increases with increasing stay probability. In other words, targets that are assumed to regularly move to a different cell are less likely to be detected.

Furthermore, the proposed **Benders' Modern** algorithm found optimal solutions for

Table 5.4: Computational results for search for a target with 0.6 stay probability by a single platform under kinematical- and resource constraints.

Input			MILP	Benders' Classic		Benders' Modern		
Stay prob.	K	T	Time(ms)	# Cuts	Time(ms)	# Cuts	Time(ms)	PD
0.6	4	0.90	8488	57	20218	63	3276	0.5502
0.6	4	0.95	7597	171	108802	179	9341	0.5769
0.6	4	1.00	8082	450	440602	456	21234	0.5958
0.6	4	1.10	8180	680	881013	1031	47071	0.6283
0.6	6	1.25	12828	14	8951	17	2831	0.6319
0.6	6	1.30	35971	74	86843	80	11514	0.6453
0.6	6	1.35	52582	248	628642	253	39847	0.6563
0.6	6	1.40	77289	718	3036011	733	111013	0.6572
0.6	8	1.60	35903	1	2940	3	1332	0.6784
0.6	8	1.65	64149	39	126670	44	15471	0.6371
0.6	8	1.70	155721	170	1375552	176	62245	0.6959
0.6	8	1.75	406504	170	1330145	668	242731	0.7016
0.6	10	1.95	64450	0	4631	2	2255	0.7062
0.6	10	2.00	133393	24	208234	29	21086	0.7138
0.6	10	2.05	274560	168	6123228	174	127461	0.7192
0.6	10	2.10	883413	-	-	817	527331	0.7227
0.6	12	2.30	211033	0	19856	2	14728	0.7237
0.6	12	2.35	281487	35	1298356	41	66233	0.7287
0.6	12	2.40	1197773	-	-	349	469974	0.7327
0.6	12	2.45	-	-	-	1624	2266222	0.7349

each of the instances, as long as T is small enough, as opposed to the algorithms in in the previous chapter.

A final remark can be made regarding the correlation between the stay probability of the target and the expected probability of detection: When comparing the expected probability of detection (PD) in the right most column of each of the Tables 5.1 - 5.4, it is clear that the expected probability of detection increases with increasing stay probability of the target.

Chapter 6

A method for cooperative search by a team of heterogeneous platforms with shared resources

When a team of platforms is employable for search, a search strategy needs to be optimized consisting of one cooperative trajectory for each platform. The problem of planning a search strategy for a team of heterogeneous platforms with shared resources that maximizes the expected probability of detection is formally described in section 2.3. This chapter describes a novel branch & price algorithm for solving this problem. The branch & price algorithm is an established process for solving large scale mixed integer programs [79–81]. It is shown to be especially powerful when the underlying sub-systems have a special combinatorial structure, e.g. a shortest path in a network flow problem [82, 83], a pairing in a crew scheduling problem [84] or a pattern in a cutting stock problem [85]. Typical for problems effectively solved by a branch & price algorithm is that the set of subsystems have a common reward function or shared resources [86]. When relaxing the so-called *linking* constraints the problem decomposes into a number of subproblems, one for each subsystem, which are more tractable to solve. The underlying structure of the huge mixed integer problem studied in the work at hand is basically a longest path problem on a directed acyclic graph [29], and the common reward is expressed in expected probability of detection. Therefore, solving the Dantzig-Wolfe reformulation of this problem by a branch & price algorithm appears

promising. Furthermore, the results of branch & price algorithms applied to a wide range of problems are shown to be highly promising in literature. Due to its success, this solution process is still an important subject of research, and improvements are under continued development [87], so that future improvements of the algorithms proposed in this chapter are possible and likely to result in an even larger reduction of computation times.

The structure of the multiple platform search problem is depicted in Figure 6-1. The hexagonal grid of the platforms are each connected to the square grid of the target, due to the glimpse probability. The main idea of the solution method in this chapter is to disconnect the grids using a Dantzig-Wolfe decomposition approach. This idea is unique in the literature on search strategy optimization. Applying Dantzig-Wolfe decomposition for search strategy

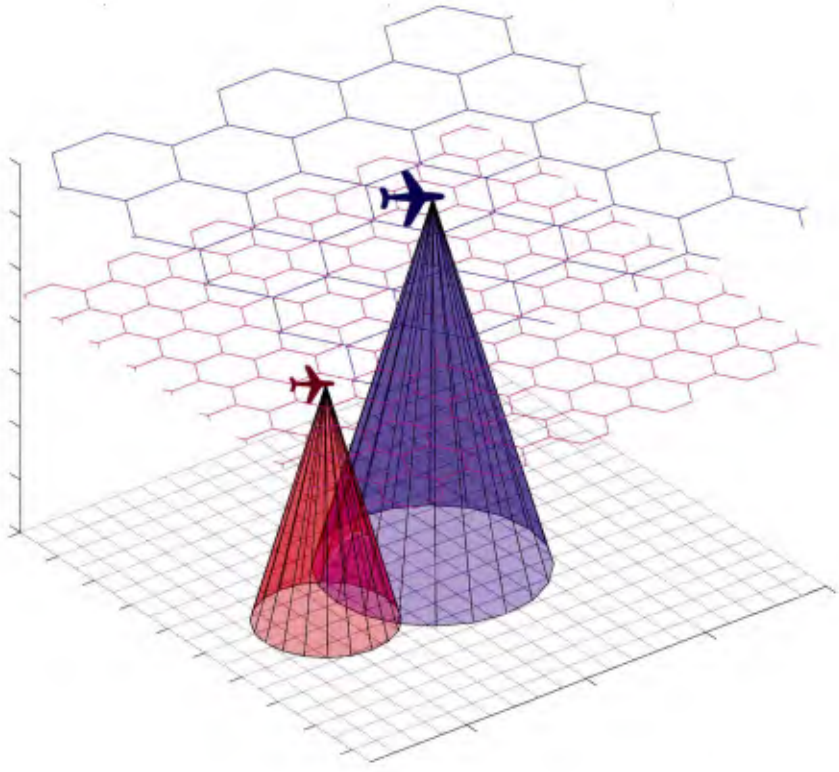

Figure 6-1: Heterogeneous grids for multiple platforms with variant turn radii.

optimization results in a decomposition of a single difficult problem into one master problem and several subproblems, one for each platform. The remaining problems are much easier and can be solved in iterations to find an optimal search strategy. The subproblems can even be solved in parallel. This idea is visualized diagrammatically in Figure 6-2.

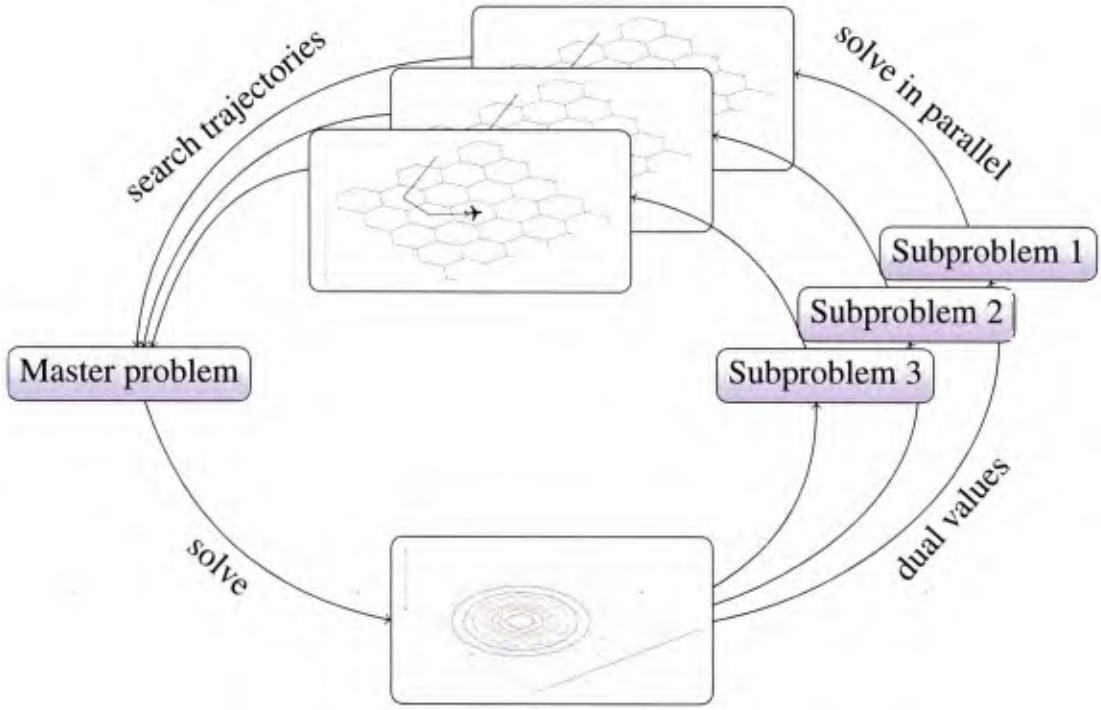

Figure 6-2: The Dantzig-Wolfe decomposition approach for moving target search strategy optimization.

The main contributions of this chapter are as follows. A linear upper bound to the non-linear original problem is proposed, which is much tighter compared to an existing linear upper bound [13]. Simulations show that the resulting optimality gap is reduced significantly in all test instances. Furthermore, to the best of the author's knowledge, this method is the first branch & price approach for solving a target search optimization problem in the literature on both *static* and *moving* target search. It results in faster runtimes compared to solving the compact formulation using a commercial solver. Furthermore, an alternative formulation of the Dantzig-Wolfe decomposition is proposed in order to obtain more meaningful dual variables. Finally, an alternative branching strategy is proposed which results in the processing of a much lower number of nodes compared to the obvious branching strategy as used in e.g. [33].

The remainder of this chapter is structured as follows. First, a mixed integer non-linear programming formulation of the problem as introduced in section 2.3 is presented in section 6.1 along with its novel and tighter linear upper bound. The Dantzig-Wolfe reformulation of the linear problem and the column generation approach to solve the relaxed problem is

presented in section 6.2. Next, the branch & price algorithm to solve the integer problem is described in section 6.3. Finally, the computational experiments in section 6.4 show the superiority and applicability of the proposed method.

6.1 Mixed integer non-linear programming formulation and linear upper bound

In this section, the compact programming formulation is presented, which yields an optimal search strategy for the team of platforms. The formulation is a mixed integer non-linear program (MINLP), presented first in subsection 6.1.1. A novel and tighter linear upper bound is then proposed in subsection 6.1.2 in order to obtain a mixed integer linear program (MILP), which is finally subject to the Dantzig-Wolfe decomposition in the next section.

6.1.1 Mixed integer non-linear programming formulation

The problem of finding an optimal search strategy for a team of heterogeneous platforms with shared resources, as stated in the problem formulation, can be formulated as the mixed integer non-linear programming formulation presented in this subsection. For the purpose of clarity throughout this chapter, let

$$w_{\psi,\boldsymbol{z}} = \sum_{u\in\mathcal{U}}\sum_{k\in\mathcal{K}}\sum_{v\in\mathcal{V}}\sum_{c\in\mathcal{C}} W_{u,k,v,c}x_{\psi,k,c}z_{u,k,v}, \tag{6.1}$$

such that consequently the formula for the glimpse probability as introduced in Equation (2.8)

$$pg_\psi = 1 - e^{-\sum_{u\in\mathcal{U}}\sum_{k\in\mathcal{K}}\sum_{v\in\mathcal{V}}\sum_{c\in\mathcal{C}} W_{u,k,v,c}x_{\psi,k,c}z_{u,k,v}},$$

can be formulated more concisely as

$$pg_\psi = 1 - e^{-w_{\psi,\boldsymbol{z}}}. \tag{6.2}$$

The problem of multi-platform search for targets with a generic motion model can be

formulated similarly to that in [13, 45] as the following mixed integer non-linear program:

$$\text{Maximize} \sum_{\psi \in \Psi} pt_\psi pg_\psi \tag{6.3}$$

Subject to

$$pg_\psi = 1 - e^{-w_{\psi,\boldsymbol{z}}} \quad \forall \psi \in \Psi \tag{6.4}$$

$$\boldsymbol{z_u} \in \boldsymbol{Z_u} \quad \forall u \in \mathcal{U} \tag{6.5}$$

$$\sum_{u \in \mathcal{U}} \sum_{k \in \mathcal{K}} \sum_{v \in \mathcal{V}} pr_{u,k,v} z_{u,k,v} \leq T \tag{6.6}$$

$$z_{u,k,v} \in \{0,1\} \quad \forall u \in \mathcal{U}, \forall k \in \mathcal{K}, \forall v \in \mathcal{V} \tag{6.7}$$

$$pg_\psi \geq 0 \quad \forall \psi \in \Psi \tag{6.8}$$

Here, the objective (6.3) is to maximize the expected probability of detection as defined in Equation (2.9). The calculation of the glimpse probability $pg_\psi \geq 0$ according to formula (6.2) for each target track $\psi \in \Psi$ is ensured by the set of constraints in (6.4). The set of constraints in (6.5) ensure that the binary vector $\boldsymbol{z_u}$ representing a search trajectory of platform u is indeed physically feasible. Finally, the set of constraints in (6.6) ensures that the cumulative resource consumption does not exceed the given limit T.

6.1.2 Linear upper bound

In general, mixed integer non-linear problems are very difficult to solve. In real-life search missions it is necessary to have an optimal search strategy ready as soon as the first platform is ready for takeoff. Therefore, alternative linear objective functions have been suggested in [13] for use, instead of the non-linear objective function. In this section, an improvement on this in terms of the optimality gap is proposed.

Let

$$w_\psi^U = \max_{\boldsymbol{z}} w_{\psi,\boldsymbol{z}} \quad \text{and} \quad w_\psi^L = \min_{\boldsymbol{z}} w_{\psi,\boldsymbol{z}} \tag{6.9}$$

denote the upper and lower bounds of $w_{\psi,\boldsymbol{z}}$, respectively.

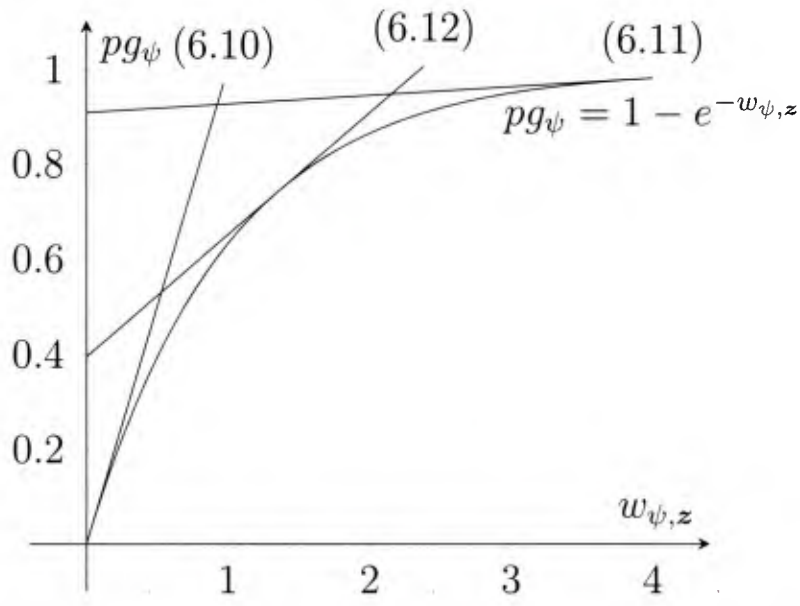

Figure 6-3: Linear overestimators of the original non-linear glimpse probability function.

Using the fact that $1 - e^{-z} \leq z$, constraint (6.4) is replaced by the following linear overestimator to obtain the linear objective function that was proposed in [18] and used in [13] for real-world field experiments:

$$pg_\psi \leq \sum_{u \in \mathcal{U}} \sum_{k \in \mathcal{K}} \sum_{v \in \mathcal{V}} \sum_{c \in \mathcal{C}} W_{u,k,v,c} x_{\psi,k,c} z_{u,k,v}, \tag{6.10}$$

Next, two additional linear overestimators are proposed to tighten the upper bound of the glimpse probability pg_ψ. Both overestimators are tangent lines of Equation (6.2). The first is the tangent line at the point $(w_\psi^U, 1 - e^{-w_\psi^U})$ and has the following equation:

$$pg_\psi \leq e^{-w_\psi^U} \left(w_{\psi,z} - w_\psi^U\right) + \left(1 - e^{-w_\psi^U}\right), \tag{6.11}$$

And the last linear overestimator is the tangent line of Equation (6.2) at the point $(\frac{e}{2}, 1 - e^{-\frac{e}{2}})$, hence

$$pg_\psi \leq e^{-\frac{e}{2}} \left(w_{\psi,z} - \frac{e}{2}\right) + \left(1 - e^{-\frac{e}{2}}\right). \tag{6.12}$$

Notice that it is possible to tighten the upper bound of the glimpse probability even further by adding tangent lines on additional points along the exponential objective function.

However, because the three linear constraints are added for *each* possible target track ψ, the number of constraints quickly grows to a point that computation may become intractable. By means of computational experiments it was found that the combination of the constraints in (6.10), (6.11), and (6.12) yields a proper trade-off between tightness of the bound and computational burden. These overestimators are shown graphically in Figure 6-3. By replacing the non-linear set of constraints (6.4) in the original problem (6.3)-(6.8) with the three proposed overestimators (6.10), (6.11), and (6.12), the following mixed integer linear problem is obtained:

$$\text{Maximize} \sum_{\psi \in \Psi} pt_{\psi} pg_{\psi} \tag{6.13}$$

Subject to

$$pg_{\psi} \leq w_{\psi,\boldsymbol{z}} \quad \forall \psi \in \Psi \tag{6.14}$$

$$pg_{\psi} \leq e^{-w_{\psi}^{U}} \left(w_{\psi,\boldsymbol{z}} - w_{\psi}^{U}\right) + \left(1 - e^{-w_{\psi}^{U}}\right) \quad \forall \psi \in \Psi \tag{6.15}$$

$$pg_{\psi} \leq e^{-\frac{e}{2}} \left(w_{\psi,\boldsymbol{z}} - \frac{e}{2}\right) + \left(1 - e^{-\frac{e}{2}}\right) \quad \forall \psi \in \Psi \tag{6.16}$$

$$\boldsymbol{z_u} \in \boldsymbol{Z_u} \quad \forall u \in \mathcal{U} \tag{6.17}$$

$$\sum_{u \in \mathcal{U}} \sum_{k \in \mathcal{K}} \sum_{v \in \mathcal{V}} pr_{u,k,v} z_{u,k,v} \leq T \tag{6.18}$$

$$z_{u,k,v} \in \{0,1\} \quad \forall u \in \mathcal{U}, \forall k \in \mathcal{K}, \forall v \in \mathcal{V} \tag{6.19}$$

$$pg_{\psi} \geq 0 \quad \forall \psi \in \Psi \tag{6.20}$$

Here, the objective (6.13) remains to find a binary vector that represents a physically feasible trajectory (ensured by the set of constraints in (6.17)) that maximizes the expected probability of detection. An upper bound on the glimpse probability pg_{ψ} for each possible target track $\psi \geq 0$ is ensured by the sets of linear overestimators in (6.14)-(6.16). Figure 6-4 shows a search scenario in which two platforms are modeled to search for one east moving target. The optimal strategy is obtained by solving the mixed integer linear program in (6.13)-(6.20). This holds for Figure 6-5 as well, which shows a similar scenario but with additional shared resource constraints in the form of time exposed to risk.

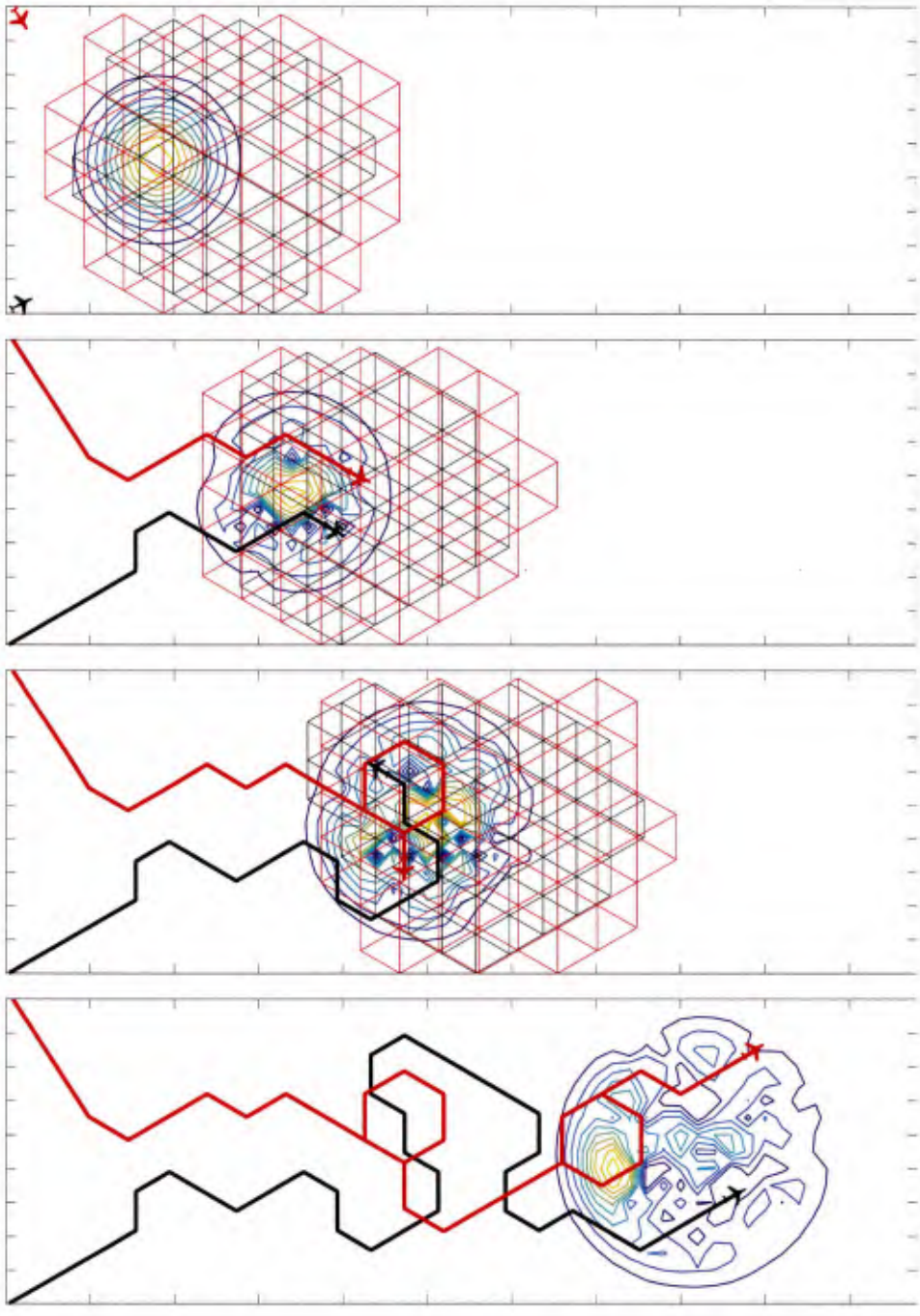

Figure 6-4: A multiple platform search scenario. Each platform is modeled to move over its own hexagonal network. The black platform has a smaller turn radius, therefore the corresponding hexagonal network has smaller scale compared to the red hexagonal network. Nevertheless, the solution method optimizes the search strategy for both platforms in a centralized manner. Platform collision is avoided by flying at different heights with sufficient vertical distance.

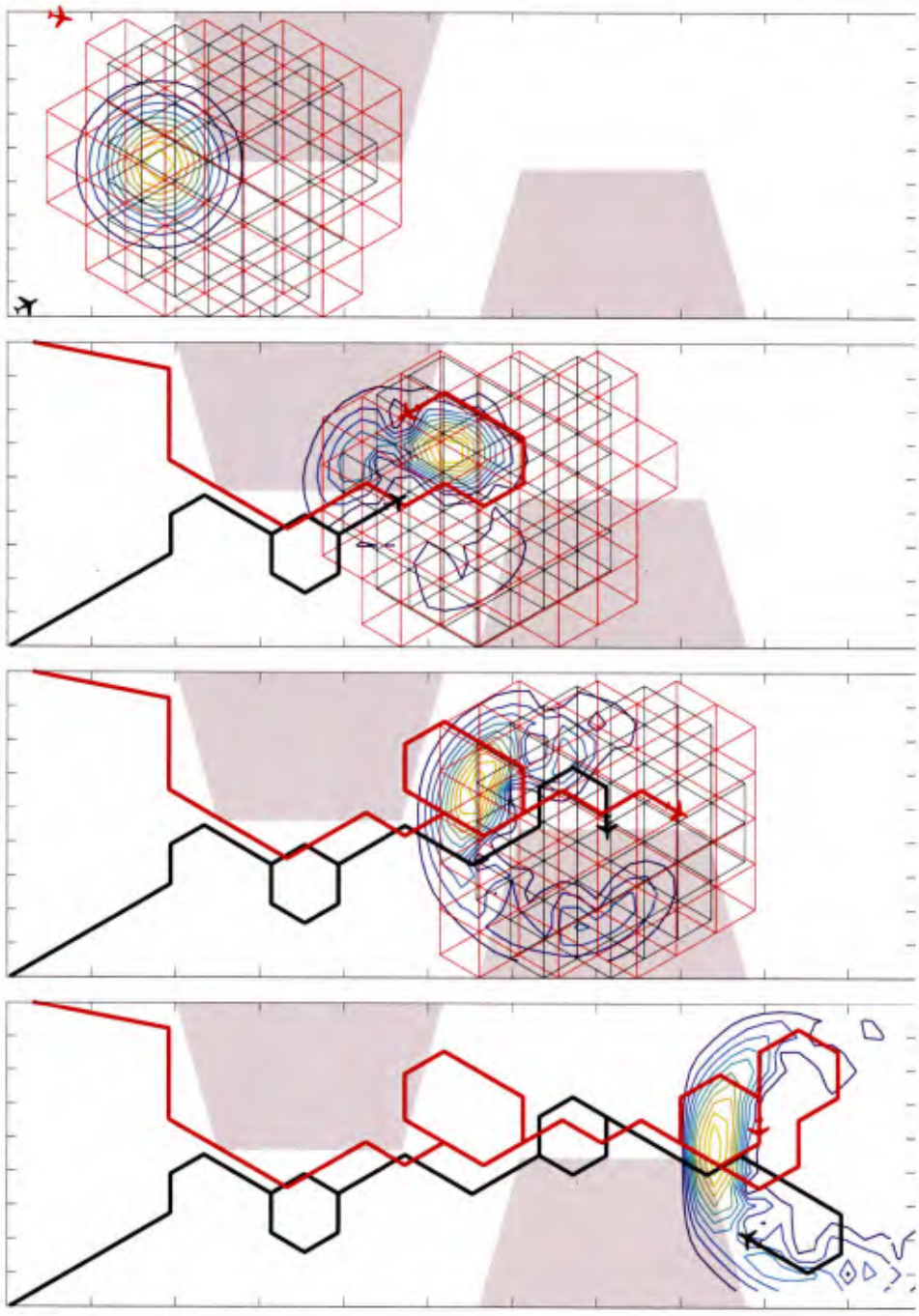

Figure 6-5: A multiple platform search scenario with shared resource constraints. The gray polygons represent areas in which the platforms are exposed to risk. The combined time exposed to risk of both platforms is constrained to at most 16.66% of the total mission time.

Instances of this linearization of the original non-linear problem (6.3)-(6.8) can be solved with a commercial branch & bound solver. However, when the number of time steps or

the number of nodes increases, the cardinality of the problem becomes too large and the problem becomes intractable. In order to decrease the runtime on solving such instances, the linearization in (6.13)-(6.20) can be decomposed and subsequently solved in iterations. The proposed Dantzig-Wolfe decomposition is presented in the next subsection.

6.2 Dantzig-Wolfe decomposition

Column generation is proven to be a method for solving huge linear programs more efficiently [88]. The first step in applying column generation to the linear program of interest is a reformulation of the program in a manner proposed by Dantzig and Wolfe [89]. For the problem at hand, the Dantzig-Wolfe reformulation results in deciding between the possible physically feasible trajectories, instead of selecting K single waypoints that yield an optimal physically feasible trajectory, and is presented in subsection 6.2.1. Such a reformulation, however, results in an extremely large amount of decision variables; one for each physically feasible trajectory. This can be avoided by iteratively adding one or more physically feasible trajectories (or *columns* in general terms of column generation) that yield maximum positive reduced cost. The procedure of adding such columns is described in subsection 6.2.2.

6.2.1 Dantzig-Wolfe reformulation

The Dantzig-Wolfe reformulation is obtained by using the fact that an extreme point $z_{u,\omega}$ of the polytope defined by that of the convex hull of $\boldsymbol{Z_u}$ in constraint (6.17) represents a physically feasible trajectory $\omega \in \Omega$ on the network G. Therefore, each physically feasible trajectory can be formulated as a convex combination of physically feasible trajectories as follows:

$$\sum_{\omega \in \Omega} z_{u,k,v,\omega} \lambda_{u,\omega} = z_{u,k,v} \quad \forall u \in \mathcal{U}, \forall k \in \mathcal{K}, \forall v \in \mathcal{V} \tag{6.21}$$

$$\sum_{\omega \in \Omega} \lambda_{u,\omega} = 1 \quad \forall u \in \mathcal{U} \tag{6.22}$$

$$\lambda_{u,\omega} \geq 0 \quad \forall u \in \mathcal{U}, \forall \omega \in \Omega \tag{6.23}$$

Here, the decision variable $\lambda_{u,\omega}$ represents the selection of physically feasible trajectory $\omega \in \Omega$ for platform $u \in \mathcal{U}$. To obtain the master problem MP, Equations (6.21)-(6.23) are substituted for $\boldsymbol{z_u}$ in (6.14),(6.15),(6.16), and (6.18). Furthermore, the constraint in (6.17) is dropped because the structural information of the trajectory is contained in ω using the parameter $z_{u,k,v,\omega}$. This parameter has a value of 1 assigned if (k, v) is a waypoint on the trajectory ω for platform u, and is 0 otherwise. Finally, auxiliary decision variables $\boldsymbol{z'} = \{z'_{u,k,v} \in \mathbb{R} : u \in \mathcal{U}, k \in \mathcal{K}, v \in \mathcal{V}\}$ are introduced. The master problem MP of the Dantzig-Wolfe reformulation can be formulated as follows:

$$\text{Maximize} \sum_{\psi \in \Psi} pt_\psi pg_\psi \tag{6.24}$$

Subject to

$$pg_\psi \leq w_{\psi,\boldsymbol{z'}} \quad \forall \psi \in \Psi \tag{6.25}$$

$$pg_\psi \leq e^{-w^U_\psi}\left(w_{\psi,\boldsymbol{z'}} - w^U_\psi\right) + \left(1 - e^{-w^U_\psi}\right) \quad \forall \psi \in \Psi \tag{6.26}$$

$$pg_\psi \leq e^{-\frac{e}{2}}\left(w_{\psi,\boldsymbol{z'}} - \frac{e}{2}\right) + \left(1 - e^{-\frac{e}{2}}\right) \quad \forall \psi \in \Psi \tag{6.27}$$

$$\sum_{u \in \mathcal{U}} \sum_{k \in \mathcal{K}} \sum_{v \in \mathcal{V}} pr_{u,k,v} \sum_{\omega \in \Omega} z_{u,k,v,\omega} \lambda_{u,\omega} \leq T \tag{6.28}$$

$$\sum_{\omega \in \Omega} z_{u,k,v,\omega} \lambda_{u,\omega} \geq z'_{u,k,v} \quad \forall u \in \mathcal{U}, \forall k \in \mathcal{K}, \forall v \in \mathcal{V} \tag{6.29}$$

$$\sum_{\omega \in \Omega} \lambda_{u,\omega} = 1 \quad \forall u \in \mathcal{U} \tag{6.30}$$

$$\lambda_{u,\omega} \geq 0 \quad \forall u \in \mathcal{U}, \forall \omega \in \Omega \tag{6.31}$$

$$\sum_{\omega \in \Omega} z_{u,k,v,\omega} \lambda_{u,\omega} = z_{u,k,v} \quad \forall u \in \mathcal{U}, \forall k \in \mathcal{K}, \forall v \in \mathcal{V} \tag{6.32}$$

$$z_{u,k,v} \in \{0, 1\} \quad \forall u \in \mathcal{U}, \forall k \in \mathcal{K}, \forall v \in \mathcal{V} \tag{6.33}$$

$$z'_{u,k,v} \in \mathbb{R} \quad \forall u \in \mathcal{U}, \forall k \in \mathcal{K}, \forall v \in \mathcal{V} \tag{6.34}$$

$$pg_\psi \geq 0 \quad \forall \psi \in \Psi \tag{6.35}$$

This master problem is only partly formulated in the standard way. Usually in literature, the convex combination of trajectories is achieved by replacing the original decision

variables $z_{u,k,v}$ by the term $\sum_{\omega\in\Omega} z_{u,k,v,\omega}\lambda_{u,\omega}$ at each occurrence. However, when following this procedure to obtain a Dantzig-Wolfe reformulation of (6.13)-(6.20), three dual variables corresponding to the constraints (6.25), (6.26), and (6.27) would be obtained for each possible target track. However, this information is rather useless for finding search trajectories to enter the basis. To find an optimal search trajectory to enter the basis, information on the possible gain of each waypoint for each platform is needed, i.e. information for each tuple (u, k, v). To this aim, the vector of auxiliary decision variables $\boldsymbol{z'}$ is introduced. Each auxiliary decision variable $z'_{u,k,v}$ replaces its corresponding original decision variable $z_{u,k,v}$ and is restricted by constraint (6.29) to take at most the value of $\sum_{\omega\in\Omega} z_{u,k,v,\omega}\lambda_{u,\omega}$. Constraint (6.29) now produces useful dual information for each waypoint, which is needed to price out trajectories with maximum reduced cost. Furthermore, the constraints (6.30) and (6.31) ensure the non-negative convex combination of trajectories.

By relaxing the binary variables in (6.33), the necessity of linking the $\boldsymbol{x}$ and $\boldsymbol{\lambda}$ variables becomes obsolete. Therefore, the set of constraints (6.32) can be dropped as well. When the number of nodes in $\mathcal{V}$ becomes larger, or when the duration of the search mission increases, the number of possible physically feasible trajectories and hence the number of columns becomes excessively large. Therefore the algorithm is initiated with only a subset of Ω and columns are added iteratively. This procedure is well known as *column generation* [88]. The linear relaxation of MP with a subset of Ω is the following restricted master problem RMP:

$$\text{Maximize} \sum_{\psi\in\Psi} pt_\psi pg_\psi \tag{6.36}$$

Subject to

$$pg_\psi \leq w_{\psi,\boldsymbol{z'}} \quad \forall\psi\in\Psi \tag{6.37}$$

$$pg_\psi \leq e^{-w^U_\psi}\left(w_{\psi,\boldsymbol{z'}} - w^U_\psi\right) + \left(1 - e^{-w^U_\psi}\right) \quad \forall\psi\in\Psi \tag{6.38}$$

$$pg_\psi \leq e^{-\frac{e}{2}}\left(w_{\psi,\boldsymbol{z'}} - \frac{e}{2}\right) + \left(1 - e^{-\frac{e}{2}}\right) \quad \forall\psi\in\Psi \tag{6.39}$$

$$\sum_{u\in\mathcal{U}}\sum_{k\in\mathcal{K}}\sum_{v\in\mathcal{V}} pr_{u,k,v}\sum_{\omega\in\Omega} z_{u,k,v,\omega}\lambda_{u,\omega} \leq T \tag{6.40}$$

$$\sum_{\omega\in\Omega} z_{u,k,v,\omega}\lambda_{u,\omega} \geq z'_{u,k,v} \quad \forall u \in \mathcal{U}, \forall k \in \mathcal{K}, \forall v \in \mathcal{V} \tag{6.41}$$

$$\sum_{\omega\in\Omega} \lambda_{u,\omega} = 1 \quad \forall u \in \mathcal{U} \tag{6.42}$$

$$\lambda_{u,\omega} \geq 0 \quad \forall u \in \mathcal{U}, \forall \omega \in \Omega \tag{6.43}$$

$$z'_{u,k,v} \in \mathbb{R} \quad \forall u \in \mathcal{U}, \forall k \in \mathcal{K}, \forall v \in \mathcal{V} \tag{6.44}$$

$$pg_{\psi} \geq 0 \quad \forall \psi \in \Psi \tag{6.45}$$

The remaining problem consists of a restricted selection of main decision variables $\boldsymbol{\lambda}$ for the trajectories of the platforms and, furthermore, of the auxiliary variables $\boldsymbol{z'}$ to obtain a meaningful dual variable for each waypoint for each platform. The process of finding a column to enter the basis in RMP is called *reduced cost pricing* and is described in the following subsection.

6.2.2 Duality and reduced cost pricing

To avoid an extremely large number of columns in the restricted master problem RMP, columns are priced out iteratively. The column with maximum positive reduced cost is the column being priced out and will be added to the RMP. Finding a column with maximum positive reduced cost is known as a pricing problem , which is described after the derivation of the reduced cost of a column.

First, associate dual variables $\boldsymbol{\rho^{(1)}} = \{\rho_{\psi}^{(1)} \leq 0 : \psi \in \Psi\}$ with the constraints in (6.37), dual variables $\boldsymbol{\rho^{(2)}} = \{\rho_{\psi}^{(2)} \leq 0 : \psi \in \Psi\}$ with the constraints in (6.38) and dual variables $\boldsymbol{\rho^{(3)}} = \{\rho_{\psi}^{(3)} \leq 0 : \psi \in \Psi\}$ with the constraints in (6.39). The dual variables $\pi_1 \geq 0$, $\boldsymbol{\sigma} = \{\sigma_{u,k,v} \geq 0 : u \in \mathcal{U}, k \in \mathcal{K}, v \in \mathcal{V}\}$ and $\boldsymbol{\pi_0} = \{\pi_{0,u} \in \mathbb{R} : u \in \mathcal{U}\}$ are associated with the constraints in (6.40), (6.41), and (6.42), respectively. The Langrangian L_{RMP} of RMP then amounts to:

$$
\begin{aligned}
L_{RMP} = & \sum_{\psi \in \Psi} pt_{\psi} pg_{\psi} \\
& - \sum_{\psi \in \Psi} \rho_{\psi}^{(1)} \left(\sum_{u \in \mathcal{U}} \sum_{k \in \mathcal{K}} \sum_{v \in \mathcal{V}} \sum_{c \in \mathcal{C}} W_{u,k,v,c} x_{\psi,k,c} z'_{u,k,v} - pg_{\psi} \right) \\
& - \sum_{\psi \in \Psi} \rho_{\psi}^{(2)} \left(e^{-w_{\psi}^{U}} \left(\sum_{u \in \mathcal{U}} \sum_{k \in \mathcal{K}} \sum_{v \in \mathcal{V}} \sum_{c \in \mathcal{C}} W_{u,k,v,c} x_{\psi,k,c} z'_{u,k,v} - w_{\psi}^{U} \right) + \left(1 - e^{-w_{\psi}^{U}} \right) - pg_{\psi} \right) \\
& - \sum_{\psi \in \Psi} \rho_{\psi}^{(3)} \left(e^{-\frac{e}{2}} \left(\sum_{u \in \mathcal{U}} \sum_{k \in \mathcal{K}} \sum_{v \in \mathcal{V}} \sum_{c \in \mathcal{C}} W_{u,k,v,c} x_{\psi,k,c} z'_{u,k,v} - \frac{e}{2} \right) + \left(1 - e^{-\frac{e}{2}} \right) - pg_{\psi} \right) \\
& - \pi_1 \left(\sum_{u \in \mathcal{U}} \sum_{k \in \mathcal{K}} \sum_{v \in \mathcal{V}} pr_{u,k,v} \sum_{\omega \in \Omega} z_{u,k,v,\omega} \lambda_{u,\omega} - T \right) \\
& - \sum_{u \in \mathcal{U}} \sum_{k \in \mathcal{K}} \sum_{v \in \mathcal{V}} \sigma_{u,k,v} \left(z'_{u,k,v} - \sum_{\omega \in \Omega} z_{u,k,v,\omega} \lambda_{u,\omega} \right) \\
& - \sum_{u \in \mathcal{U}} \pi_{0,u} \left(\sum_{\omega \in \Omega} \lambda_{u,\omega} - 1 \right) \qquad (6.46)
\end{aligned}
$$

A reformulation leads to:

$$
\begin{aligned}
L_{RMP} = & \sum_{\psi \in \Psi} pg_{\psi} \left(pt_{\psi} + \rho_{\psi}^{(1)} + \rho_{\psi}^{(2)} + \rho_{\psi}^{(3)} \right) \\
& - \sum_{u \in \mathcal{U}} \sum_{k \in \mathcal{K}} \sum_{v \in \mathcal{V}} z'_{u,k,v} \left(\sum_{\psi \in \Psi} \sum_{c \in \mathcal{C}} W_{u,k,v,c} x_{\psi,k,c} \left(\rho_{\psi}^{(1)} + \rho_{\psi}^{(2)} e^{-w_{\psi}^{U}} + \rho_{\psi}^{(3)} e_{\psi}^{-\frac{e}{2}} \right) - \sigma_{u,k,v} \right) \\
& + \sum_{u \in \mathcal{U}} \sum_{\omega \in \Omega} \lambda_{u,\omega} \left(\sum_{k \in \mathcal{K}} \sum_{v \in \mathcal{V}} z_{u,k,v,\omega} \sigma_{u,k,v} - \sum_{k \in \mathcal{K}} \sum_{v \in \mathcal{V}} pr_{u,k,v} z_{u,k,v,\omega} \pi_1 - \pi_{0,u} \right) \\
& - \sum_{\psi \in \Psi} \rho_{\psi}^{(2)} \left(-w_{\psi}^{U} e^{-w_{\psi}^{U}} + \left(1 - e^{-w_{\psi}^{U}} \right) \right) \\
& - \sum_{\psi \in \Psi} \rho_{\psi}^{(3)} \left(-\frac{e}{2} e_{\psi}^{-\frac{e}{2}} + \left(1 - e_{\psi}^{-\frac{e}{2}} \right) \right) \\
& + \pi_1 T + \sum_{u \in \mathcal{U}} \pi_{0,u} \qquad (6.47)
\end{aligned}
$$

Finally, the dual D_{RMP} of RMP amounts to:

$$\begin{aligned}\text{Min} \sum_{u\in\mathcal{U}}\sum_{\omega\in\Omega}\lambda_{u,\omega}\left(\sum_{k\in\mathcal{K}}\sum_{v\in\mathcal{V}} z_{u,k,v,\omega}\sigma_{u,k,v} - \sum_{k\in\mathcal{K}}\sum_{v\in\mathcal{V}} pr_{u,k,v}z_{u,k,v,\omega}\pi_1 - \pi_{0,u}\right) \\ - \sum_{\psi\in\Psi}\rho_\psi^{(2)}\left(-w_\psi^U e^{-w_\psi^U} + \left(1 - e^{-w_\psi^U}\right)\right) \\ - \sum_{\psi\in\Psi}\rho_\psi^{(3)}\left(-\frac{e}{2}e_\psi^{-\frac{e}{2}} + \left(1 - e_\psi^{-\frac{e}{2}}\right)\right) \\ + \pi_1 T + \sum_{u\in\mathcal{U}}\pi_{\mathfrak{I},u} \quad (6.48)\end{aligned}$$

Subject to

$$\rho_\psi^{(1)} + \rho_\psi^{(2)} + \rho_\psi^{(3)} = -pt_\psi \quad \forall\psi\in\Psi \quad (6.49)$$

$$\sum_{\psi\in\Psi}\sum_{c\in\mathcal{C}} W_{u,k,v,c}x_{\psi,k,c}\left(\rho_\psi^{(1)} + \rho_\psi^{(2)}e^{-w_\psi^U} + \rho_\psi^{(3)}e_\psi^{-\frac{e}{2}}\right)$$

$$= \sigma_{u,k,v} \quad \forall u\in\mathcal{U},\forall k\in\mathcal{K},\forall v\in\mathcal{V} \quad (6.50)$$

$$\rho_\psi^{(1)}, \rho_\psi^{(2)}, \rho_\psi^{(3)} \leq 0 \quad \forall\psi\in\Psi \quad (6.51)$$

$$\pi_{0,u} \in \mathbb{R} \quad \forall u\in\mathcal{U},\forall k\in\mathcal{K},\forall v\in\mathcal{V} \quad (6.52)$$

$$\sigma_{u,k,v}, \pi_1 \geq 0 \quad (6.53)$$

To select a column to enter the basis, a search trajectory with positive reduced cost must be found. By means of the dual objective function (6.48), the reduced cost $c_{u,\omega}$ of trajectory ω for platform u is defined by

$$c_{u,\omega} = \sum_{\omega\in\Omega}\lambda_{u,\omega}\left(\sum_{k\in\mathcal{K}}\sum_{v\in\mathcal{V}} z_{u,k,v,\omega}\left(\sigma_{u,k,v} - pr_{u,k,v}\pi_1\right) - \pi_{0,u}\right). \quad (6.54)$$

For the best results, at each iteration of the column generation procedure, the aim is to find a trajectory with the highest positive reduced cost $c^*_{u,\omega}$, i.e.:

$$c^*_{u,\omega} = \max_{\omega\in\Omega}\sum_{k\in\mathcal{K}}\sum_{v\in\mathcal{V}} z_{u,k,v,\omega}\left(\sigma_{u,k,v} - pr_{u,k,v}\pi_1\right) - \pi_{0,u}. \quad (6.55)$$

A trajectory ω with maximum positive reduced cost c^*_ω is priced out and added as a column to the restricted master problem RPM. When no column with positive reduced cost is found, i.e. when

$$\sum_{k\in\mathcal{K}}\sum_{v\in\mathcal{V}} z_{u,k,v,\omega}\left(\sigma_{u,k,v} - pr_{u,k,v}\pi_1\right) - \pi_{0,u} \leq 0 \tag{6.56}$$

for each possible trajectory, then the current solution to the RPM is optimal and the column generation procedure can be aborted. In addition, the procedure also terminates when equal trajectories are added consecutively.

6.2.3 Efficiently pricing out a trajectory with maximum reduced cost

The efficient procedure of finding a physically feasible trajectory with maximum reduced cost $c^*_{u,\omega}$ is described next. This problem for platform u can be formulated as follows:

$$\text{Maximize} \sum_{k\in\mathcal{K}}\sum_{v\in\mathcal{V}} z_{u,k,v}\left(\sigma_{u,k,v} - pr_{u,k,v}\pi_1\right) \tag{6.57}$$

Subject to

$$\boldsymbol{z_u} \in \boldsymbol{Z_u} \tag{6.58}$$

$$z_{u,k,v} \in \{0,1\} \quad \forall k \in \mathcal{K}, \forall v \in \mathcal{V} \tag{6.59}$$

The proposed procedure for solving (6.57)-(6.59) consists of finding a longest path on a directed acyclic graph (DAG), however with a modified reward structure. In this case, a node on the DAG for platform u has the reward $\sigma_{u,k,v} - pr_{u,k,v}\pi_1$ corresponding to a waypoint $z_{u,k,v}$ as in (6.55). This procedure is executed for each platform $u \in \mathcal{U}$ and consists of three main steps:

1. Construct the DAG $(\mathcal{N}, \mathcal{E})$ for platform u.
2. Find a longest path on the DAG (this path corresponds to a trajectory ω with maximum positive reduced cost c^*_ω).
3. Evaluate $c^*_{u,\omega^*} > 0$.

The DAG under construction in this work is a generic type which is applicable to both trajectories with kinematical constraints as in [11] and trajectories without kinematical constraints as in [13]. Furthermore, this DAG is applicable to both branching strategies proposed in section 6.3.2. To avoid confusion with nodes in the set $\mathcal{V}$ which refers to the set of nodes on the network G, the word *dag-node* is used for a node on the DAG. A dag-node n is a special structure with the following elements: a current node v, a previous node v_{prev}, a list of predecessors $\mathcal{N}_{prev}$ each of the type dag-node, and a value. The DAG $(\mathcal{N}, \mathcal{E})$ is constructed as follows. First, for each reachable node v at time $k = 1$ a dag-node n is created with the previous node empty and the current node v, i.e. $n.v_{prev} = null$ and $n.v = v$. The value of the dag-node is $\sigma_{u,k,v} - pr_{u,k,v}\pi_1$. The list of predecessors remains empty for now, i.e. $n.\mathcal{N}_{prev} = \{\}$. Each of the created nodes is added to the DAG and forms the first layer $\mathcal{N}_1$ of the DAG, which corresponds to time step $k = 1$. Additional layers are created for each time step $k \in \mathcal{K}$ with $1 < k \leq K$ and filled with dag-nodes as follows. For each node v' in $\mathcal{V}'_k$, whether or not a corresponding dag-node can be created to add to the current layer is evaluated. Here, $\mathcal{V}'_k$ typically equals $\mathcal{V}$ but is just a subset of $\mathcal{V}$ once the branching has started (see 6.3.2). First, for each dag-node n on the previous layer $\mathcal{N}_{k-1}$, a check is done to identify whether node v' is adjacent to node $n.v$ and, additionally, not adjacent to node $n.v_{prev}$. The last exclusion is of course not necessary for rotary-wing platforms (see Section 4.1.3). The dag-node n' is created with $n'.v_{prev} = n.v$ and $n'.v = v'$ and added to layer $\mathcal{N}_k$ if and only if this layer does not yet contain a dag-node n'' with $n''.v_{prev} = n.v$ and $n''.v = v'$. Moreover, the dag-node n is added to the list of predecessors of dag-node n'. Finally, for each dag-node, a directed edge is created from each of its predecessors to itself. It is straightforward to see that the resulting graph is indeed a directed acyclic graph such that for each dag-node n_k on a path $(n_1, n_2, ..., n_K)$ on the DAG it holds that $n_k \in \mathcal{N}_k$. A graphical representation of the DAG construction is presented in Figure 6-6.

Once the DAG is created, a straightforward longest-path algorithm as described in [29] is applied to find a longest path on the DAG. This algorithm has very low computational costs as it runs in $\mathcal{O}(|\mathcal{N}| + |\mathcal{E}|)$ time and hence in linear time. The path $(n_1, n_2, ..., n_K)$ corresponds to the physically feasible trajectory ω^* with maximum reduced cost c^*_ω, which is represented by the binary vector z_{u,ω^*} with $z_{u,k,v,\omega^*} = 1$ if $n_k.v = v$ and zero otherwise.

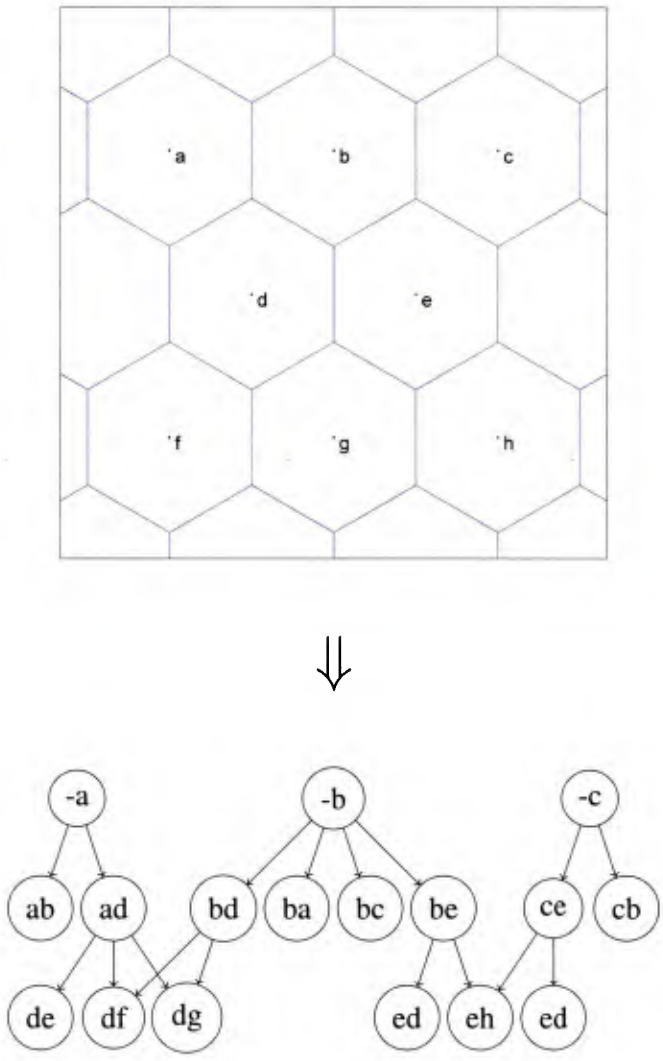

Figure 6-6: Construction example of a directed acyclic graph for pricing out a physically feasible trajectory for a fixed-wing platform with maximum reduced cost.

Let $\mathcal{V} = \{a,b,c,d,e,f,g,h\}$ and let nodes a, b, and c be the nodes which are reachable at time $k = 1$. Then the directed acyclic graph for pricing out a trajectory with maximum reduced cost is constructed as shown. Each path on the DAG represents a physically feasible trajectory for a fixed-wing platform. Notice that there exists no edge between bd and de, because e is adjacent to b. There exists no edge between be and ed either, because d is adjacent to b. Moreover, nodes df, dg, de, and eh all have multiple predecessors.

Finally, it is evaluated whether $c^*_{u,\omega^*} > 0$ by means of Equation (6.56). If the reduced cost of trajectory ω^* is indeed positive, trajectory ω^* is added to the reduced master problem. Otherwise, no physically feasible trajectory with positive reduced cost exists and the column generation procedure terminates. This procedure is presented in pseudo code in Algorithm 3.

Algorithm 3 Pricing out a trajectory with max reduced cost for platform u

Initiate DAG $\mathcal{N} \leftarrow \{\}$
Initiate layer for $k = 1$ $\mathcal{N}_1 \leftarrow \{\}$
for each reachable node $v \in \mathcal{R}_1$ **do**
  Create dag-node n with $n.v_{prev} = null$, $n.v = v$, $n.value = \sigma_{u,k,v} - pr_{u,k\,v}\pi_1$, and $\mathcal{N}_{prev} = \{\}$
  $\mathcal{N}_1 \leftarrow \mathcal{N}_1 \cup n$
end for
for $1 < k \leq K$ **do**
  Initiate layer $\mathcal{N}_k \leftarrow \{\}$
  for each node $v' \in \mathcal{V}'_k$ **do**
    for each dag-node $n \in \mathcal{N}_{k-1}$ **do**
      if $a_{v',n.v}$ **and** $\tilde{a}_{v',n.v_{prev}}$ **then**
        if There exists no dag-node $n' \in \mathcal{N}$ with $n'.v_{prev} = n.v$ and $n'.v = v'$ **then**
          Create dag-node $n'' \in \mathcal{N}$ with $n''.v_{prev} = v'$ and $n''.v = n.v$
          $\mathcal{N}_k \leftarrow \mathcal{N}_k \cup n''$
        end if
        $n''.\mathcal{N}_{pred} \leftarrow n''.\mathcal{N}_{pred} \cup n$
      end if
    end for
  end for
  $\mathcal{N} \leftarrow \mathcal{N} \cup \mathcal{N}_k$
end for
Trajectory with maximum positive reduced cost $\omega^* \leftarrow$ longest path algorithm on the DAG and conversion to its corresponding trajectory
if $c^*_{u,\omega^*} > 0$ **then**
  Add trajectory ω^* to the RMP
else
  STOP
end if

This algorithm for pricing out a trajectory with max reduced cost is part of the column generation procedure for optimizing the restricted master problem. In its turn, the column generation procedure is part of the branch & price algorithm, which is presented in the next section.

6.3 Branch & price algorithm

The result of the column generation procedure may contain fractional waypoints. A platform is, of course, indivisible and therefore a binary solution is required. To obtain a binary solution, the column generation procedure from section 6.2 is incorporated in a branch & bound algorithm. The result of combining column generation with branch & bound is known as branch & price and is presented in subsection 6.3.1. Two optional branching strategies are presented in subsection 6.3.2.

6.3.1 Solving the integer problem

Let $\mathcal{Q}$ be the queue of nodes on the branch & price tree. To avoid confusion with nodes in $\mathcal{V}$, a node in $\mathcal{Q}$ is referred to as a *bp-node* throughout this section. The queue of bp-nodes in $\mathcal{Q}$ is initiated with a single bp-node n_0 referred to as the *root* bp-node. Other bp-nodes will be added to the tree later on. First, the bp-node with highest priority is pulled from the queue $\mathcal{Q}$ to be processed. At his point, root bp-node n_0 is the only bp-node in the queue and, hence, the root bp-node is processed first. Processing a bp-node starts with checking whether a feasible solution exists in the RMP. Initially, there are no columns yet added, i.e. $\Omega = \emptyset$, and constraint (6.30) can not be satisfied. Therefore, a feasible solution must be generated first by generating one feasible trajectory for each platform. In the root bp-node, each trajectory in $\boldsymbol{Z_u}$ is feasible when its required resource does not exceed the limit T. However, the combination of trajectories of the platforms might exceed the limit T. Therefore, feasible trajectories are created by solving the following minimal resource

trajectory problem PR_u for each platform u:

$$\text{Minimize} \sum_{k \in \mathcal{K}} \sum_{v \in \mathcal{V}} pr_{u,k,v} z_{u,k,v} \tag{6.60}$$

Subject to

$$\boldsymbol{z_u} \in \boldsymbol{Z_u} \tag{6.61}$$

$$z_{u,k,v} \in \{0,1\} \quad \forall k \in \mathcal{K}, \forall v \in \mathcal{V} \tag{6.62}$$

This problem yields a physically feasible trajectory with minimal required resource cumulative over the visited waypoints on the trajectory. Minimization problem PR_u for fixed platform u is solved by polynomial time Algorithm 3 however with dag-node values $pr_{u,k,v}$. If the cumulative resource consumption over the trajectories for each platform exceeds the limit T, then no feasible solution exists. In this case, the currently processed bp-node is pruned because the sub problems on this branch are infeasible as well. Otherwise, the feasible trajectories are added to the RMP and the column generation procedure can start in order to find an optimal solution to the RMP. The possibly fractional result $\boldsymbol{\lambda}$ represents a relaxed strategy consisting of trajectories for each platform. If the solution is an integer, the lower bound is updated in case it is improved and the bp-node can be pruned. On the other hand, pruning also takes place when the result $\boldsymbol{\lambda}$ is fractional and the corresponding objective value $f(\boldsymbol{\lambda})$ is *less* than or equal to the current lower bound. In both cases the bp-node is pruned because no better solutions can be found further down the tree. If, however, the result $\boldsymbol{\lambda}$ is fractional and the corresponding objective value $f(\boldsymbol{\lambda})$ is higher than the current lower bound, this bp-node is branched to create a set of child bp-nodes. These are then added to the priority queue $\mathcal{Q}$. Two branching strategies are presented and discussed in the next subsection 6.3.2. The process is repeated until the priority queue $\mathcal{Q}$ is empty. The optimal objective value to the MP is represented by the value of LB and the corresponding strategy is an optimal strategy. This algorithm is presented in pseudo code in Algorithm 4.

Algorithm 4 Branch & price for moving target search

Initiate lower bound $LB \leftarrow 0$
Initiate priority queue $\mathcal{Q} \leftarrow$ root bp-node n_0
Initiate column set $\Omega \leftarrow \emptyset$
while $\mathcal{Q} \neq \emptyset$ **do**
  $n \leftarrow$ pull bp-node with highest priority from queue $\mathcal{Q}$
  if RMP has no feasible solution **then**
    $\boldsymbol{\omega} \leftarrow$ generate feasible strategy
    if No feasible strategy exists **then**
      Prune bp-node n
    else
      $\Omega \leftarrow \Omega \cup \boldsymbol{\omega}$
    end if
  end if
  $\boldsymbol{\lambda} \leftarrow$ optimize RMP by column generation
  if Strategy $\boldsymbol{\lambda}$ is integer **then**
    if Objective value $f(\boldsymbol{\lambda}) > LB$ **then**
      Improved lower bound found $LB \leftarrow f(\boldsymbol{\lambda})$
    end if
    Prune bp-node n
  else
    if Objective value $f(\boldsymbol{\lambda}) \leq LB$ **then**
      Prune bp-node n
    else
      $\mathcal{Q} \leftarrow$ branch
    end if
  end if
end while

Computational experiments using this algorithm are presented in section 6.4. Furthermore, the output of this algorithm while solving one specific instance of the search problem is presented in Appendix C.

6.3.2 Branching strategies

Two branching strategies are described in this subsection which can be applied in step (24) in Algorithm 4. The first branching strategy is referred to as *waypoint-branching* and is aligned with the branching strategy from the branch & bound algorithm in section 4.3. It corresponds to extending the standard branching strategy in literature [18, 19, 32, 36] to a branching strategy for the multi-searcher problem. An attempt to use this type of branching for multiple searchers was shown to lead to an extremely large search tree [33] (p.56-59) such that computations are only tractable for very small instances. The waypoint-branching procedure is as follows. Recall from section 4.3 that in the case with one platform, for 3 (for a fixed-wing platform) or 7 (for a rotary-wing platform) allowed waypoints for time $k+1$ a child bp-node is added to the tree. In multi-platform search, a child bp-node is added to the tree for each possible combination of possible waypoints for each of the platforms. With two fixed-wing platforms, the number of child bp-nodes is $3^2 = 9$. With three rotary-wing platforms, the number of child bp-nodes is already $7^3 = 343$ and the problem becomes clearly intractable even on a very small search area.

Therefore, a novel branching strategy is proposed in the following, called *strip-branching*. A search tree using strip-branching is much smaller and, furthermore, is more balanced. A well-balanced tree results in pruning more dominated search trajectories in a bp-node. Basically, one path from the root bp-node to a leaf on the search tree represents a tightly connected set of trajectories. One can imagine the set of trajectories like a strip over the search area, hence the name strip-branching. A bp-node on the branch & price tree restricts, for time k, the waypoint to be within a *set* of waypoints. Selecting from a set of waypoints at time k is different from waypoint-branching, in which one waypoint is fixed for time k. A more formal description of this branching procedure is as follows. Let $n \in \mathcal{Q}$ be the bp-node to branch on with $\mathcal{V}_{u,n}$ the set of allowed nodes on the platform network at time $n.k$ for platform u. The procedure starts with the selection of adjacent nodes on the platform network to $\mathcal{V}_{u,n}$, for each platform $u \in \mathcal{U}$. This set is then divided into two subsets $\mathcal{V}_{1,u}$ and $\mathcal{V}_{2,u}$ in a way that the sum over the $z_{u,k+1,v}$ variables in (6.44) is closest to 0.5 for each subset. The nodes for which $z_{u,k+1,v} = 0$ are equally divided over the subsets. Finally, for

each combination of the subsets $\mathcal{V}_{1,u}$ and $\mathcal{V}_{2,u}$ for all platforms, a bp-node for time $\dot{\kappa}+1$ is created and added to the queue $\mathcal{Q}$. In the case with one platform, each bp-node has two child bp-nodes. In the case with two platforms, each bp-node has four child bp-nodes. In the case with three platforms, each bp-node has eight child bp-nodes, etc. The branch & price tree still grows exponentially fast, however it is significantly slower when compared to waypoint-branching. Even though in all experiments the algorithm converged, this is not the case in theory. Improvements to this strategy are therefore necessary. Nevertheless, the computational results in section 6.4.2 confirm the huge improvement in run times when using strip-branching compared to waypoint-branching.

6.4 Computational experiments

The proposed method is tested in simulations in this section. The proposed linear relaxation is benchmarked against the linear upper bound proposed in [13] and the branch & price procedure with both waypoint-branching and strip-branching options is benchmarked against solving the compact formulation using IBM Cplex.

All tests were performed on an Intel(R) CoreTM i7-4810MQ CPU processor with 2.80 GHz and a usable memory of 15.6 GB. The instance generator is written in Matlab; IBM Cplex with default parameter settings is used to solve the MILPs and the master problem in the branch & price procedure. The branch & price procedure is programmed in Java.

The remainder of this section is structured as follows. First, the test-bed is described in subsection 6.4.1, followed by the simulation results in subsection 6.4.2.

6.4.1 Testbed

The testbed consists of a set of representative test instances, and the set of algorithms to compare their performance in solving those instances. First, a description of the used target properties is given, followed by a description of the properties of the heterogeneous platforms. Finally, a summarizing overview of the algorithms is listed.

The search area in all tests was modeled as a 30×30 square grid. The target under consideration was of the diffusing type. In other words, the target moved north, east, south,

or west, or stayed in its cell, with equal probability in each time step. Furthermore, the start position C_0 of the target was bivariate normally distributed ($C_0 \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$), with $\boldsymbol{\mu} = [15, 15]$ and $\boldsymbol{\Sigma} = [6, 0; 0, 6]$. From the total set of possible target tracks, the number of possible target tracks is restricted to 250 and their probabilities are normalized. For each platform $u \in \mathcal{U}$ an effectiveness of search is given by parameter $W_{u,k,v,c}$, which depends in these experiments on the distance from node v to cell c and the sensor quality as follows:

$$W_{u,k,v,c} = Q_u \left(||v - c|| \right)^{-1}, \tag{6.63}$$

where Q_u is the quality indicator of the sensor on platform u. Search for this target is conducted by a team of heterogeneous platforms. The differing properties are the speed s, radius of a turn l, and sensor quality Q. Table 6.1 lists the properties of the platforms and in which combination they are used to form a team.

Table 6.1: Properties and configurations of the search teams.

Number of platforms N_u				
2	**3**	**4**	**Turn radius** l	**Sensor quality** Q
	x	x	1.0	0.025
	x	x	1.0	0.050
		x	1.5	0.050
		x	2.5	0.100
x			1.5	0.100
	x		2.0	0.150
x			2.5	0.200

The size of the set of nodes for each of the platforms is 50 in each instance. One instance is generated for each of the durations $K = 4, 6, 8, 10, 12, 14$ for each of the team configurations with 2, 3, and 4 platforms. This results in a set of 18 instances, for each of which the shared resource limit is infinite, i.e. $T = \infty$. A similar set of instances is generated for the 2 platform case, with the sole difference being that the shared resource limit is finite and each visited waypoint costs an amount of resources equal to the expected probability of detection of the target at that waypoint. As a result, a total of 24 test instances are generated.

Finally, each instance is solved using 4 algorithms:

- **MILP**: Solving the compact formulation (6.13)-(6.20) using IBM Cplex.
- **BP Waypoints**: Solving the extensive formulation using the proposed branch & price algorithm with the waypoint-branching strategy.
- **BP Strips**: Solving the extensive formulation using the proposed branch & price algorithm with the strip-branching strategy.
- **MILP Old**: Solving the compact formulation (6.13)-(6.20) without the proposed tightening linear overestimators (6.15) and (6.16) using IBM Cplex.

By means of the test results, the runtimes of the branch & price algorithms are compared with solving the MILP using IBM Cplex. These three algorithms solve the problem optimally and thus with equal expected probabilities of detection and equal optimality gaps. The upper bound for the optimality gap is the optimal value of the objective function and the lower bound is determined by calculation of the actual expected probability of detection when using the original glimpse probability as in Equation (2.8). This *new* optimality gap is benchmarked against the *old* optimality gap produced by the **MILP Old** algorithm. The results are presented in the next subsection.

6.4.2 Results and analysis

The computational results are very promising, both from a run time point of view as well as on reducing the optimality gap. The results are summarized in Table 6.2. For each instance, the input parameters K, N_u, and T are listed, followed by the run time of **MILP** in *ms*, the number of nodes processed by **BP Waypoints**, the run time of **BP Waypoints** in *ms*, the number of nodes processed by **BP Strips**, the run time of **BP Waypoints** in *ms*, the old optimality gap and the new optimality gap in *percentages*. In the case that an algorithm has not found an optimal solution after 0.5×10^7 ms, the algorithm was aborted and the symbol "$-$" was noted as a result in Table 6.2.

A comparison of the algorithms **BP Waypoints** and **BP Strips** is done first. Both **BP** algorithms already solve the integer problem in the root bp-node of the branch & price

Table 6.2: Computational results for search by multiple heterogeneous platforms with shared resources (with run times in *ms*, number of nodes in *units*, and the optimality gap in *percentages*).

Input			MILP	BP Waypoints		BP Strips		Opt. gap	
K	N_u	T	Time	Nodes	Time	Nodes	Time	Old	New
4	2	∞	5842	10322	2909070	17	3114	21.79	16.91
6	2	∞	12497	1	2565	1	2571	33.04	6.21
8	2	∞	14310	1	3051	1	3116	30.66	0.36
10	2	∞	28839	1	4569	1	4508	32.03	1.14
12	2	∞	56462	1	5137	1	5079	29.28	3.59
14	2	∞	104770	-	-	9	23016	25.78	7.99
4	3	∞	6474	-	-	33	5394	15.31	14.95
6	3	∞	17315	-	-	57	18080	26.50	15.12
8	3	∞	26955	1	5723	1	5063	34.54	4.22
10	3	∞	35153	1	5240	1	5802	32.37	0.82
12	3	∞	373412	1	8433	1	8567	31.19	0.31
14	3	∞	101736	1	9642	1	9666	26.81	2.01
4	4	∞	17391	-	-	17	11206	17.64	16.03
6	4	∞	39398	-	-	33	28567	27.50	14.07
8	4	∞	52685	1	18523	1	17895	34.27	3.27
10	4	∞	78456	1	23342	1	21847	30.21	0.32
12	4	∞	1846795	1	28014	1	27283	29.75	1.02
14	4	∞	1092396	1	37265	1	37046	28.38	3.56
4	2	0.35	11093	-	-	305	67329	20.40	18.01
6	2	0.5	18132	-	-	85	20448	31.29	8.58
8	2	0.65	21623	-	-	33	26949	32.99	0.81
10	2	0.7	158285	-	-	2881	2289837	33.67	0.51
12	2	0.8	1473179	-	-	1917	2734678	30.71	0.67
14	2	0.9	1271035	-	-	125	4789907	29.33	2.63

tree in many instances. In this case, the algorithms are exactly the same, because the difference exists only in the branching strategy. A logical consequence is that the run times of both algorithms are similar in these cases. However, the strip-branching strategy shows a huge advantage when branching is necessary. The **BP Strips** algorithm only needs to process 17 bp-nodes on the first instance, whereas the **BP Waypoints** algorithm needs to

process over ten thousand bp-nodes. The run time is a factor of 1000 longer. As a result, the **BP Strips** algorithm is favorable over the **BP Waypoints** algorithm in all instances. Now, when comparing the run times of the **BP Strips** algorithm with **MILP**, one sees that the run times of the **BP Strips** algorithm are much shorter in all but one instance with $T = \infty$. The run times improved by as much as 6669%. Consider furthermore that the subproblem optimization for each platform is currently implemented in sequence. Additional improvements can be realized when these subproblems are solved in parallel. When $T < \infty$, however, the number of bp-nodes to process becomes larger and consequently the run times become longer. Nevertheless, the run times remain in the same order of magnitude compared to the run times of **MILP**, so an improvement of the branching strategy will most likely lead to improved run times as well. The resource constrained problem is clearly much harder to solve for all algorithms.

When looking at the optimality gaps in the two rightmost columns of Table 6.2, the superiority of the proposed tighter upper bound becomes very clear. Whereas the old optimality gap is larger than 30% in many of the instances, the new optimality gap is often even lower than 1%. The optimality gaps for each instance are shown in Figure 6-7 as well. The largest difference between the old and the new optimality gaps is obtained on the instance with $K = 10$, $N_u = 2$, and $T = 0.7$. On this instance, the old optimality gap is 33.67% whereas the new optimality gap is as small as 0.51%. An interesting cyclic behavior can be recognized in Figure 6-7. This can be explained by means of Figure 6-3. Recall that the linear overestimator (6.10) results in the old upper bound, whereas the linear overestimator (6.10) combined with the new proposed overestimators (6.11) and (6.12) result in the new upper bound. On instances with just a few time steps available for search, the value of $w_{\psi,\boldsymbol{z}}$ is relatively small. Therefore, the glimpse probability is more likely to be bound from above by the old overestimator (6.10). Resulting in similar values of the old and new optimality gap. However, the value of $w_{\psi,\boldsymbol{z}}$ increases with K due to Equation (6.1). Then, without the glimpse probability being bounded by the new proposed overestimators (6.11) and (6.12) the glimpse probability is only bounded by the value 1 because it is a probability. The obvious result is a large optimality gap. Now if K increases further, the lower bound increases, because the actual value of expected probability of detection is likely

to increase for search strategies of longer duration. Consequently, the optimality gap reduces again slightly. On the other hand, when the glimpse probability is bounded by the new proposed overestimators (6.11) and (6.12), its value lays very close to the actual glimpse probability. This results in a very small new optimality gap.

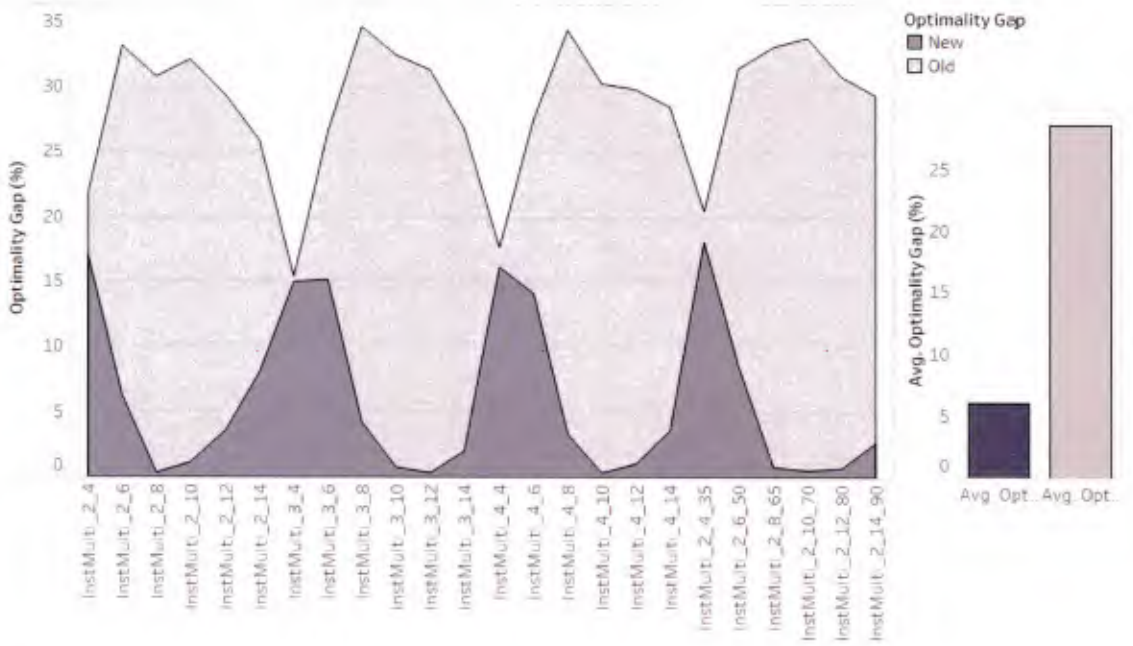

Figure 6-7: Old and new optimality gaps on the instances of the computational experiments.

Finally, the average of the optimality gaps as shown on the right in Figure 6-7 shows the major reduction of the optimality gaps as well when using the proposed upper bound. The average of the old optimality gaps is well over 25%, whereas the average of the old optimality gaps is just a little over 5%.

This major improvement results in a much higher expected probability of detection, which is of course the main goal of search strategy optimization.

Chapter 7

Field experiments towards decision support in search for radiological material

The focus of the work presented in this dissertation so far has mainly been on solving theoretical search problems. However, the solutions to these theoretical problems will be incorporated in a tool to provide decision support in real-life search missions . In this chapter, a first step towards the development of such a tool is described. It consists of a field experiment in order to find the constraints and sensor ranges that provide the input parameters for the algorithms as proposed throughout this dissertation. Subsequently, a fictive but realistic scenario is described in which a suspect carrying radiological material must be detected in order to prevent a terrorist attack. The proposed algorithm 4 for multiple search platforms is used to provide decision support in several scenarios, using realistic input parameters for search effectiveness obtained by the described field experiments. The search effectiveness significantly influences the expected probability of detection throughout the search.

The remainder of this chapter is organized as follows. First, the field experiments for the determination of the gamma-ray sensor ranges are described in section 7.1. Next, in section 7.2, the obtained ranges are used as input parameters for the algorithms used to optimize the

search strategy in computational experiments for realistic search scenarios.

7.1 Determination of gamma-ray sensor ranges

During the period from 20-03-2017 until 23-03-2017, the author, in cooperation with the U.S. Naval Postgraduate School Center for Network Innovation and Experimentation (CENETIX) [90], directed by Prof. Dr. Alexander Bordetsky, conducted field experiments in order to determine the constraints of UAVs equipped with gamma-ray sensors for detection and identification of hazardous material. The experiment took place at the NPS Field Experiment Laboratory. One part of a more extensive field experiment was the determination of gamma-ray sensor ranges when equipped by a DJI Matrice 600 Hexacopter. The test bed of this field experiment is described first in the following subsection, followed by the results.

7.1.1 Field experiment testbed

The aim of this field experiment was to determine the maximal range from the UAV to the target in order to detect the source. The field experiments were conducted for two types of radiological sources and two gamma-ray sensors. The first source was a Cobalt-60 source. This type of radioactive material is frequently used in radiotherapy. It has a half-life of 5.27 years and produces two gamma-rays with energies of 1.17 MeV and 1.33 MeV per photon. The second source, Cesium-137, has a half-life of 30.17 years and produces one gamma-ray with an energy of 0.66 MeV per photon. When measuring radiological material, a spectrum can be established which can be used for the identification of the type of material by looking at the photon peak(s). A spectrum of a Cobalt-60 source shows two peaks at 1173.2 keV and 1332.5 keV, and a spectrum of a Cesium-137 source shows one peak at 662 keV as shown in Figure 7-8. Two sensors were tested. Both sensors are high-efficiency radio-isotope identification devices and will be referred to by sensor 1 and sensor 2. In each round of experiments, the DJI Matrice 600 Hexacopter was equipped with one of the sensors. Sensor 2 automatically sends detection events with information to the remote advice and assist cell (see Figure 7-1), whereas the ranges from sensor 1 needed to be reported manually as shown in Figure 7-2.

Figure 7-1: The remote advice and assist cell at the NPS Field Experiment Laboratory.

7.1.2 Field experiment results

All the data gathered in this experiment is publicly available at the CENETIX Resource Portal [91] in the Observer's Notepad under chapters WAS CR PhI-PhIV. The maximal sensor ranges as determined from the field experiments are listed in Table 7.1 and range between 3 m and 6 m. Sensor 2 appears to be a more sensitive sensor, able to pick up gamma-rays from a larger distance compared to sensor 1. Cobalt-60 has a higher gamma-ray dose constant compared to Cesium-137 and can therefore be described as the stronger source. This results in a larger possible range of detection for a Cobalt-60 source. Finally, the speed of the UAV did not seem to influence the detection rate. The fastest tested speed of 40 km/h still yielded positive detection events.

Table 7.1: Approximate sensor ranges found in field experiments.

	Sensor 1	**Sensor 2**
Cobalt-60	4 m	6 m
Cesium-137	3 m	5 m

The spectra of a selection of measurements from sensor 1 on the Cesium-137 source are

Figure 7-2: M. Raap at the remote advice and assist cell receiving information from the field.

Figure 7-3: Heat map overlay generated by the sensor 1 software. The color red represents relatively high radiation, accurately suggesting that the source is located in that area in the cabinet.

shown in Figures 7-4, 7-6, and7-8. The spectrum shown in 7-4 was taken at a distance of 6m, with a corresponding image taken by the imaging component of sensor 1 as shown in Figure 7-5. Here, the UAV is too far away from the source for a detection event. As a result, no spike is visible in the spectrum. At a distance of 3 m, a detection event occurred. The spectrum of the measurement at that distance is shown in Figure 7-6, showing a spike at 662 keV suggesting the presence of a Cesium-137 source. A corresponding image is shown in Figure 7-7. The strongest signal was received when the UAV was hovering over the source. The respective spectrum is shown in Figure 7-8 showing a large spike, with an image shown in Figure 7-9. The GPS track of the UAV when equipped with sensor 1 is shown in Figures 7-10 and 7-11, along with the locations of both radiological sources.

7.2 Computational experiments for detection of a radiological dispersion device

In this section, computational experiments are deduced for a realistic scenario in order to show the applicability of the proposed method for optimization of a search strategy for multiple heterogeneous platforms. The input parameters of the algorithms used in these

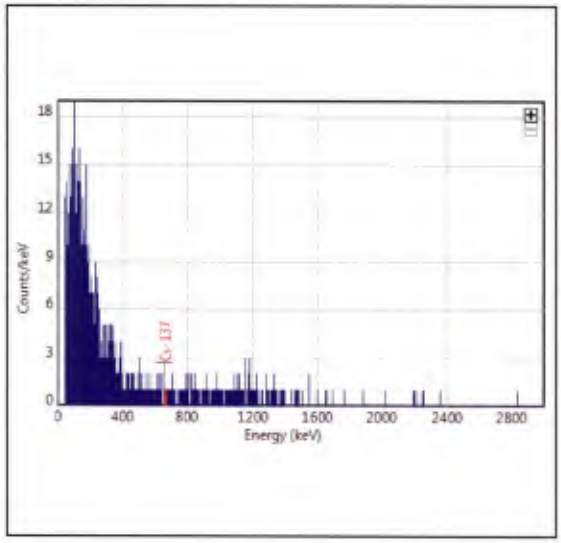

Figure 7-4: Spectrum by sensor 1 at 6m distance to the Cesium-137 source. No detection has taken place due to the large distance.

Figure 7-5: Image by sensor 1 at 6m distance to the Cesium-137 source.

experiments are either estimations of real-life units such as geographical distances, or determined by the field experiments described in the previous section. First, a scenario of a radiological dispersion threat in the center of San Francisco is described in the following subsection, followed by a model of the search mission and a description of the instances used in the computational experiments. The results of these computational experiments are presented and analyzed in the final subsection of this chapter.

7.2.1 Scenario of a radiological dispersion threat in San Francisco

Imagine the following scenario describing a terrorist threat in the center of San Francisco, CA, USA: A person who is associated with a terrorist group is carrying a radiological dispersion device (RDD) at Union Square. A radiological dispersion device is "any device, including any weapon or equipment, other than a nuclear explosive device, specifically designed to employ radioactive material by disseminating it to cause destruction, damage, or injury by means of the radiation produced by the decay of such material" [92]. The dispersion effect of an RDD is limited and will not cause many casualties. However, the psychological effects, such as fear and overreaction from civilians, are much larger. The

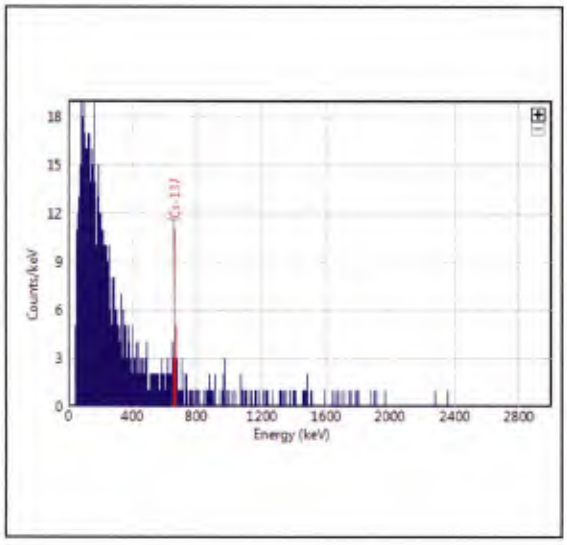

Figure 7-6: Spectrum by sensor 1 at 3m distance to the Cesium-137 source. A detection event had occurred and the material was identified.

Figure 7-7: Image by sensor 1 at 3m distance to the Cesium-137 source.

typical aim of detonating such a device is therefore to disrupt rather than destroy [93]. The greatest security risk comes from, a.o., Cobalt-60 and Cesium-137 [93]. In this scenario, a positive signal is received at Union Square but, because of the range to detection, the person carrying the RDD is not immediately identified and consequently lost. A search mission is initiated to search for the target with a team of search platforms, each equipped with either sensors.

7.2.2 Modeling of the San Francisco search mission

To provide decision support, the following model of the search mission is set up. First, the search area as depicted in Figure 7-12 is discretized into 50×50 equally-sized square cells. The target is estimated to be moving south and west with probability 0.3, and with probability 0.2 to the north and east on this grid. Another grid with the dimension 10×10 is additionally generated for search strategy optimization. The latter grid contains fewer cells in order to keep the problem tractable. Finally, a well informed estimation for the search effectiveness of each sensor-source combination is a necessary input for the optimization algorithms. These estimated values for search effectiveness can be calculated as follows.

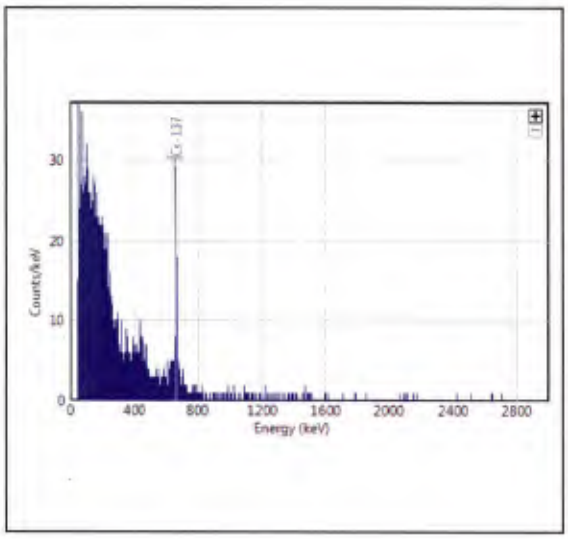

Figure 7-8: Spectrum by sensor 1 while hovering over the Cesium-137 source. A strong detection event had occurred and the material was identified.

Figure 7-9: Image by sensor 1 while hovering over the Cesium-137 source.

The search area has the dimension 20.71 km $\times$ 20.71 km, which corresponds to 428.90 km^2, leading to 4.29km^2 on one cell on the search grid. On average, 20% of this area is reachable by neither the target nor the search platforms. The remainder of the area consists 60% of buildings and the remaining 40% is outdoors. A time period k is modeled to represent 30 min in real life. Furthermore, each platform moves at 20 km/h when outdoors and 5 km/h when indoors. Consequently, a platform u covers $2.5 \times 2r_u$ km^2 in a time period when indoors, and covers $8.0 \times 2r_u$ km^2 when outdoors. Here, the parameter r_u is the sensor range in km of the sensor equipped by platform u. Consequently, a fraction of $(2.5/2.06)2r_u$ of the indoor area and a fraction of $(8.0/1.37)2r_u$ of the outdoor area is covered by platform u, resulting in an overall coverage fraction of $6.13r_u$. Substituting the sensor ranges from table 7.1 results in the search effectiveness parameters as listed in table 7.2.

Table 7.2: Search effectiveness estimations for each sensor-source combination.

	Sensor 1	**Sensor 2**
Cobalt-60	0.03064	0.03677
Cesium-137	0.01838	0.03064

Figure 7-10: GPS track of the DJI Matrice 600 Hexacopter (blue diamond) with sensor 1 near the Cesium-137 source (orange diamond).

Figure 7-11: GPS track of the DJI Matrice 600 Hexacopter (blue diamond) with sensor 1 near the Cobalt-60 source (red diamond).

Obviously, some strong assumptions on the movements of the radiological source and on the detection rates are made for this computational experiment. The aim of these experiments is therefore by no means to predict the actual expected probability of detection, but rather an example of how the proposed algorithm behaves in a more realistic scenario, compared to existing algorithms. The optimal search strategies were calculated for a total of 28 instances, 14 for both types of radiological material, Cobalt-60 and Cesium-137. For team sizes of 3 and 4 platforms a number of 7 instances were generated with mission duration $K = 12, 14, 16, 18, 20, 22, 24$. Since one time period is modeled to last 30 minutes, $K = 12$ represents a search duration of 6 hours in real time. In a team of size four, two platforms had sensor 1 equipped and two platforms had sensor 2 equipped. In a team of size three, two platforms had sensor 1 equipped and one platform had an sensor 2 equipped. The search effectiveness parameters were used as described in Table 7.2. The results for each of these instances are presented and evaluated in the next subsection.

7.2.3 Computational results

The results of the computational experiments for search for a radiological dispersion device, as shown in Table 7.3, show the applicability of the model and algorithms to support decision making in complex search scenarios. The left part of the table contains the input parameters

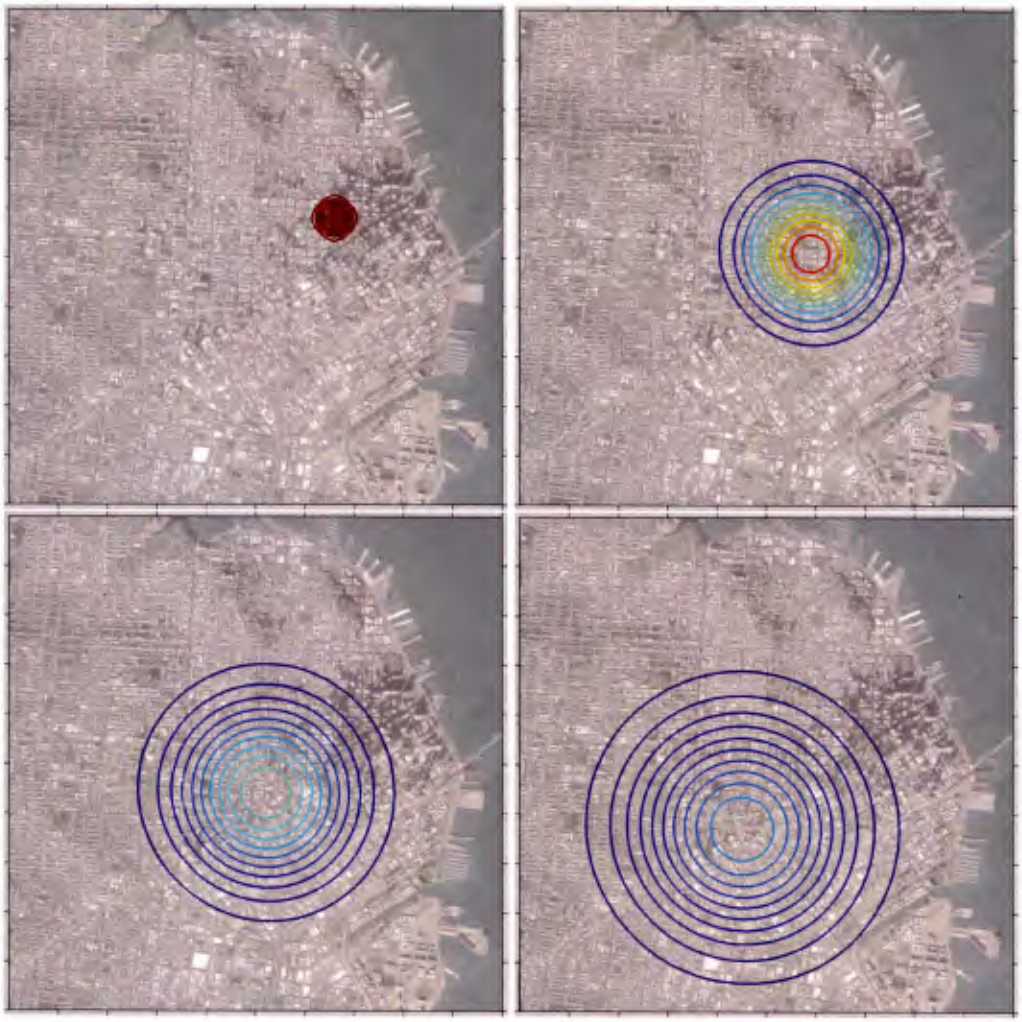

Figure 7-12: Maps of San Francisco with heat map overlays, evolving according to the RDD motion model.

mission duration K, team size N_u, and source. The right part of the table contains the corresponding computational results. Each instance was solved using IBM-Cplex (**MILP**) and using the proposed branch & price algorithm with the strip-branching strategy (**BP Strips**) from chapter 6. The computation times are shown for both algorithms, as well as the number of nodes processed by the branch & price algorithm. The resulting expected probability of detection is given, as well as the old and the new optimality gaps as described in chapter 6.

A straightforward observation can be made from Table 7.3: the expected probability of detection increases with the duration of the search, as well as with the number of platforms.

Table 7.3: Computational results for search for a radiological dispersion device by multiple heterogeneous platforms.

Input			MILP	BP Strips			Opt. gap (%)	
K	N_u	Source	Time (ms)	Nodes	Time (ms)	PD	Old	New
12	3	Cobald-60	6344	1	3671	0.26	13.93	13.93
14	3	Cobald-60	5266	1	4775	0.26	14.31	14.31
16	3	Cobald-60	9788	1	4142	0.35	20.33	16.03
18	3	Cobald-60	24428	121	39379	0.40	24.55	13.34
20	3	Cobald-60	11879	1	4874	0.50	33.19	6.36
22	3	Cobald-60	14605	1	6556	0.61	35.28	3.45
24	3	Cobald-60	10070	1	6978	0.75	36.60	0.00
12	3	Cesium-137	4935	1	3890	0.19	10.98	10.98
14	3	Cesium-137	12020	1	3605	0.30	14.70	14.70
16	3	Cesium-137	7419	1	3794	0.39	19.14	19.14
18	3	Cesium-137	9344	1	4260	0.53	22.70	21.73
20	3	Cesium-137	6183	1	5076	0.62	33.38	5.72
22	3	Cesium-137	19370	1	6338	0.57	27.25	13.54
24	3	Cesium-137	8688	1	6261	0.64	35.27	4.02
12	4	Cobald-60	8383	97	85954	0.37	18.62	17.21
14	4	Cobald-60	20098	57	90846	0.44	21.99	20.21
16	4	Cobald-60	9418	1	6245	0.59	34.41	1.96
18	4	Cobald-60	10073	1	6809	0.58	35.11	5.58
20	4	Cobald-60	12437	1	7019	0.67	34.42	3.34
22	4	Cobald-60	10714	1	7715	0.84	36.16	2.08
24	4	Cobald-60	11907	1	7960	0.86	35.95	3.66
12	4	Cesium-137	7721	1	4835	0.23	12.88	12.88
14	4	Cesium-137	6931	1	5724	0.30	16.19	15.06
16	4	Cesium-137	18529	1	6155	0.48	26.19	12.72
18	4	Cesium-137	11526	1	6505	0.60	30.70	8.71
20	4	Cesium-137	11834	1	7174	0.65	36.26	3.26
22	4	Cesium-137	11962	1	7115	0.67	36.61	1.66
24	4	Cesium-137	13437	1	7503	0.70	35.50	0.65

When comparing the results of search for a Cobalt-60 and a Cesium-137 source, one can see that the expected probability of detection for the Cobalt-60 source is higher for an equal number of platforms and search duration. This can be explained by the larger ranges

necessary for detection, as presented in Table 7.1 and consequently the search effectiveness for this type of source is higher, as presented in Table 7.2. For most of the instances, the branch & price algorithm finds an optimal integer solution in the root node. In all these cases, the branch & price algorithm solves the problem faster than IBM Cplex solves the MILP formulation. In just a few cases multiple nodes needed processing, resulting in a longer run time. In such cases, solving the MILP formulation can produce a timely and optimal solution. Consider furthermore that the subproblem optimization for each platform is currently implemented in sequence. Additional improvements in terms of computation time can be realized when these subproblems are solved in parallel.

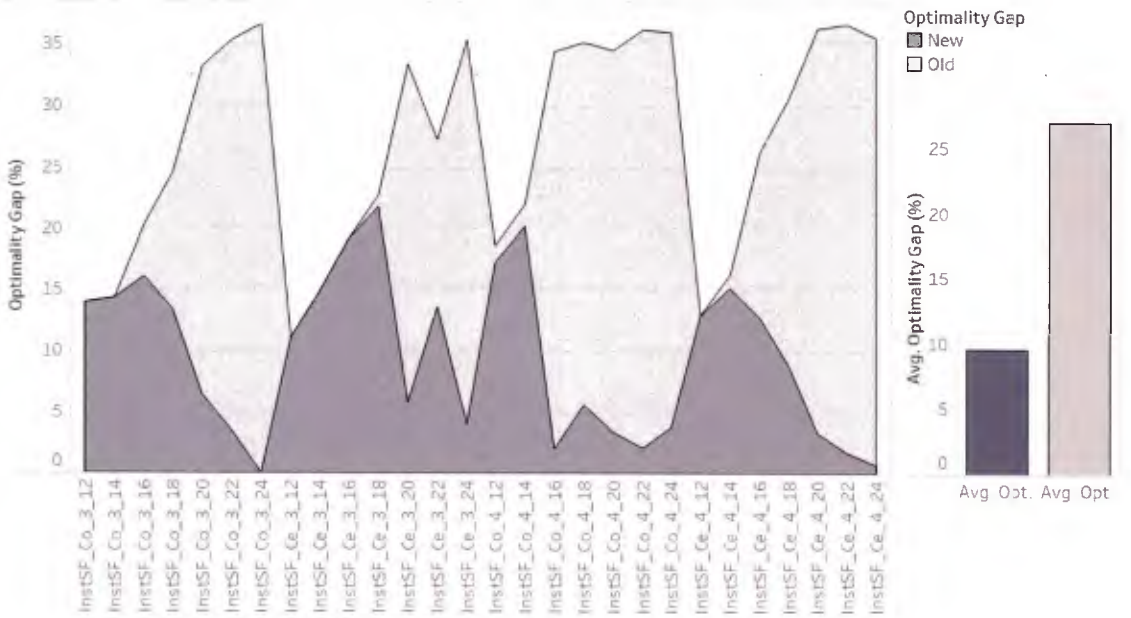

Figure 7-13: Old and new optimality gaps on the instances of the computational experiments.

Finally, a large reduction of the optimality gap is made. In one case, the instance with the Cobalt-60 source, $K = 24$ and $N_u = 3$, the optimality gap was reduced from 36.6% to less than 0.01%. The optimality gaps resulting from both linear upper bounds are shown graphically as well for each instance in Figure 7-13. The average optimality gap of the proposed upper bound is significantly smaller compared to the average optimality gap of the old upper bound, namely just a little under 10% versus over 25%.

Altogether, the proposed model and algorithms find optimal search strategies in a very short time span, such that the search can start as soon as the platforms are ready for

deployment.

Chapter 8

Conclusions

The search for missing persons or wreckage in non-urban environments such as at sea or in the mountains has been necessary in the past, and will be necessary in the future. Disasters in aviation and the freight industry, environmental disasters, mistakes or unfortunate series of events on extreme adventure trips, and terrorist threats all can potentially be the basis of a coordinated search mission. Search missions are often conducted by sensor equipped ground or aerial vehicles or vessels. For example, the Airbus Military C-295 is a typically employed in naval search missions. With the increasing technological improvements of unmanned aerial vehicles, it becomes more and more attractive to additionally employ such vehicles as well. Due to the complexity of timely search mission planning, it is important for the mission leader to utilize decision support in the form of a search strategy and for the pilot to utilize decision support in the form of an optimal flight trajectory.

Methods for these types of decision support are provided in this dissertation in chapters 4-7. A structural overview of these chapters is presented as a flowchart in Figure 8-1. The blocks on the left represent the new methods presented in chapters 4-6. The method in chapter 6 for cooperative search by a team of heterogeneous platforms with shared resources is used to find optimal search strategies in search for radiological material in a scenario of terrorist threat as described in chapter 7 and represented by the middle block in the lowest row in the flowchart. Finally, based upon this work, the ongoing research and development of a decision support tool for wide area search is represented by the final block in the flowchart. Each element will be summarized in the following.

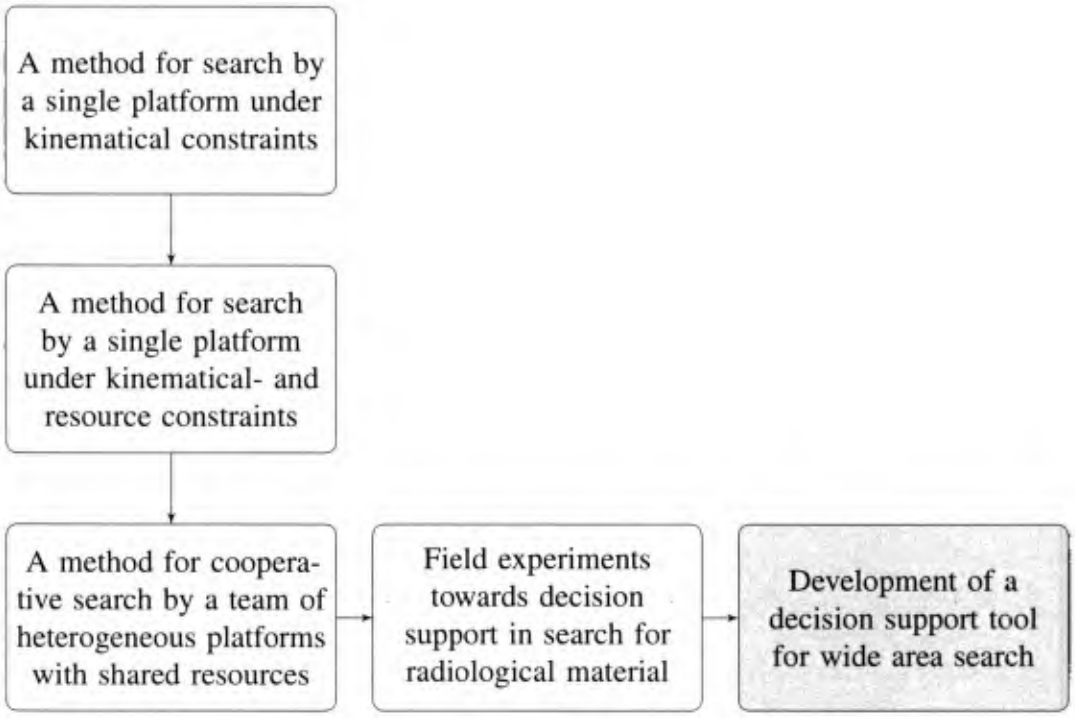

Figure 8-1: Flowchart of the chapters containing the main contributions of this dissertation. The white blocks represent its content and the gray block represents the ongoing research and development based on the presented work.

- **In chapter 4, a method for search by a single platform under kinematical constraints is presented.**

A physically feasible search trajectory is planned depending on the search speed, relocation speed, and the minimum turn radius of the sensor platform. Depending on its sensor performance as well as the initial probability map of the location of the target and its probabilistic motion, a search trajectory is then optimized by solving the proposed mixed integer linear program. This program is efficiently solved by a generalization of an existing branch & bound algorithm for moving target search to be applicable to kinematical constraints and heterogeneous grids. The results from computational experiments show the applicability of the proposed model and the effectiveness of the algorithm. Improvements can be accomplished by finding a tighter and more easily computable upper bound on the probability of detection, which has been an extensively researched goal since 1979.

- **In chapter 5, a method for search by a single platform under kinematical and resource constraints is presented.**

In certain situations, the search trajectory is subject to limited resources. In this context, a resource is a collective concept for e.g. fuel or risk. Solving the problem of finding a physically feasible search trajectory with limited resources that maximizes the probability of detection can not be solved efficiently by the previous branch & bound algorithm. Therefore, a Benders' decomposition algorithm was proposed to solve the problem efficiently. This is, to the best of the author's knowledge, the first Benders' decomposition approach for target search in general in literature. For the case with scarce resources the proposed method solves significantly faster compared to solving the proposed mixed integer programming formulation using a commercial solver. A probable improvement to this solution could be to find a stabilization technique for this specific Benders' decomposition algorithm.

- **In chapter 6, a method for cooperative search by a team of heterogeneous platforms with shared resources is presented.**

When multiple platforms are available for search, a search strategy is necessary for effective cooperative search. Planning such a search strategy is inherently non-linear and therefore even harder to solve. Moreover, the problem grows exponentially in the number of platforms. In this chapter, an easier to solve problem was proposed that yields an upper bound on the joint probability of detection. The proposed upper bound is tight. Computational experiments show an optimality gap of less than 1% on many instances, which is a large improvement of an existing upper bound of ca. 30% on the same instances. Furthermore, a branch & price procedure is proposed for solving the problem more efficiently. Especially in cases with no resource constraints, the run time of the algorithm is up to two orders of magnitude shorter compared to solving the problem using the currently best commercial branch & bound solver, IBM Cplex. Improvements to the upper bound can be made by fine tuning the point of tangency of the overestimators as well as the number of overestimators. The most promising improvement to the branch & price procedure can be accomplished by finding further improvements of the proposed branching strategy.

- **In chapter 7, the proposed branch & price method is applied to a more realistic scenario.**

The proposed branch & price method is applied to a more realistic scenario in order to show how it could provide decision support in a real-life search mission. First, by means of field experiments in cooperation with the U.S. Naval Postgraduate School Center for Network Innovation and Experimentation (CENETIX), the ranges for detection of two radiological sensors have been determined. These sensors are used to search for a moving suspect carrying a radiological dispersion device in a fictive but realistic scenario in the center of San Francisco. The branch & price algorithm is then used to optimize the search strategy for a team of multiple heterogeneous search platforms, taking the actual measured sensor ranges as input parameters. The results of computational experiments regarding this scenario show the efficiency of the proposed algorithm in terms of run time, such that an optimal search strategy is present at the start of the search mission. This chapter provides a conceptual description of a decision support tool.

- **Ongoing research and development of a decision support tool for wide area search.**

At the time of writing, a Master's student from the Department of Computer Science at the Universität der Bundeswehr München is implementing a part of this decision support tool in the CENETIX Resource Portal [90]. In September 2017, extensive field experiments for wide area search will be conducted in the San Francisco Bay Area supported by this tool. The largest challenges toward the deployment of such a tool are mainly the determination of an accurate motion model of the target, as well as accurate values for search effectiveness.

Finally, the methods proposed in this dissertation are focused to be applicable to manned and unmanned, rotary-wing and fixed-wing, autonomous and manually controlled aerial vehicles. When multiple aerial vehicles are employed for search, they are assumed to fly at different altitudes in order to avoid collisions. However, the methods are not restricted to deploy aerial vehicles, but are applicable to other types of searchers, e.g. ground vehicles

and vessels, as well. As a final note, the overall result of a search mission can be improved when sensor scheduling and communication constraints are taken into account.

Appendix A

Output of the branch & bound algorithm

```
Instance: Stay. Prob = 0.0, K = 4, Q = 0.5, bound = MEAN (see Table 4.1)
Queue size: 1; k: -1; UB: 1.7976931348623157E308  LB: 0.0; []; Branch
Queue size: 41; k: 0; UB: 0.4981277200522136  LB: 0.0; [(0,25)]; Branch
Queue size: 46; k: 1; UB: 0.4411029406304844  LB: 0.0; [(0,25), (1,18)]; Branch
Queue size: 48; k: 2; UB: 0.4215668252748087  LB: 0.0; [(0,25), (1,18), (2,12)]; Branch
Queue size: 50; k: 3; UB: 0.4215668252748087  LB: 0.41955721301426113; [(0,25), (1,18), (2,12), (3,13)]; Leaf
>>> New best trajectory found
Queue size: 49; k: 3; UB: 0.4215668252748087  LB: 0.414507535834295; [(0,25), (1,18), (2,12), (3,6)]; Leaf
Queue size: 48; k: 3; UB: 0.4215668252748087  LB: 0.41401819521056693; [(0,25), (1,18), (2,12), (3,5)]; Leaf
Queue size: 47; k: 2; UB: 0.418198217198972  LB: 0.41955721301426113; [(0,25), (1,18), (2,11)]; Prune
Queue size: 46; k: 2; UB: 0.41616155307235836  LB: 0.41955721301426113; [(0,25), (1,18), (2,17)]; Prune
Queue size: 45; k: 1; UB: 0.4411029406304844  LB: 0.41955721301426113; [(0,25), (1,19)]; Branch
Queue size: 47; k: 2; UB: 0.4215668252748087  LB: 0.41955721301426113; [(0,25), (1,19), (2,12)]; Branch
Queue size: 49; k: 3; UB: 0.4215668252748087  LB: 0.41955721301426113; [(0,25), (1,19), (2,12), (3,11)]; Leaf
Queue size: 48; k: 3; UB: 0.4215668252748087  LB: 0.414507535834295; [(0,25), (1,19), (2,12), (3,5)]; Leaf
Queue size: 47; k: 3; UB: 0.4215668252748087  LB: 0.41401819521056693; [(0,25), (1,19), (2,12), (3,6)]; Leaf
Queue size: 46; k: 2; UB: 0.418198217198972  LB: 0.41955721301426113; [(0,25), (1,19), (2,13)]; Prune
Queue size: 45; k: 2; UB: 0.41616155307235836  LB: 0.41955721301426113; [(0,25), (1,19), (2,20)]; Prune
Queue size: 44; k: 1; UB: 0.4403974075842623  LB: 0.41955721301426113; [(0,25), (1,26)]; Branch
Queue size: 46; k: 2; UB: 0.41375328590348626  LB: 0.41955721301426113; [(0,25), (1,26), (2,20)]; Prune
Queue size: 45; k: 2; UB: 0.40492049560563526  LB: 0.41955721301426113; [(0,25), (1,26), (2,27)]; Prune
Queue size: 44; k: 2; UB: 0.4040577137299878  LB: 0.41955721301426113; [(0,25), (1,26), (2,33)]; Prune
Queue size: 43; k: 1; UB: 0.44039740758426227  LB: 0.41955721301426113; [(0,25), (1,24)]; Branch
Queue size: 45; k: 2; UB: 0.4137532859034862  LB: 0.41955721301426113; [(0,25), (1,24), (2,17)]; Prune
Queue size: 44; k: 2; UB: 0.4049204956056352  LB: 0.41955721301426113; [(0,25), (1,24), (2,23)]; Prune
Queue size: 43; k: 2; UB: 0.4040577137299878  LB: 0.41955721301426113; [(0,25), (1,24), (2,30)]; Prune
Queue size: 42; k: 1; UB: 0.43577046590686225  LB: 0.41955721301426113; [(0,25), (1,31)]; Branch
Queue size: 44; k: 2; UB: 0.4010257829301909  LB: 0.41955721301426113; [(0,25), (1,31), (2,30)]; Prune
Queue size: 43; k: 2; UB: 0.39514421428876034  LB: 0.41955721301426113; [(0,25), (1,31), (2,37)]; Prune
Queue size: 42; k: 2; UB: 0.3925855654025549  LB: 0.41955721301426113; [(0,25), (1,31), (2,36)]; Prune
Queue size: 41; k: 1; UB: 0.43577046590686214  LB: 0.41955721301426113; [(0,25), (1,32)]; Branch
Queue size: 43; k: 2; UB: 0.40102578293019087  LB: 0.41955721301426113; [(0,25), (1,32), (2,33)]; Prune
Queue size: 42; k: 2; UB: 0.3951442142887603  LB: 0.41955721301426113; [(0,25), (1,32), (2,37)]; Prune
Queue size: 41; k: 2; UB: 0.3925855654025549  LB: 0.41955721301426113; [(0,25), (1,32), (2,38)]; Prune
Queue size: 40; k: 0; UB: 0.49824598277517484  LB: 0.41955721301426113; [(0,19)]; Branch
Queue size: 45; k: 1; UB: 0.44095661481704346  LB: 0.41955721301426113; [(0,19), (1,25)]; Branch
```

```
Queue size: 47; k: 2; UB: 0.4201520312262602  LB: 0.41955721301426113; [(0,19), (1,25), (2,24)]; Branch
Queue size: 49; k: 3; UB: 0.4201520312262602  LB: 0.41675161450683773; [(0,19), (1,25), (2,24), (3,17)]; Leaf
Queue size: 48; k: 3; UB: 0.4201520312262602  LB: 0.41286335928458684; [(0,19), (1,25), (2,24), (3,23)]; Leaf
Queue size: 47; k: 3; UB: 0.4201520312262602  LB: 0.4120175882184358; [(0,19), (1,25), (2,24), (3,30)]; Leaf
Queue size: 46; k: 2; UB: 0.4164166269207039  LB: 0.41955721301426113; [(0,19), (1,25), (2,31)]; Prune
Queue size: 45; k: 2; UB: 0.41586538933777006  LB: 0.41955721301426113; [(0,19), (1,25), (2,32)]; Prune
Queue size: 44; k: 1; UB: 0.4414619567380097  LB: 0.41955721301426113; [(0,19), (1,18)]; Branch
Queue size: 46; k: 2; UB: 0.4201238385379238  LB: 0.41955721301426113; [(0,19), (1,18), (2,24)]; Branch
Queue size: 48; k: 3; UB: 0.4201238385379238  LB: 0.4161328491935187; [(0,19), (1,18), (2,24), (3,31)]; Leaf
Queue size: 47; k: 3; UB: 0.4201238385379238  LB: 0.4121718240459119; [(0,19), (1,18), (2,24), (3,30)]; Leaf
        .
        .
        .
Queue size: 13; k: 1; UB: 0.37858438111059334  LB: 0.42173750336160015; [(0,29), (1,35)]; Prune
Queue size: 12; k: 1; UB: 0.37888810945995427  LB: 0.42173750336160015; [(0,29), (1,22)]; Prune
Queue size: 11; k: 0; UB: 0.46451584374850174  LB: 0.42173750336160015; [(0,2)]; Branch
Queue size: 13; k: 1; UB: 0.4119482888843435  LB: 0.42173750336160015; [(0,2), (1,6)]; Prune
Queue size: 12; k: 1; UB: 0.4014075371768325  LB: 0.42173750336160015; [(0,2), (1,7)]; Prune
Queue size: 11; k: 1; UB: 0.38454554555464354  LB: 0.42173750336160015; [(0,2), (1,1)]; Prune
Queue size: 10; k: 0; UB: 0.46451584374850163  LB: 0.42173750336160015; [(0,0)]; Branch
Queue size: 12; k: 1; UB: 0.4119482888843434  LB: 0.42173750336160015; [(0,0), (1,5)]; Prune
Queue size: 11; k: 1; UB: 0.4014075371768324  LB: 0.42173750336160015; [(0,0), (1,4)]; Prune
Queue size: 10; k: 1; UB: 0.38454554555464354  LB: 0.42173750336160015; [(0,0), (1,1)]; Prune
Queue size: 9; k: 0; UB: 0.45918642492075706  LB: 0.42173750336160015; [(0,39)]; Branch
Queue size: 11; k: 1; UB: 0.4079808388098804  LB: 0.42173750336160015; [(0,39), (1,33)]; Prune
Queue size: 10; k: 1; UB: 0.397859354274288  LB: 0.42173750336160015; [(0,39), (1,38)]; Prune
Queue size: 9; k: 1; UB: 0.38004476992909275  LB: 0.42173750336160015; [(0,39), (1,34)]; Prune
Queue size: 8; k: 0; UB: 0.45918642492075706  LB: 0.42173750336160015; [(0,35)]; Branch
Queue size: 10; k: 1; UB: 0.40798083880988045  LB: 0.42173750336160015; [(0,35), (1,30)]; Prune
Queue size: 9; k: 1; UB: 0.397859354274288  LB: 0.42173750336160015; [(0,35), (1,36)]; Prune
Queue size: 8; k: 1; UB: 0.38004476992909286  LB: 0.42173750336160015; [(0,35), (1,29)]; Prune
Queue size: 7; k: 0; UB: 0.46054550738230465  LB: 0.42173750336160015; [(0,28)]; Branch
Queue size: 9; k: 1; UB: 0.408560892595028  LB: 0.42173750336160015; [(0,28), (1,27)]; Prune
Queue size: 8; k: 1; UB: 0.3968252551043142  LB: 0.42173750336160015; [(0,28), (1,21)]; Prune
Queue size: 7; k: 1; UB: 0.3790258298862931  LB: 0.42173750336160015; [(0,28), (1,34)]; Prune
Queue size: 6; k: 0; UB: 0.46054550738230465  LB: 0.42173750336160015; [(0,22)]; Branch
Queue size: 8; k: 1; UB: 0.408560892595028  LB: 0.42173750336160015; [(0,22), (1,23)]; Prune
Queue size: 7; k: 1; UB: 0.3968252551043142  LB: 0.42173750336160015; [(0,22), (1,16)]; Prune
Queue size: 6; k: 1; UB: 0.3790258298862931  LB: 0.42173750336160015; [(0,22), (1,29)]; Prune
Queue size: 5; k: 0; UB: 0.453501278294147  LB: 0.42173750336160015; [(0,15)]; Branch
Queue size: 7; k: 1; UB: 0.4015653594238919  LB: 0.42173750336160015; [(0,15), (1,14)]; Prune
Queue size: 6; k: 1; UB: 0.3958019077620494  LB: 0.42173750336160015; [(0,15), (1,21)]; Prune
Queue size: 5; k: 1; UB: 0.370209044477105 6  LB: 0.42173750336160015; [(0,15), (1,8)]; Prune
Queue size: 4; k: 0; UB: 0.453501278294147  LB: 0.42173750336160015; [(0,9)]; Branch
Queue size: 6; k: 1; UB: 0.40156535942389193  LB: 0.42173750336160015; [(0,9), (1,10)]; Prune
Queue size: 5; k: 1; UB: 0.3958019077620494  LB: 0.42173750336160015; [(0,9), (1,16)]; Prune
Queue size: 4; k: 1; UB: 0.370209044477105  LB: 0.42173750336160015; [(0,9), (1,3)]; Prune
Queue size: 3; k: 0; UB: 0.4526143179679348  LB: 0.42173750336160015; [(0,3)]; Branch
Queue size: 5; k: 1; UB: 0.40030901923391027  LB: 0.42173750336160015; [(0,3), (1,10)]; Prune
Queue size: 4; k: 1; UB: 0.39801359047506374  LB: 0.42173750336160015; [(0,3), (1,4)]; Prune
Queue size: 3; k: 1; UB: 0.36986553653213994  LB: 0.42173750336160015; [(0,3), (1,9)]; Prune
Queue size: 2; k: 0; UB: 0.45261431796793483  LB: 0.42173750336160015; [(0,8)]; Branch
Queue size: 4; k: 1; UB: 0.4003090192339103  LB: 0.42173750336160015; [(0,8), (1,14)]; Prune
Queue size: 3; k: 1; UB: 0.3980135904750637  LB: 0.42173750336160015; [(0,8), (1,7)]; Prune
Queue size: 2; k: 1; UB: 0.36986553653213994  LB: 0.42173750336160015; [(0,8), (1,15)]; Prune
Queue size: 1; k: 0; UB: 0.4467578731541683  LB: 0.42173750336160015; [(0,40)]; Branch
Queue size: 2; k: 1; UB: 0.39677139352811197  LB: 0.42173750336160015; [(0,40), (1,37)]; Prune
Queue size: 1; k: 1; UB: 0.39444875213511554  LB: 0.42173750336160015; [(0,40), (1,36)]; Prune
#nodes processed: 510
#nodes pruned: 354
Optimal search trajectory: [(0,12), (1,18), (2,25), (3,26)], with PD: 0.42173750336160015
```

Appendix B

Output of the Benders' decomposition algorithm

```
Instance: Stay. Prob = 0.4, K = 8, T = 1.55 (see Table 5.3)
UB: 1000.0  LB: 0.6568849076774645; [(0,9), (1,3), (2,0), (3,1), (4,2), (5,6), (6,7), (7,14)]
>>> Accepting new incumbent with value 0.6568849076774645
UB: 0.6568849076772949  LB: 0.6568849076774645; [(0,9), (1,3), (2,0), (3,1), (4,2), (5,6), (6,7), (7,14)]
UB: 248.31873585861527  LB: 0.6555833451711454; [(0,2), (1,6), (2,7), (3,14), (4,20), (5,27), (6,33), (7,40)
UB: 244.8847112396549  LB: 0.6536467709366761; [(0,3), (1,0), (2,1), (3,2), (4,6), (5,7), (6,14), (7,20)]
UB: 20.519563135668935  LB: 0.658447000760591; [(0,3), (1,4), (2,1), (3,2), (4,6), (5,7), (6,14), (7,20)]
>>> Accepting new incumbent with value 0.658447000760591
UB: 0.6584470007591925  LB: 0.658447000760591; [(0,3), (1,4), (2,1), (3,2), (4,6), (5,7), (6,14), (7,20)]
UB: 210.35488730201297  LB: 0.6588656665761777; [(0,40), (1,33), (2,27), (3,20), (4,14), (5,7), (6,6), (7,2)]
>>> Accepting new incumbent with value 0.6588656665761777
UB: 0.658865666576693  LB: 0.6588656665761777; [(0,40), (1,33), (2,27), (3,20), (4,14), (5,7), (6,6), (7,2)]
UB: 0.6588656650362914  LB: 0.6588656665761777; [(0,40), (1,33), (2,27), (3,20), (4,14), (5,7), (6,6), (7,2)]
UB: 219.54034991584996  LB: 0.6563269993115337; [(0,14), (1,7), (2,6), (3,2), (4,1), (5,4), (6,3), (7,9)]
UB: 215.47076630590368  LB: 0.656180116430844; [(0,27), (1,20), (2,14), (3,7), (4,6), (5,2), (6,1), (7,0)]
UB: 223.5063385977126  LB: 0.6560305299870032; [(0,7), (1,6), (2,2), (3,1), (4,4), (5,3), (6,9), (7,8)]
UB: 240.20800375931407  LB: 0.656495138520342; [(0,0), (1,1), (2,5), (3,6), (4,7), (5,14), (6,20), (7,27)]
UB: 24.90531259578653  LB: 0.6574946010348435; [(0,3), (1,0), (2,1), (3,5), (4,6), (5,7), (6,14), (7,20)]
UB: 57.1280766957651  LB: 0.657026123215057; [(0,14), (1,7), (2,6), (3,5), (4,1), (5,0), (6,3), (7,9)]
UB: 217.28900870337912  LB: 0.6558353220064969; [(0,1), (1,2), (2,6), (3,7), (4,14), (5,20), (6,27), (7,33)]
UB: 19.30223774501106  LB: 0.6583388440975738; [(0,10), (1,3), (2,0), (3,1), (4,2), (5,6), (6,7), (7,14)]
UB: 203.08194913043937  LB: 0.6587863132854611; [(0,8), (1,9), (2,3), (3,0), (4,1), (5,5), (6,6), (7,7)]
UB: 52.562993098284856  LB: 0.6624454244648206; [(0,15), (1,9), (2,3), (3,0), (4,1), (5,2), (6,6), (7,7)]
>>> Accepting new incumbent with value 0.6624454244648206
UB: 18.966634206735 62  LB: 0.6556432713056434; [(0,8), (1,9), (2,3), (3,0), (4,1), (5,2), (6,6), (7,7)]
UB: 0.6624454244647424  LB: 0.6624454244648209; [(0,15), (1,9), (2,3), (3,0), (4,1), (5,2), (6,6), (7,7)]
>>> Accepting new incumbent with value 0.6624454244648209
UB: 56.655643271306566  LB: 0.6594325845804899; [(0,8), (1,9), (2,3), (3,0), (4,1), (5,2), (6,6), (7,13)]
UB: 40.23049280057592  LB: 0.657347426483223; [(0,4), (1,1), (2,2), (3,6), (4,7), (5,14), (6,20), (7,27)]
UB: 218.35257293477258  LB: 0.6587340491091543; [(0,20), (1,14), (2,7), (3,6), (4,5), (5,1), (6,0), (7,3)]
UB: 71.08145445079963  LB: 0.6581790030401538; [(0,20), (1,14), (2,7), (3,6), (4,2), (5,1), (6,4), (7,3)]
UB: 213.68320025499054  LB: 0.6612315777700716; [(0,33), (1,27), (2,20), (3,14), (4,7), (5,6), (6,2), (7,1)]
UB: 49.88130029619707  LB: 0.6568581560449529; [(0,7), (1,6), (2,5), (3,1), (4,0), (5,3), (6,9), (7,8)]
UB: 78.43804131819874  LB: 0.6563437561934647; [(0,7), (1,6), (2,2), (3,1), (4,0), (5,3), (6,9), (7,15)]
UB: 22.43804131819878  LB: 0.6525842036517943; [(0,7), (1,6), (2,2), (3,1), (4,0), (5,3), (6,9), (7,8)]
UB: 217.10523114901807  LB: 0.6596615524681942; [(0,6), (1,2), (2,1), (3,0), (4,3), (5,9), (6,15), (7,21)]
```

```
UB: 80.89666532283813  LB: 0.6546189357315272; [(0,14), (1,7), (2,6), (3,2), (4,1), (5,0), (6,3), (7,10)]
UB: 24.896665322825502  LB: 0.6531617283187299; [(0,14), (1,7), (2,6), (3,2), (4,1), (5,0), (6,3), (7,9)]
UB: 35.095389377234554  LB: 0.6597283028005759; [(0,27), (1,20), (2,14), (3,7), (4,6), (5,5), (6,1), (7,0)]
UB: 56.656180116430995  LB: 0.6592564241817966; [(0,27), (1,20), (2,14), (3,7), (4,6), (5,2), (6,1), (7,4)]
UB: 29.17922178349122  LB: 0.6551246007142941; [(0,20), (1,14), (2,7), (3,6), (4,2), (5,1), (6,0), (7,3)]
UB: 75.05793907826947  LB: 0.6559351253023317; [(0,0), (1,1), (2,2), (3,6), (4,7), (5,14), (6,20), (7,26)]
UB: 56.655935125302136  LB: 0.6554322906673942; [(0,0), (1,1), (2,2), (3,6), (4,7), (5,14), (6,20), (7,19)]
UB: 19.05793907826947  LB: 0.6519264692732137; [(0,0), (1,1), (2,2), (3,6), (4,7), (5,14), (6,20), (7,27)]
UB: 20.300173982595705  LB: 0.6587739705214973; [(0,14), (1,13), (2,6), (3,2), (4,1), (5,0), (6,3), (7,9)]
UB: 19.02812341466628  LB: 0.6595106744013935; [(0,13), (1,6), (2,2), (3,1), (4,0), (5,3), (6,9), (7,8)]
Nr of benders cuts: 40
Optimal search trajectory: [(0,15), (1,9), (2,3), (3,0), (4,1), (5,2), (6,6), (7,7)], with PD: 0.6624454244648209
```

Appendix C

Output of the branch & price algorithm

```
Instance: K = 14, Nu = 2, T = inf (see Table 6.2)
Queue size: 1;
Red. cost traj.: omega_0_0 [(0,39), (1,44), (2,49), (3,50), (4,51), (5,52), (6,48), (7,43), (8,36), (9,28), (10,20),
     (11,13), (12,8), (13,7)]; c*: 0.0
Red. cost traj.: omega_1_0 [(0,42), (1,48), (2,49), (3,50), (4,51), (5,52), (6,47), (7,41), (8,33), (9,25), (10,18),
     (11,12), (12,8), (13,7)]; c*: 0.0
RMP solved. z: 0.5980374938023966; omega_0_0: 1.0; omega_1_0: 1.0;
Red. cost traj.: omega_0_1 [(0,34), (1,26), (2,19), (3,20), (4,28), (5,35), (6,34), (7,26), (8,19), (9,20), (10,28),
     (11,35), (12,34), (13,26)]; c*: 0.23440980457097224
Red. cost traj.: omega_1_1 [(0,31), (1,23), (2,22), (3,29), (4,37), (5,38), (6,31), (7,23), (8,22), (9,29), (10,37),
     (11,38), (12,31), (13,23)]; c*: 0.3565524099779534
RMP solved. z: 0.9769185224864061; omega_0_1: 1.0; omega_1_1: 1.0;
Red. cost traj.: omega_0_2 [(0,26), (1,34), (2,35), (3,28), (4,20), (5,19), (6,26), (7,34), (8,35), (9,28), (10,20),
     (11,19), (12,12), (13,7)]; c*: 0.2293923534978039
Red. cost traj.: omega_1_2 [(0,31), (1,23), (2,17), (3,18), (4,25), (5,32), (6,31), (7,30), (8,22), (9,16), (10,11),
     (11,12), (12,18), (13,24)]; c*: 0.36354166363192436
RMP solved. z: 0.9769185224864062; omega_0_1: 1.0; omega_1_1: 1.0;
Red. cost traj.: omega_0_3 [(0,34), (1,26), (2,19), (3,20), (4,28), (5,35), (6,34), (7,26), (8,19), (9,20), (10,14),
     (11,9), (12,8), (13,7)]; c*: 0.22712969639614447
Red. cost traj.: omega_1_3 [(0,39), (1,38), (2,30), (3,23), (4,24), (5,32), (6,39), (7,38), (8,30), (9,23), (10,17),
     (11,12), (12,8), (13,7)]; c*: 0.3605580533468167
RMP solved. z: 0.9769185224864062; omega_0_1: 1.0; omega_1_1: 1.0;
Red. cost traj.: omega_0_4 [(0,34), (1,26), (2,19), (3,20), (4,28), (5,35), (6,34), (7,33), (8,25), (9,18), (10,19),
     (11,13), (12,8), (13,7)]; c*: 0.22659284049620823
Red. cost traj.: omega_1_4 [(0,22), (1,23), (2,31), (3,39), (4,45), (5,44), (6,37), (7,30), (8,23), (9,24), (10,18),
     (11,12), (12,11), (13,16)]; c*: 0.3560204925702279
RMP solved. z: 0.9769185224864061; omega_0_1: 1.0; omega_1_1: 1.0;
        .
        .
        .
RMP solved. z: 0.9771083957511488; omega_0_1: 0.7306; omega_0_11: 0.2694; omega_1_1: 0.9772; omega_1_24: 0.0228;
Red. cost traj.: omega_0_16 [(0,26), (1,34), (2,35), (3,28), (4,20), (5,19), (6,26), (7,34), (8,35), (9,28), (10,20),
     (11,13), (12,8), (13,7)]; c*: 0.1508149596739335
Red. cost traj.: omega_1_25 [(0,31), (1,23), (2,16), (3,15), (4,21), (5,29), (6,30), (7,31), (8,32), (9,25), (10,18),
     (11,12), (12,8), (13,7)]; c*: 0.22820363149432893
RMP solved. z: 0.9771083957511488; omega_0_1: 0.7306; omega_0_11: 0.2694; omega_1_1: 0.9772; omega_1_24: 0.0228;
Red. cost traj.: omega_0_17 [(0,39), (1,44), (2,49), (3,50), (4,51), (5,52), (6,48), (7,43), (8,36), (9,28), (10,20),
     (11,13), (12,8), (13,7)]; c*: 0.1508149596739335
Red. cost traj.: omega_1_26 [(0,37), (1,43), (2,49), (3,50), (4,45), (5,39), (6,31), (7,30), (8,22), (9,16), (10,11),
     (11,12), (12,18), (13,24)]; c*: 0.22820363149432893
RMP solved. z: 0.9771083957511488; omega_0_1: 0.7306; omega_0_11: 0.2694; omega_1_1: 0.9772; omega_1_24: 0.0228;
```

```
k: -1; UB: 0.9771083957511488; LB: 0.0; Branch
Queue size: 4;
Red. cost traj.: omega_0_18 [(0,41), (1,40), (2,45), (3,50), (4,51), (5,52), (6,48), (7,43), (8,36), (9,28), (10,20),
      (11,13), (12,8), (13,7)]; c*: 0.0
Red. cost traj.: omega_1_27 [(0,42), (1,48), (2,49), (3,50), (4,51), (5,52), (6,47), (7,41), (8,33), (9,25), (10,18),
      (11,12), (12,8), (13,7)]; c*: 0.0
RMP solved. z: 0.9771068025084846; omega_0_1: 0.6735; omega_0_11: 0.3265; omega_1_1: 1.0;
Red. cost traj.: omega_0_19 [(0,34), (1,40), (2,45), (3,50), (4,51), (5,52), (6,48), (7,43), (8,36), (9,28), (10,20),
      (11,13), (12,8), (13,7)]; c*: 0.15095166545226926
Red. cost traj.: omega_1_28 [(0,31), (1,23), (2,16), (3,15), (4,21), (5,29), (6,30), (7,23), (8,24), (9,32), (10,39),
      (11,38), (12,30), (13,23)]; c*: 0.22847311364866266
RMP solved. z: 0.9771068025084846; omega_0_1: 0.6735; omega_0_11: 0.3265; omega_1_1: 1.0;
Red. cost traj.: omega_1_29 [(0,42), (1,48), (2,49), (3,50), (4,51), (5,52), (6,47), (7,41), (8,33), (9,25), (10,18),
      (11,17), (12,23), (13,30)]; c*: 0.22847311364866266
RMP solved. z: 0.9771068025084847; omega_0_1: 0.6735; omega_0_11: 0.3265; omega_1_1: 1.0;
Red. cost traj.: omega_1_30 [(0,31), (1,38), (2,44), (3,50), (4,51), (5,52), (6,47), (7,41), (8,33), (9,25), (10,18),
      (11,12), (12,8), (13,7)]; c*: 0.22847311364866263
RMP solved. z: 0.9771068025084847; omega_0_1: 0.6735; omega_0_11: 0.3265; omega_1_1: 1.0;
Red. cost traj.: omega_1_31 [(0,42), (1,48), (2,49), (3,50), (4,45), (5,39), (6,31), (7,30), (8,22), (9,16), (10,17),
      (11,12), (12,8), (13,7)]; c*: 0.22847311364866263
RMP solved. z: 0.9771068025084847; omega_0_1: 0.6735; omega_0_11: 0.3265; omega_1_1: 1.0;
Red. cost traj.: omega_1_32 [(0,31), (1,38), (2,44), (3,50), (4,51), (5,52), (6,47), (7,41), (8,33), (9,25), (10,18),
      (11,17), (12,23), (13,30)]; c*: 0.22847311364866263
RMP solved. z: 0.9771068025084847; omega_0_1: 0.6735; omega_0_11: 0.3265; omega_1_1: 1.0;
Red. cost traj.: omega_1_33 [(0,31), (1,23), (2,17), (3,18), (4,25), (5,32), (6,31), (7,30), (8,22), (9,16), (10,17),
      (11,12), (12,8), (13,7)]; c*: 0.2284731136486626
RMP solved. z: 0.9771068025084846; omega_0_1: 0.6735; omega_0_11: 0.3265; omega_1_1: 1.0;
Red. cost traj.: omega_1_34 [(0,31), (1,38), (2,45), (3,50), (4,49), (5,43), (6,37), (7,30), (8,23), (9,24), (10,18),
      (11,12), (12,8), (13,7)]; c*: 0.22838536509372281
RMP solved. z: 0.9771068025084846; omega_0_1: 0.6735; omega_0_11: 0.3265; omega_1_1: 1.0;
k: 0; UB: 0.9771068025084846; LB: 0.0; Branch
Queue size: 7;
Red. cost traj.: omega_0_20 [(0,51), (1,50), (2,45), (3,40), (4,41), (5,47), (6,48), (7,43), (8,36), (9,28), (10,20),
      (11,13), (12,8), (13,7)]; c*: 0.0
Red. cost traj.: omega_1_35 [(0,42), (1,48), (2,49), (3,50), (4,51), (5,52), (6,47), (7,41), (8,33), (9,25), (10,18),
      (11,12), (12,8), (13,7)]; c*: 0.0
RMP solved. z: 0.9769185224864063; omega_0_1: 1.0; omega_1_1: 1.0;
Red. cost traj.: omega_0_21 [(0,34), (1,26), (2,19), (3,20), (4,28), (5,35), (6,34), (7,33), (8,25), (9,18), (10,19),
      (11,13), (12,8), (13,7)]; c*: 0.15236332140223863
Red. cost traj.: omega_1_36 [(0,31), (1,23), (2,17), (3,18), (4,25), (5,32), (6,31), (7,30), (8,22), (9,16), (10,17),
      (11,12), (12,8), (13,7)]; c*: 0.23177675836999073
RMP solved. z: 0.9769185224864061; omega_0_1: 1.0; omega_1_1: 1.0;
k: 1; UB: 0.9769185224864061; LB: 0.0; >>> New best strategy found
Queue size: 6;
Red. cost traj.: omega_0_22 [(0,41), (1,40), (2,45), (3,50), (4,51), (5,52), (6,48), (7,43), (8,36), (9,28), (10,20),
      (11,13), (12,8), (13,7)]; c*: 0.0
Red. cost traj.: omega_1_37 [(0,42), (1,48), (2,49), (3,50), (4,51), (5,52), (6,47), (7,41), (8,33), (9,25), (10,18),
      (11,12), (12,8), (13,7)]; c*: 0.0
RMP solved. z: 0.9758224483397067; omega_0_11: 1.0; omega_1_1: 1.0;
Red. cost traj.: omega_0_23 [(0,34), (1,33), (2,25), (3,18), (4,19), (5,27), (6,34), (7,33), (8,25), (9,18), (10,19),
      (11,13), (12,8), (13,7)]; c*: 0.14697463705587602
Red. cost traj.: omega_1_38 [(0,31), (1,23), (2,22), (3,29), (4,37), (5,38), (6,39), (7,40), (8,33), (9,25), (10,24),
      (11,23), (12,30), (13,38)]; c*: 0.22569111845909334
RMP solved. z: 0.9758224483397065; omega_0_11: 1.0; omega_1_1: 1.0;
Red. cost traj.: omega_0_24 [(0,41), (1,34), (2,26), (3,19), (4,20), (5,28), (6,35), (7,34), (8,26), (9,19), (10,20),
      (11,28), (12,35), (13,34)]; c*: 0.14724124978537376
Red. cost traj.: omega_1_39 [(0,42), (1,48), (2,49), (3,50), (4,45), (5,39), (6,31), (7,30), (8,22), (9,16), (10,17),
      (11,24), (12,31), (13,30)]; c*: 0.22391741266028053
RMP solved. z: 0.9758224483397065; omega_0_11: 1.0; omega_1_1: 1.0;
Red. cost traj.: omega_0_25 [(0,34), (1,40), (2,45), (3,50), (4,51), (5,52), (6,48), (7,43), (8,36), (9,28), (10,27),
      (11,19), (12,12), (13,7)]; c*: 0.14641000628808815
```

```
Red. cost traj.: omega_1_40 [(0,31), (1,23), (2,22), (3,29), (4,37), (5,38), (6,39), (7,40), (8,33), (9,25), (10,18),
      (11,17), (12,23), (13,30)]; c*: 0.22259591204021714
RMP solved. z: 0.9758224483397065; omega_0_11: 1.0; omega_1_1: 1.0;
Red. cost traj.: omega_0_26 [(0,34), (1,33), (2,25), (3,18), (4,19), (5,27), (6,34), (7,33), (8,25), (9,18), (10,19),
      (11,13), (12,8), (13,7)]; c*: 0.14544041956104137
Red. cost traj.: omega_1_41 [(0,42), (1,48), (2,49), (3,50), (4,51), (5,52), (6,47), (7,41), (8,33), (9,25), (10,18),
      (11,12), (12,8), (13,7)]; c*: 0.22224841225938555
RMP solved. z: 0.9758224483397065; omega_0_11: 1.0; omega_1_1: 1.0;
k: 1; UB: 0.9758224483397065; LB: 0.9769185224864061; Prune
Queue size: 5;
Red. cost traj.: omega_0_27 [(0,41), (1,40), (2,45), (3,50), (4,51), (5,52), (6,48), (7,43), (8,36), (9,28), (10,20),
      (11,13), (12,8), (13,7)]; c*: 0.0
Red. cost traj.: omega_1_42 [(0,36), (1,43), (2,49), (3,50), (4,51), (5,52), (6,47), (7,41), (8,33), (9,25), (10,18),
      (11,12), (12,8), (13,7)]; c*: 0.0
RMP solved. z: 0.9749896144997193; omega_0_1: 0.7453; omega_0_11: 0.2547; omega_1_24: 1.0;
Red. cost traj.: omega_0_28 [(0,34), (1,26), (2,19), (3,20), (4,28), (5,35), (6,34), (7,26), (8,18), (9,12), (10,13),
      (11,20), (12,27), (13,26)]; c*: 0.1585449899443072
Red. cost traj.: omega_1_43 [(0,22), (1,23), (2,31), (3,38), (4,45), (5,46), (6,47), (7,41), (8,33), (9,25), (10,24),
      (11,23), (12,30), (13,38)]; c*: 0.2565063120771721
RMP solved. z: 0.9749898339647257; omega_0_1: 0.7141; omega_0_11: 0.2655; omega_0_28: 0.0204; omega_1_24: 1.0;
Red. cost traj.: omega_0_29 [(0,34), (1,40), (2,45), (3,50), (4,51), (5,52), (6,48), (7,43), (8,36), (9,28), (10,20),
      (11,13), (12,8), (13,7)]; c*: 0.15861638819631063
Red. cost traj.: omega_1_44 [(0,30), (1,38), (2,45), (3,50), (4,49), (5,43), (6,37), (7,30), (8,23), (9,24), (10,18),
      (11,12), (12,11), (13,7)]; c*: 0.24824881340765903
RMP solved. z: 0.9749898339647257; omega_0_1: 0.7141; omega_0_11: 0.2655; omega_0_28: 0.0204; omega_1_24: 1.0;
Red. cost traj.: omega_1_45 [(0,30), (1,38), (2,45), (3,50), (4,49), (5,43), (6,37), (7,30), (8,23), (9,24), (10,18),
      (11,12), (12,11), (13,16)]; c*: 0.24731328276044515
RMP solved. z: 0.9749898339647255; omega_0_1: 0.7141; omega_0_11: 0.2655; omega_0_28: 0.0204; omega_1_24: 1.0;
Red. cost traj.: omega_0_30 [(0,34), (1,40), (2,45), (3,50), (4,51), (5,52), (6,48), (7,43), (8,36), (9,28), (10,20),
      (11,19), (12,12), (13,7)]; c*: 0.1586163881963107
Red. cost traj.: omega_1_46 [(0,24), (1,17), (2,16), (3,22), (4,30), (5,31), (6,39), (7,40), (8,33), (9,25), (10,24),
      (11,23), (12,30), (13,38)]; c*: 0.24658587404429338
RMP solved. z: 0.9749898339647254; omega_0_1: 0.7141; omega_0_11: 0.2655; omega_0_28: 0.0204; omega_1_24: 1.0;
Red. cost traj.: omega_1_47 [(0,30), (1,38), (2,44), (3,50), (4,51), (5,52), (6,47), (7,41), (8,33), (9,25), (10,24),
      (11,23), (12,30), (13,38)]; c*: 0.24100166183846125
RMP solved. z: 0.9749898339647253; omega_0_1: 0.7141; omega_0_11: 0.2655; omega_0_28: 0.0204; omega_1_24: 1.0;
Red. cost traj.: omega_1_48 [(0,37), (1,43), (2,49), (3,50), (4,45), (5,38), (6,31), (7,23), (8,22), (9,29), (10,37),
      (11,38), (12,31), (13,23)]; c*: 0.23985170466188666
RMP solved. z: 0.9749898339647254; omega_0_1: 0.7141; omega_0_11: 0.2655; omega_0_28: 0.0204; omega_1_24: 1.0;
Red. cost traj.: omega_0_31 [(0,34), (1,40), (2,45), (3,50), (4,51), (5,52), (6,48), (7,43), (8,36), (9,28), (10,20),
      (11,13), (12,8), (13,7)]; c*: 0.15861638819631063
Red. cost traj.: omega_1_49 [(0,24), (1,23), (2,30), (3,38), (4,39), (5,46), (6,47), (7,41), (8,33), (9,25), (10,18),
      (11,17), (12,23), (13,31)]; c*: 0.23776248646461542
RMP solved. z: 0.9749898339647258; omega_0_1: 0.7141; omega_0_11: 0.2655; omega_0_28: 0.0204; omega_1_24: 1.0;
Red. cost traj.: omega_1_50 [(0,30), (1,31), (2,39), (3,45), (4,51), (5,52), (6,47), (7,41), (8,33), (9,25), (10,18),
      (11,17), (12,23), (13,30)]; c*: 0.23852511265276796
RMP solved. z: 0.9749898339647256; omega_0_1: 0.7141; omega_0_11: 0.2655; omega_0_28: 0.0204; omega_1_24: 1.0;
Red. cost traj.: omega_1_51 [(0,39), (1,32), (2,24), (3,23), (4,30), (5,38), (6,39), (7,40), (8,33), (9,25), (10,24),
      (11,23), (12,30), (13,38)]; c*: 0.2389450697062649
RMP solved. z: 0.9749898339647256; omega_0_1: 0.7141; omega_0_11: 0.2655; omega_0_28: 0.0204; omega_1_24: 1.0;
Red. cost traj.: omega_1_52 [(0,24), (1,23), (2,30), (3,38), (4,39), (5,46), (6,47), (7,41), (8,33), (9,25), (10,24),
      (11,23), (12,30), (13,38)]; c*: 0.23859183350139568
RMP solved. z: 0.9749898339647256; omega_0_1: 0.7141; omega_0_11: 0.2655; omega_0_28: 0.0204; omega_1_24: 1.0;
Red. cost traj.: omega_1_53 [(0,37), (1,43), (2,49), (3,50), (4,51), (5,52), (6,47), (7,41), (8,33), (9,25), (10,18),
      (11,17), (12,23), (13,30)]; c*: 0.23812317469311584
RMP solved. z: 0.9749898339647256; omega_0_1: 0.7141; omega_0_11: 0.2655; omega_0_28: 0.0204; omega_1_24: 1.0;
k: 0; UB: 0.9749898339647256; LB: 0.9769185224864061; Prune
Queue size: 4;
Red. cost traj.: omega_0_32 [(0,38), (1,44), (2,49), (3,50), (4,51), (5,52), (6,48), (7,43), (8,36), (9,28), (10,20),
      (11,13), (12,8), (13,7)]; c*: 0.0
```

```
Red. cost traj.: omega_1_54 [(0,42), (1,48), (2,49), (3,50), (4,51), (5,52), (6,47), (7,41), (8,33), (9,25), (10,13),
      (11,12), (12,8), (13,7)]; c*: 0.0
RMP solved. z: 0.9654836230860869; omega_0_2: 1.0; omega_1_1: 1.0;
Red. cost traj.: omega_0_33 [(0,40), (1,33), (2,25), (3,18), (4,19), (5,27), (6,34), (7,33), (8,25), (9,18), (10,13),
      (11,27), (12,34), (13,33)]; c*: 0.21691011821102257
Red. cost traj.: omega_1_55 [(0,42), (1,48), (2,49), (3,50), (4,45), (5,39), (6,31), (7,23), (8,22), (9,29), (10,37),
      (11,38), (12,31), (13,23)]; c*: 0.30230300248608943
RMP solved. z: 0.9654836230860869; omega_0_2: 1.0; omega_1_1: 1.0;
Red. cost traj.: omega_0_34 [(0,27), (1,19), (2,18), (3,25), (4,33), (5,34), (6,27), (7,19), (8,18), (9,25), (10,33),
      (11,34), (12,27), (13,19)]; c*: 0.21369172283368656
Red. cost traj.: omega_1_56 [(0,42), (1,48), (2,49), (3,50), (4,45), (5,39), (6,31), (7,30), (8,22), (9,16), (10,17),
      (11,24), (12,31), (13,30)]; c*: 0.3023030024860895
RMP solved. z: 0.9749967485333461; omega_0_34: 1.0; omega_1_1: 1.0;
Red. cost traj.: omega_0_35 [(0,33), (1,25), (2,18), (3,19), (4,27), (5,35), (6,41), (7,40), (8,33), (9,26), (10,27),
      (11,20), (12,13), (13,12)]; c*: 0.15573289373629637
Red. cost traj.: omega_1_57 [(0,42), (1,48), (2,49), (3,50), (4,51), (5,52), (6,47), (7,41), (8,33), (9,25), (10,18),
      (11,17), (12,11), (13,7)]; c*: 0.2155179789120364
RMP solved. z: 0.9749967485333458; omega_0_34: 1.0; omega_1_1: 1.0;
Red. cost traj.: omega_0_36 [(0,26), (1,34), (2,35), (3,28), (4,20), (5,19), (6,26), (7,34), (8,35), (9,28), (10,20),
      (11,19), (12,26), (13,33)]; c*: 0.15671412664767623
Red. cost traj.: omega_1_58 [(0,31), (1,38), (2,44), (3,50), (4,51), (5,52), (6,47), (7,41), (8,33), (9,25), (10,18),
      (11,17), (12,23), (13,31)]; c*: 0.21437347586823127
        .
        .
        .
RMP solved. z: 0.9753075958153836; omega_0_34: 0.5645; omega_0_52: 0.4355; omega_1_1: 1.0;
Red. cost traj.: omega_0_61 [(0,27), (1,26), (2,33), (3,40), (4,41), (5,35), (6,27), (7,26), (8,33), (9,40), (10,41),
      (11,35), (12,27), (13,19)]; c*: 0.1570442087380076
RMP solved. z: 0.9753075958153836; omega_0_34: 0.5645; omega_0_52: 0.4355; omega_1_1: 1.0;
Red. cost traj.: omega_0_62 [(0,39), (1,44), (2,49), (3,50), (4,51), (5,52), (6,48), (7,43), (8,36), (9,35), (10,27),
      (11,19), (12,13), (13,14)]; c*: 0.15692998595924446
RMP solved. z: 0.9753075958153836; omega_0_34: 0.5645; omega_0_52: 0.4355; omega_1_1: 1.0;
k: 0; UB: 0.9753075958153836; LB: 0.9769185224864061; Prune
Queue size: 3;
Red. cost traj.: omega_0_63 [(0,51), (1,50), (2,45), (3,40), (4,41), (5,47), (6,48), (7,43), (8,36), (9,28), (10,20),
      (11,13), (12,8), (13,7)]; c*: 0.0
Red. cost traj.: omega_1_62 [(0,42), (1,43), (2,49), (3,50), (4,51), (5,52), (6,47), (7,41), (8,33), (9,25), (10,18),
      (11,12), (12,8), (13,7)]; c*: 0.0
RMP solved. z: 0.8674830677852969; omega_0_1: 1.0; omega_1_59: 1.0;
Red. cost traj.: omega_0_64 [(0,34), (1,26), (2,19), (3,20), (4,28), (5,35), (6,34), (7,26), (8,19), (9,20), (10,28),
      (11,35), (12,34), (13,33)]; c*: 0.2391390463013009
Red. cost traj.: omega_1_63 [(0,44), (1,38), (2,39), (3,32), (4,24), (5,23), (6,30), (7,38), (8,39), (9,32), (10,24),
      (11,23), (12,30), (13,38)]; c*: 0.4806085169476201
RMP solved. z: 0.9293494738842998; omega_0_1: 1.0; omega_1_63: 1.0;
Red. cost traj.: omega_0_65 [(0,34), (1,26), (2,19), (3,20), (4,28), (5,35), (6,41), (7,40), (8,33), (9,26), (10,19),
      (11,13), (12,8), (13,7)]; c*: 0.23138219612865596
Red. cost traj.: omega_1_64 [(0,42), (1,43), (2,44), (3,45), (4,39), (5,32), (6,24), (7,23), (8,30), (9,38), (10,39),
      (11,32), (12,24), (13,23)]; c*: 0.46319389091931035
RMP solved. z: 0.9293494738842998; omega_0_1: 1.0; omega_1_63: 1.0;
Red. cost traj.: omega_0_66 [(0,34), (1,26), (2,19), (3,20), (4,28), (5,35), (6,34), (7,33), (8,25), (9,18), (10,19),
      (11,27), (12,34), (13,33)]; c*: 0.23002533665742825
Red. cost traj.: omega_1_65 [(0,31), (1,38), (2,37), (3,29), (4,22), (5,23), (6,24), (7,25), (8,33), (9,40), (10,39),
      (11,31), (12,23), (13,16)]; c*: 0.37251796938721987
RMP solved. z: 0.9293494738842999; omega_0_1: 1.0; omega_1_63: 1.0;
Red. cost traj.: omega_1_66 [(0,31), (1,30), (2,37), (3,44), (4,45), (5,39), (6,31), (7,30), (8,22), (9,16), (10,17),
      (11,24), (12,31), (13,30)]; c*: 0.389553444505835
RMP solved. z: 0.9441708064549769; omega_0_1: 1.0; omega_1_66: 1.0;
Red. cost traj.: omega_0_67 [(0,34), (1,26), (2,19), (3,20), (4,28), (5,35), (6,41), (7,40), (8,33), (9,26), (10,19),
      (11,13), (12,8), (13,7)]; c*: 0.21742021326611338
Red. cost traj.: omega_1_67 [(0,44), (1,38), (2,31), (3,23), (4,22), (5,29), (6,37), (7,38), (8,31), (9,23), (10,17),
      (11,12), (12,8), (13,7)]; c*: 0.33526279289162786
```

RMP solved. z: 0.9441708064549766; omega_0_1: 1.0; omega_1_66: 1.0;
Red. cost traj.: omega_1_68 [(0,31), (1,24), (2,17), (3,16), (4,22), (5,30), (6,31), (7,24), (8,18), (9,12), (10,11), (11,16), (12,23), (13,31)]; c*: 0.3231104372842089
RMP solved. z: 0.9441708064549766; omega_0_1: 1.0; omega_1_66: 1.0;
Red. cost traj.: omega_1_69 [(0,44), (1,37), (2,30), (3,31), (4,39), (5,45), (6,44), (7,37), (8,30), (9,31), (10,24), (11,17), (12,11), (13,7)]; c*: 0.31913262499874234
RMP solved. z: 0.9441708064549766; omega_0_1: 1.0; omega_1_66: 1.0;
Red. cost traj.: omega_1_70 [(0,31), (1,38), (2,37), (3,29), (4,22), (5,23), (6,24), (7,32), (8,39), (9,38), (10,30), (11,23), (12,24), (13,32)]; c*: 0.32274084894508415
RMP solved. z: 0.9441708064549768; omega_0_1: 1.0; omega_1_66: 1.0;
Red. cost traj.: omega_1_71 [(0,31), (1,30), (2,22), (3,16), (4,17), (5,24), (6,31), (7,30), (8,22), (9,16), (10,11), (11,12), (12,18), (13,24)]; c*: 0.31777002645140884
RMP solved. z: 0.9441708064549768; omega_0_1: 1.0; omega_1_66: 1.0;
Red. cost traj.: omega_1_72 [(0,44), (1,38), (2,30), (3,23), (4,24), (5,32), (6,39), (7,38), (8,30), (9,22), (10,16), (11,17), (12,24), (13,31)]; c*: 0.3307742551221864
RMP solved. z: 0.9441708064549766; omega_0_1: 1.0; omega_1_66: 1.0;
Red. cost traj.: omega_1_73 [(0,31), (1,24), (2,17), (3,16), (4,22), (5,30), (6,31), (7,24), (8,17), (9,16), (10,22), (11,30), (12,31), (13,24)]; c*: 0.32793751679528516

.
.
.

RMP solved. z: 0.9603514239521236; omega_0_1: 1.0; omega_1_76: 1.0;
Red. cost traj.: omega_1_99 [(0,44), (1,45), (2,39), (3,31), (4,23), (5,22), (6,29), (7,37), (8,38), (9,31), (10,23), (11,16), (12,11), (13,7)]; c*: 0.2915779050579745
RMP solved. z: 0.9603514239521236; omega_0_1: 1.0; omega_1_76: 1.0;
Red. cost traj.: omega_1_100 [(0,44), (1,38), (2,39), (3,32), (4,24), (5,23), (6,30), (7,29), (8,21), (9,15), (10,16), (11,23), (12,30), (13,38)]; c*: 0.2915146389096226
RMP solved. z: 0.9603514239521236; omega_0_1: 1.0; omega_1_76: 1.0;
Red. cost traj.: omega_1_101 [(0,31), (1,32), (2,25), (3,18), (4,17), (5,23), (6,30), (7,29), (8,21), (9,15), (10,16), (11,23), (12,30), (13,38)]; c*: 0.2912445275501204
RMP solved. z: 0.9603514239521236; omega_0_1: 1.0; omega_1_76: 1.0;
Red. cost traj.: omega_1_102 [(0,31), (1,38), (2,45), (3,50), (4,49), (5,43), (6,37), (7,30), (8,23), (9,24), (10,18), (11,12), (12,8), (13,7)]; c*: 0.29056489251653406
RMP solved. z: 0.9603514239521236; omega_0_1: 1.0; omega_1_76: 1.0;
k: 1; UB: 0.9603514239521236; LB: 0.9769185224864061; Prune
Queue size: 2;
Red. cost traj.: omega_0_69 [(0,41), (1,40), (2,45), (3,50), (4,51), (5,52), (6,48), (7,43), (8,36), (9,28), (10,20), (11,13), (12,8), (13,7)]; c*: 0.0
Red. cost traj.: omega_1_103 [(0,42), (1,43), (2,49), (3,50), (4,51), (5,52), (6,47), (7,41), (8,33), (9,25), (10,18), (11,12), (12,8), (13,7)]; c*: 0.0
RMP solved. z: 0.958351824806755; omega_0_11: 1.0; omega_1_76: 1.0;
Red. cost traj.: omega_0_70 [(0,34), (1,40), (2,39), (3,32), (4,25), (5,26), (6,27), (7,28), (8,36), (9,42), (10,41), (11,34), (12,27), (13,19)]; c*: 0.2170531518985872
Red. cost traj.: omega_1_104 [(0,44), (1,45), (2,39), (3,31), (4,30), (5,22), (6,16), (7,17), (8,18), (9,25), (10,32), (11,31), (12,23), (13,16)]; c*: 0.2886105340278401
RMP solved. z: 0.9583518248067545; omega_0_11: 1.0; omega_1_76: 1.0;
Red. cost traj.: omega_0_71 [(0,41), (1,40), (2,33), (3,26), (4,27), (5,35), (6,41), (7,40), (8,33), (9,26), (10,27), (11,20), (12,13), (13,12)]; c*: 0.2111849977761295
Red. cost traj.: omega_1_105 [(0,31), (1,38), (2,37), (3,29), (4,22), (5,23), (6,31), (7,39), (8,45), (9,44), (10,37), (11,30), (12,22), (13,16)]; c*: 0.2881750691425751
RMP solved. z: 0.9583518248067545; omega_0_11: 1.0; omega_1_76: 1.0;
Red. cost traj.: omega_0_72 [(0,34), (1,40), (2,45), (3,50), (4,51), (5,52), (6,48), (7,43), (8,36), (9,35), (10,34), (11,26), (12,19), (13,20)]; c*: 0.21023095199796332
Red. cost traj.: omega_1_106 [(0,31), (1,38), (2,45), (3,50), (4,49), (5,43), (6,37), (7,30), (8,23), (9,24), (10,18), (11,12), (12,8), (13,7)]; c*: 0.28801519694752276
RMP solved. z: 0.9583518248067545; omega_0_11: 1.0; omega_1_76: 1.0;
Red. cost traj.: omega_0_73 [(0,34), (1,40), (2,45), (3,50), (4,51), (5,52), (6,48), (7,43), (8,36), (9,28), (10,20), (11,19), (12,26), (13,33)]; c*: 0.20734481865378823
RMP solved. z: 0.9583518248067545; omega_0_11: 1.0; omega_1_76: 1.0;
Red. cost traj.: omega_0_74 [(0,34), (1,40), (2,45), (3,50), (4,51), (5,52), (6,48), (7,43), (8,36), (9,28), (10,27), (11,26), (12,18), (13,12)]; c*: 0.20926341692590988

```
RMP solved. z: 0.9583518248067545; omega_0_11: 1.0; omega_1_76: 1.0;
Red. cost traj.: omega_0_75 [(0,34), (1,33), (2,25), (3,18), (4,19), (5,27), (6,34), (7,33), (8,25), (9,18), (10,19),
      (11,27), (12,35), (13,41)]; c*: 0.21148929369390368
RMP solved. z: 0.9583518248067545; omega_0_11: 1.0; omega_1_76: 1.0;
Red. cost traj.: omega_0_76 [(0,34), (1,35), (2,28), (3,20), (4,19), (5,26), (6,33), (7,40), (8,41), (9,35), (10,27),
      (11,26), (12,18), (13,12)]; c*: 0.20908765737830917
RMP solved. z: 0.9583518248067545; omega_0_11: 1.0; omega_1_76: 1.0;
Red. cost traj.: omega_0_77 [(0,34), (1,33), (2,25), (3,18), (4,19), (5,27), (6,34), (7,33), (8,25), (9,18), (10,19),
      (11,27), (12,34), (13,33)]; c*: 0.20714930565181988
RMP solved. z: 0.9583518248067545; omega_0_11: 1.0; omega_1_76: 1.0;
Red. cost traj.: omega_0_78 [(0,34), (1,40), (2,45), (3,50), (4,51), (5,52), (6,48), (7,43), (8,36), (9,35), (10,27),
      (11,26), (12,18), (13,12)]; c*: 0.2055649561006503
RMP solved. z: 0.9583518248067545; omega_0_11: 1.0; omega_1_76: 1.0;
Red. cost traj.: omega_0_79 [(0,34), (1,35), (2,28), (3,20), (4,19), (5,26), (6,34), (7,35), (8,36), (9,29), (10,2.),
      (11,20), (12,27), (13,34)]; c*: 0.20488384014018296
RMP solved. z: 0.9583518248067545; omega_0_11: 1.0; omega_1_76: 1.0;
Red. cost traj.: omega_0_80 [(0,34), (1,33), (2,25), (3,18), (4,19), (5,27), (6,34), (7,33), (8,25), (9,18), (10,19),
      (11,27), (12,34), (13,41)]; c*: 0.20006713849371605
RMP solved. z: 0.9583518248067545; omega_0_11: 1.0; omega_1_76: 1.0;
Red. cost traj.: omega_0_81 [(0,41), (1,40), (2,33), (3,26), (4,27), (5,35), (6,41), (7,40), (8,33), (9,26), (10,19),
      (11,13), (12,8), (13,7)]; c*: 0.19940788922777392
RMP solved. z: 0.9583518248067545; omega_0_11: 1.0; omega_1_76: 1.0;
Red. cost traj.: omega_0_82 [(0,41), (1,35), (2,27), (3,26), (4,33), (5,40), (6,41), (7,42), (8,36), (9,28), (10,27),
      (11,26), (12,18), (13,12)]; c*: 0.19919679961479583
RMP solved. z: 0.9583518248067545; omega_0_11: 1.0; omega_1_76: 1.0;
Red. cost traj.: omega_0_83 [(0,34), (1,33), (2,25), (3,18), (4,19), (5,27), (6,34), (7,40), (8,39), (9,32), (10,25),
      (11,26), (12,34), (13,41)]; c*: 0.19934999366020104
RMP solved. z: 0.9583518248067545; omega_0_11: 1.0; omega_1_76: 1.0;
Red. cost traj.: omega_0_84 [(0,41), (1,40), (2,33), (3,26), (4,27), (5,35), (6,41), (7,40), (8,33), (9,26), (10,19),
      (11,13), (12,8), (13,7)]; c*: 0.19919679961479586
RMP solved. z: 0.9583518248067545; omega_0_11: 1.0; omega_1_76: 1.0;
k: 1; UB: 0.9583518248067545; LB: 0.9769185224864061; Prune
Queue size: 1;
Red. cost traj.: omega_0_85 [(0,38), (1,44), (2,49), (3,50), (4,51), (5,52), (6,48), (7,43), (8,36), (9,28), (10,20),
      (11,13), (12,8), (13,7)]; c*: 0.0
Red. cost traj.: omega_1_107 [(0,36), (1,43), (2,49), (3,50), (4,51), (5,52), (6,47), (7,41), (8,33), (9,25), (10,18),
      (11,12), (12,8), (13,7)]; c*: 0.0
RMP solved. z: 0.9739966679206231; omega_0_34: 1.0; omega_1_24: 1.0;
Red. cost traj.: omega_0_86 [(0,27), (1,35), (2,41), (3,40), (4,33), (5,26), (6,27), (7,28), (8,36), (9,42), (10,41),
      (11,34), (12,27), (13,19)]; c*: 0.15910964166492356
Red. cost traj.: omega_1_108 [(0,37), (1,30), (2,31), (3,32), (4,25), (5,18), (6,17), (7,23), (8,30), (9,38), (10,39),
      (11,32), (12,24), (13,23)]; c*: 0.2987079112529644
RMP solved. z: 0.9739966679206234; omega_0_34: 1.0; omega_1_24: 1.0;
Red. cost traj.: omega_0_87 [(0,33), (1,40), (2,41), (3,35), (4,27), (5,26), (6,33), (7,40), (8,41), (9,35), (10,27),
      (11,26), (12,18), (13,12)]; c*: 0.16366668515574934
Red. cost traj.: omega_1_109 [(0,22), (1,23), (2,31), (3,38), (4,37), (5,29), (6,22), (7,23), (8,31), (9,38), (10,37),
      (11,29), (12,22), (13,16)]; c*: 0.26165614690082284
RMP solved. z: 0.9739966679206233; omega_0_34: 1.0; omega_1_24: 1.0;
Red. cost traj.: omega_0_88 [(0,27), (1,26), (2,25), (3,32), (4,39), (5,40), (6,34), (7,26), (8,19), (9,20), (10,23),
      (11,35), (12,34), (13,26)]; c*: 0.15717567742637598
Red. cost traj.: omega_1_110 [(0,36), (1,43), (2,44), (3,38), (4,30), (5,23), (6,17), (7,11), (8,10), (9,15), (10,22),
      (11,30), (12,31), (13,24)]; c*: 0.27117849089434864
RMP solved. z: 0.9739966679206234; omega_0_34: 1.0; omega_1_24: 1.0;
Red. cost traj.: omega_0_89 [(0,25), (1,33), (2,34), (3,27), (4,19), (5,18), (6,25), (7,33), (8,34), (9,27), (10,20),
      (11,13), (12,12), (13,7)]; c*: 0.15537149049873317
Red. cost traj.: omega_1_111 [(0,36), (1,43), (2,44), (3,38), (4,30), (5,23), (6,24), (7,25), (8,33), (9,40), (10,39),
      (11,31), (12,23), (13,16)]; c*: 0.25873174719643893
        .
        .
        .
RMP solved. z: 0.9740490113478565; omega_0_34: 0.8548; omega_0_94: 0.1452; omega_1_24: 1.0;
```

```
Red. cost traj.: omega_1_212 [(0,16), (1,15), (2,21), (3,29), (4,30), (5,23), (6,17), (7,11), (8,10), (9,15), (10,22),
      (11,23), (12,24), (13,32)]; c*: 0.23784846328474196
RMP solved. z: 0.9740490113478565; omega_0_34: 0.8548; omega_0_94: 0.1452; omega_1_24: 1.0;
Red. cost traj.: omega_1_213 [(0,16), (1,15), (2,21), (3,29), (4,30), (5,31), (6,39), (7,40), (8,33), (9,25), (10,24),
      (11,17), (12,11), (13,7)]; c*: 0.23784669033022643
RMP solved. z: 0.9740490113478565; omega_0_34: 0.8548; omega_0_94: 0.1452; omega_1_24: 1.0;
Red. cost traj.: omega_1_214 [(0,39), (1,45), (2,44), (3,37), (4,30), (5,31), (6,32), (7,25), (8,18), (9,17), (10,23),
      (11,31), (12,38), (13,37)]; c*: 0.23785657565646307
RMP solved. z: 0.9740490113478565; omega_0_34: 0.8548; omega_0_94: 0.1452; omega_1_24: 1.0;
Red. cost traj.: omega_1_215 [(0,30), (1,22), (2,16), (3,17), (4,24), (5,31), (6,39), (7,40), (8,33), (9,25), (10,18),
      (11,17), (12,23), (13,22)]; c*: 0.23784399427530598
RMP solved. z: 0.9740490113478565; omega_0_34: 0.8548; omega_0_94: 0.1452; omega_1_24: 1.0;
Red. cost traj.: omega_1_216 [(0,30), (1,38), (2,44), (3,50), (4,51), (5,52), (6,47), (7,41), (8,33), (9,25), (10,18),
      (11,17), (12,23), (13,31)]; c*: 0.23784444860536405
RMP solved. z: 0.9740490113478565; omega_0_34: 0.8548; omega_0_94: 0.1452; omega_1_24: 1.0;
Red. cost traj.: omega_1_217 [(0,37), (1,43), (2,49), (3,50), (4,51), (5,52), (6,47), (7,41), (8,33), (9,25), (10,18),
      (11,17), (12,23), (13,31)]; c*: 0.2378467197296325
RMP solved. z: 0.9740490113478565; omega_0_34: 0.8548; omega_0_94: 0.1452; omega_1_24: 1.0;
Red. cost traj.: omega_1_218 [(0,24), (1,32), (2,39), (3,38), (4,30), (5,23), (6,17), (7,11), (8,10), (9,15), (10,22),
      (11,23), (12,31), (13,32)]; c*: 0.23784750884694675
RMP solved. z: 0.9740490113478565; omega_0_34: 0.8548; omega_0_94: 0.1452; omega_1_24: 1.0;
Red. cost traj.: omega_1_219 [(0,30), (1,37), (2,44), (3,45), (4,39), (5,31), (6,23), (7,17), (8,18), (9,25), (10,32),
      (11,31), (12,23), (13,16)]; c*: 0.23785954329547163
RMP solved. z: 0.9740490113478565; omega_0_34: 0.8548; omega_0_94: 0.1452; omega_1_24: 1.0;
Red. cost traj.: omega_1_220 [(0,16), (1,23), (2,31), (3,38), (4,37), (5,29), (6,22), (7,23), (8,31), (9,38), (10,37),
      (11,29), (12,22), (13,23)]; c*: 0.23782008974763516
RMP solved. z: 0.9740490113478565; omega_0_34: 0.8548; omega_0_94: 0.1452; omega_1_24: 1.0;
k: 0; UB: 0.9740490113478565; LB: 0.9769185224864061; Prune
#nodes processed: 9
#nodes pruned: 7
Optimal search strategy, with PD: 0.9769185224864061
[(0,34), (1,26), (2,19), (3,20), (4,28), (5,35), (6,34), (7,26), (8,19), (9,20), (10,28), (11,35), (12,34), (13,26)]
[(0,31), (1,23), (2,22), (3,29), (4,37), (5,38), (6,31), (7,23), (8,22), (9,29), (10,37), (11,38), (12,31), (13,23)]
```

Index

Bibliography

[1] D. Dumery. The mediterranean sea: a front door to irregular migration. UN High Commissioner for Refugees (UNHCR), 2016. URL `http://www.refworld.org/topic,50ffbce4132,50ffbce414a,58371b8d4,0,,,.html`. Last accessed: 22 Dec 2016.

[2] T. Kratzke, L. Stone, and J. Frost. Search and rescue optimal planning system. In *Proceedings of the 13th Conference on Information Fusion*, pages 1–8, jul 2010.

[3] K. E. Trummel and J. R. Weisinger. The complexity of the optimal searcher path problem. *Operations Research*, 34(2):324–327, mar 1986.

[4] J. C. Foraker. *Optimal search for moving targets in continuous time and spcce using consistent approximations*. PhD thesis, Naval Postgraduate School, Monterey, CA., 2011.

[5] J. L. Hibey. Control-theoretic approach to optimal search for a class of Markovian targets. In *Proceedings of the 1982 American Control Conference*, pages 705–709. IEEE, 1982.

[6] A. Ohsumi. Stochastic control with searching a randomly moving target. In *Proceedings of the 1984 American Control Conference*, pages 500–504, jun 1984.

[7] C. L. Walton, Q. Gong, I. Kaminer, and J. O. Royset. Optimal motion planning for searching for uncertain targets. In *World Congress*, volume 19, pages 8977–8982, 2014.

[8] J. Nygårds, P. Skoglar, J. Karlholm, M. Ulvklo, and R. Björström. Towards concurrent sensor and path planning-a survey of planning methods applicable to uav surveillance. *Sensor Technology Scientific Report ISSN*, pages 1650–1942, 2005.

[9] M. Raap, S. Meyer-Nieberg, S. Pickl, and M. Zsifkovits. Aerial vehicle search-path optimization: A novel method for emergency operations. *Journal of Optimization Theory and Applications*, 172(3):965–983, 2017.

[10] M. Raap. Cooperative search for moving targets in a maritime environment using a RHC-BILP approach. Master's thesis, Universität Ulm, 2014.

[11] M. Raap, M. Zsifkovits, and S. Pickl. Trajectory optimization under kinematical constraints for moving target search. *Computers & Operations Research*, 88(Supplement C):324 – 331, 2017.

[12] M. Raap, M. Moll, M. Zsifkovits, and S. Pickl. Utilizing Dual Information for Moving Target Search Trajectory Optimization. In B. Hardy, A. Qazi, and S. Ravizza, editors, *5th Student Conference on Operational Research (SCOR 2016)*, volume 50 of *OpenAccess Series in Informatics (OASIcs)*, pages 1:1–1:10, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[13] M. Kress and J. O. Royset. Aerial search optimization model (ASOM) for uavs in special operations. *Military Operations Research*, 13(1):23–33, 2008.

[14] F. Bourgault, T. Furukawa, and H. Durrant-Whyte. Optimal search for a lost target in a bayesian world. In S. Yuta, H. Asama, E. Prassler, T. Tsubouchi, and S. Thrun, editors, *Field and Service Robotics*, volume 24 of *Springer Tracts in Advanced Robotics*, pages 209–222. Springer Berlin Heidelberg, 2006.

[15] J. Norris. *Markov Chains*. Number 2008 in Cambridge series in statistical and probabilistic mathematics. Cambridge University Press, 1999.

[16] T. Furukawa, H. Durrant-Whyte, and B. Lavis. The element-based method - theory and its application to bayesian search and tracking. In *Proceedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2807–2812, oct 2007.

[17] J. N. Eagle and J. R. Yee. An optimal branch-and-bound procedure for the constrained path, moving target search problem. *Operations Research*, 38(1):110–114, feb 1990.

[18] G. H. Martins. A new branch and bound procedure for computing optimal search paths. Master's thesis, Naval Postgraduate School, Monterey, CA, 1993.

[19] A. R. Washburn. Branch and bound methods for a search problem. *Naval Research Logistics (NRL)*, 45(3):243–257, 1998.

[20] S. J. Benkoski, M. G. Monticino, and J. R. Weisinger. A survey of the search theory literature. *Naval Research Logistics*, 38(4):469–494, 1991.

[21] P. Skoglar. UAV path and sensor planning methods for multiple ground target search and tracking-A literature survey. 2007.

[22] S. Waharte and N. Trigoni. Supporting search and rescue operations with UAVs. In *Proceedings of the 2010 International Conference on Emerging Security Technologies*, pages 142–147, sep 2010.

[23] T. H. Chung, G. A. Hollinger, and V. Isler. Search and pursuit-evasion in mobile robotics. *Autonomous Robots*, 31(4):299–316, 2011.

[24] J. N. Eagle. The optimal search for a moving target when the search path is constrained. *Operations Research*, 32(5):1107–1115, 1984.

[25] T. Stewart. Search for a moving target when searcher motion is restricted. *Computers & Operations Research*, 6(3):129–140, 1979.

[26] S. S. Brown. Optimal search for a moving target in discrete time and space with an exponential detection function. In K. B. Haley and L. D. Stone, editors, *Search Theory and Applications*, volume 8 of *NATO Conference Series*, pages 221–229. Springer US, 1980.

[27] T. Stweart. Experience with a branch-and-bound algorithm for constrained searcher motion. In *Search theory and applications*, pages 247–253. Springer, 1980.

[28] M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval Research Logistics*, 3(1-2):95–110, 1956.

[29] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithms*. MIT Press, Cambridge, MA, USA, 2009.

[30] A. R. Washburn. *Branch and bound methods for search problems*. Monterey, California. Naval Postgraduate School, 1995.

[31] A. R. Washburn. Search for a moving target: The FAB algorithm. *Operations Research*, 31(4):739–751, 1983.

[32] H. Lau, S. Huang, and G. Dissanayake. Discounted MEAN bound for the optimal searcher path problem with non-uniform travel times. *European Journal of Operational Research*, 190(2):383–397, 2008.

[33] H. Sato. *Path optimization for single and multiple searchers: models and algorithms*. PhD thesis, Naval Postgraduate School, 2008.

[34] M. Morin, L. Lamontagne, I. Abi-zeid, P. Lang, and P. Maupin. The optimal searcher path problem with a visibility criterion in discrete time and space. In *Proceedings of the 12th International Conference on Information Fusion*, pages 2217–2224, 2009.

[35] R. Hohzaki and K. Iida. Optimal strategy of route and look for the path constrained search problem with reward criterion. *European Journal of Operational Research*, 100 (1):236–249, 1997.

[36] H. Sato and J. O. Royset. Path optimization for the resource-constrained searcher. *Naval Research Logistics*, 57(5):422–440, 2010.

[37] J. N. Eagle. *The approximate solution of a simple constrained search path moving target problem using moving horizon policies*. Monterey, California. Naval Postgraduate School, 1984.

[38] L. C. Thomas and J. N. Eagle. Criteria and approximate methods for path-constrained moving-target search problems. *Naval Research Logistics*, 42(1):27–38, 1995.

[39] S.-P. Hong, S.-J. Cho, and M.-J. Park. A pseudo-polynomial heuristic for path-constrained discrete-time markovian-target search. *European Journal of Operational Research*, 193(2):351–364, mar 2009.

[40] P. Lanillos, E. Besada-Portas, G. Pajares, and J. Ruz. Minimum time search for lost targets using cross entropy optimization. In *Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 602–609, oct 2012.

[41] M. Morin, A. P. Papillon, I. Abi-Zeid, F. Laviolette, and C. G. Quimper. Constraint programming for path planning with uncertainty. In M. Milano, editor, *Principles and Practice of Constraint Programming*, Lecture Notes in Computer Science, pages 988–1003. Springer Berlin Heidelberg, 2012.

[42] M. Morin, L. Lamontagne, I. Abi-Zeid, and P. Maupin. The ant search algorithm: An ant colony optimization algorithm for the optimal searcher path problem with visibility. In *Canadian Conference on Artificial Intelligence*, pages 196–207. Springer, 2010.

[43] T. J. Stewart. Optimizing search with positive information feedback. *Naval Research Logistics*, 32(2):263–274, 1985.

[44] S.-P. Hong, S.-J. Cho, M.-J. Park, and M.-G. Lee. Optimal search-relocation trade-off in markovian-target searching. *Computers & Operations Research*, 36(6):2097–2104, jun 2009.

[45] R. F. Dell, J. N. Eagle, G. H. Alves Martins, and A. G. Santos. Using multiple searchers in constrained-path, moving-target search problems. *Naval Research Logistics*, 43(4): 463–480, 1996.

[46] J. O. Royset and H. Sato. Route optimization for multiple searchers. *Naval Research Logistics*, 57(8):701–717, 2010.

[47] S. Lee and J. R. Morrison. Decision support scheduling for maritime search and rescue planning with a system of uavs and fuel service stations. In *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1168–1177, June 2015.

[48] G. E. Collins, J. R. Riehl, and P. S. Vegdahl. A UAV routing and sensor control optimization algorithm for target search. In *Proceedings of the 2007 Defense and Security Symposium*, pages 65610D–65610D. International Society for Optics and Photonics, 2007.

[49] F. Bourgault, T. Furukawa, and H. Durrant-Whyte. Coordinated decentralized search for a lost target in a bayesian world. In *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, pages 48–53, oct 2003.

[50] J. Hu, L. Xie, and J. Xu. Vision-based multi-agent cooperative target search. In *Proceedings of the 2012 International Conference on Control Automation Robotics Vision*, pages 895–900, 2012.

[51] X. Xiao, Z. Dong, J. Wu, and H. Duan. A cooperative approach to multiple uavs searching for moving targets based on a hybrid of virtual force and receding horizon. In *IEEE 10th International Conference on Industrial Informatics*, pages 1228–1233. IEEE, 2012.

[52] E.-M. Wong, F. Bourgault, and T. Furukawa. Multi-vehicle bayesian search for multiple lost targets. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 3169–3174, apr 2005.

[53] F. Bourgault, T. Furukawa, and H. Durrant-Whyte. Decentralized bayesian negotiation for cooperative search. In *Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 2681–2686, sep 2004.

[54] F. M. Delle Fave, Z. Xu, A. Rogers, and N. R. Jennings. Decentralised coordination of unmanned aerial vehicles for target search using the Max-Sum algorithm. In *Proceedings of the 2010 Workshop on Agents in Real Time and Environment*, pages 35–44, may 2010.

[55] D. T. Cole. *A cooperative UAS architecture for information-theoretic search and track.* PhD thesis, University of Sydney, 2009.

[56] H. Peng, M.-l. Huo, Z.-z. Liu, and W. Xu. Simulation analysis of cooperative target search strategies for multiple UAVs. In *Control and Decision Conference (CCDC), 2015 27th Chinese*, pages 4855–4859. IEEE, 2015.

[57] W. Meng, Z. He, R. Su, P. K. Yadav, R. Teo, and L. Xie. Decentralized multi-UAV flight autonomy for moving convoys search and track. *IEEE Transactions on Control Systems Technology*, PP(99):1–8, 2016.

[58] D. J. MacKay. *Information Theory, Inference & Learning Algorithms.* Cambridge University Press, New York, NY, USA, 2002.

[59] S. Kullback and R. A. Leibler. On information and sufficiency. *The annals of mathematical statistics*, pages 79–86, 1951.

[60] D. P. Bertsekas. *Dynamic Programming and Optimal Control*, volume 2. Athena Scientific, 4th edition, 2012.

[61] M. Asselin. *An introduction to aircraft performance.* AIAA Education Series, American Institute of Aeronautics and Astronautics, 1997.

[62] D. P. Raymer et al. *Aircraft design: a conceptual approach.* AIAA Education Series, American Institute of Aeronautics and Astronautics, 2nd edition, 1992.

[63] NATO. ATP-10(D) Search and Rescue. 1995.

[64] E. Álvarez-Miranda, I. Ljubić, and P. Mutzel. The maximum weight connected subgraph problem. In *Facets of Combinatorial Optimization*, pages 245–270. Springer, 2013.

[65] D. B. West et al. *Introduction to graph theory*, volume 2. Prentice Hall Upper Saddle River, 2001.

[66] A. M. Geoffrion. Generalized benders decomposition. *Journal of optimization theory and applications*, 10(4):237–260, 1972.

[67] T. Bektaş. Formulations and benders decomposition algorithms for multidepot salesmen problems with load balancing. *European Journal of Operational Research*, 216 (1):83–93, 2012.

[68] Y. Dumas, J. Desrosiers, E. Gelinas, and M. M. Solomon. An optimal algorithm for the traveling salesman problem with time windows. *Operations Research*, 43(2):367–371, 1995.

[69] J. H. Bookbinder and K. E. Reece. Vehicle routing considerations in distribution system design. *European Journal of Operational Research*, 37(2):204–213, 1988.

[70] A. Federgruen and P. Zipkin. A combined vehicle routing and inventory allocation problem. *Operations Research*, 32(5):1019–1037, 1984.

[71] J.-F. Cordeau, F. Soumis, and J. Desrosiers. A benders decomposition approach for the locomotive and car assignment problem. *Transportation science*, 34(2):133–149, 2000.

[72] M. L. Fisher and R. Jaikumar. A generalized assignment heuristic for vehicle routing. *Networks*, 11(2):109–124, 1981.

[73] T. R. Sexton and L. D. Bodin. Optimizing single vehicle many-to-many operations with desired delivery times: I. Scheduling. *Transportation Science*, 19(4):378–410, 1985.

[74] J.-F. Cordeau, G. Stojković, F. Soumis, and J. Desrosiers. Benders decomposition for simultaneous aircraft routing and crew scheduling. *Transportation science*, 35(4): 375–388, 2001.

[75] C. Y. Lee. A cross decomposition algorithm for a multiproduct-multitype facility location problem. *Computers & Operations Research*, 20(5):527 – 540, 1993.

[76] A. I. Corréa, A. Langevin, and L.-M. Rousseau. Scheduling and routing of automated guided vehicles: A hybrid approach. *Computers & Operations Research*, 34(6):1688 – 1707, 2007.

[77] M.-C. Lai, H.-S. Sohn, T.-L. Tseng, and D. L. Bricker. A hybrid benders/genetic algorithm for vehicle routing and scheduling problem. *International Journal of Industrial Engineering*, 19(1):33–46, 2012.

[78] M. Fischetti, I. Ljubić, and M. Sinnl. Benders decomposition without separability a computational study for capacitated facility location problems. *European Journal of Operational Research*, 253(3):557–569, 2016.

[79] F. Vanderbeck. On dantzig-wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm. *Operations Research*, 48(1): 111–128, 2000.

[80] C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. Savelsbergh, and P. H. Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, 46(3):316–329, 1998.

[81] M. Lübbecke and J. Desrosiers. Selected topics in column generation. *Operations Research*, 53(6):1007–1023, 2005.

[82] C. Barnhart, C. A. Hane, and P. H. Vance. Using branch-and-price-and-cut to solve origin-destination integer multicommodity flow problems. *Operations Research*, 48 (2):318–326, 2000.

[83] R. Baldacci, E. Bartolini, A. Mingozzi, and R. Roberti. An exact solution framework for a broad class of vehicle routing problems. *Computational Management Science*, 7 (3):229–268, 2010.

[84] G. Desaulniers, J. Desrosiers, M. M. Solomon, F. Soumis, D. Villeneuve, et al. A unified framework for deterministic time constrained vehicle routing and crew scheduling problems. In *Fleet management and logistics*, pages 57–93. Springer, 1998.

[85] G. Desaulniers, J. Desrosiers, and S. Spoorendonk. Cutting planes for branch-and-price algorithms. *Networks*, 58(4):301–310, 2011.

[86] F. Vanderbeck. Implementing mixed integer column generation. In *Column generation*, pages 331–358. Springer, 2005.

[87] J. Desrosiers, J. B. Gauthier, and M. E. Lübbecke. Row-reduced column generation for degenerate master problems. *European Journal of Operational Research*, 236(2): 453–460, 2014.

[88] G. Desaulniers, J. Desrosiers, and M. Solomon. *Column generation*, volume 5. Springer Science & Business Media, 2006.

[89] G. B. Dantzig and P. Wolfe. Decomposition principle for linear programs. *Operations Research 8*, pages 101–111, 1960.

[90] U.S. Naval Postgraduate School Center for Network Innovation and Experimentation (CENETIX). URL `http://my.nps.edu/web/cenetix`. Last accessed: 13-04-2017.

[91] CENETIX Resource Portal. URL `http://my.nps.edu/web/cenetix/resource-portal`. Last accessed: 13-04-2017.

[92] J. L. Ford. Radiological Dispersal Devices. Assessing the Transnational Threat. Technical report, DTIC Document, 1998.

[93] P. R. Rickert. The likely effect of a radiological dispersion device. 2005.