Radar Resource Management for Multi-Target Tracking Using Model Predictive Control

by

Thies de Boer

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Thursday June 24, 2021 at 1:00 PM.

Student number:	4513614	
Project duration:	October 1, 2020 – June 24, 2021	
Thesis committee:	Prof. DSc. A. Yarovoy,	TU Delft, chair professor
	Dr. ir. J. N. Driessen,	TU Delft, supervisor
	M. I. Schöpe MSc.,	TU Delft, supervisor
	Dr. ir. A. J. van Genderen,	TU Delft
	Dr. P. Mohajerin Esfahani,	TU Delft, assistant professor

This thesis is confidential and cannot be made public until December 31, 2021.

TU Delft

Preface

This thesis concludes my master Embedded Systems following my bachelor Electrical Engineering at the Delft University of Technology. When I read about the topic of radar resource management and the possibility of turning it into my master thesis topic, my interest was immediately drawn.

It has been an interesting and challenging journey at the Microwave Sensing, Signals and Systems group and my thesis could not have been here before you if it were not for certain people. I would like to thank my daily supervisor Max Schöpe for guiding me through my thesis project for the last 9 months. I would also like to thank my supervisor Dr. ir. Hans Driessen for the interesting discussions we had that helped steer me into the right direction during my thesis project. Furthermore, I would like to thank Prof. DSc. Alexander Yarovoy and the other members of the MS3 group for their insightful feedback and suggestions. Finally, I would like to thank my family for their support during my studies at TU Delft.

Thies de Boer
Tinte, June 2021

Abstract

With modern multi-function radars becoming more flexible, handling the limited amount of resources of these radars becomes increasingly important. In this thesis the radar resource management (RRM) problem in a multi-target tracking scenario is considered. Partially observable Markov decision processes (POMDPs) are used to describe each tracking task. By comparing the future effect of radar actions using model predictive control (MPC), the POMDPs are solved in a non-myopic way. The model predictive control problem can be decoupled into sub-problems using Lagrangian Relaxation to reduce the computational complexity of the solution method. An algorithm based on golden section search is employed to find the Lagrange multiplier. An interacting multiple model filter is used to allow the method to be effective in RRM problems involving the tracking of targets performing a broad number of maneuvers. The novel approach is compared to an existing solution method based on policy rollout and Monte Carlo sampling. Through simulations of dynamic multi-target tracking scenarios in which the cost and computational complexity of different approaches are compared, it was shown that the computational complexity is greatly reduced while the resulting resource allocation results remain similar.

Contents

List of Figures

List of Tables

Nomenclature

AODB	Approximately Optimal Dynamic Budget balancing
CFA	Cost Function Approximation
CR	Cognitive Radar
CT	Constant Turn
CV	Constant Velocity
DBF	Digital Beamforming
DLA	Direct Lookahead Approximation
EKF	Extended Kalman Filter
EM	Electromagnetic
IMM	Interacting Multiple Model
LR	Lagrangian Relaxation
MCTS	Monte Carlo tree search
MDP	Markov Decision Process
MFR	Multi-function radar
MPC	Model Predictive Control
PFA	Policy Function Approximation
POMDP	Partially Observable Markov Decision Process
RCS	Radar Cross Section
RRM	Radar Resource Management
SNR	Signal-to-noise ratio
VFA	Value Function Approximation

1 Introduction

1.1. Introduction

The notion that conducting objects reflect radio waves goes back as far as the late 19th century. This has been used to detect the presence of objects, for example ships, since 1903. Early work used continuous wave (CW) transmissions, and relied upon interference between a transmitted wave and the signal received from a moving target. The main limitation of this method was that it could not be used to detect range. By modulation the transmitter output, for example by sending out pulses, the range could be measured based on the time it takes for a pulse to return. This technique formed the foundation for radar systems, where the term radar is the contraction of RAdio Detection And Ranging. Initially, the main drive behind the development of radar systems was to use them for military purposes, such as detecting the presence of enemy targets. After World War II, more interest arose to use radar systems for civilian applications, such as air traffic control, weather forecast, navigation and spacecraft among others. In recent years, advances in the field of radar led to the development of phased array antennas that allow the application of so-called digital beamforming (DBF). Together with advanced signal processing and arbitrary waveform generation, modern multi-function radars (MFR) became increasingly flexible. These radar systems can quickly adapt to changes to the environment by automatically adjusting the measurement parameters during runtime. This adaptive process is often called radar resource management (RRM) and can be considered to be a part of cognitive radar (CR) [4–8]. Possible applications for RRM can be found in many domains, such as autonomous driving and traffic monitoring or maritime and air surveillance. In this thesis a solution method to the RRM problem will be presented. This thesis shares part of its content with the paper in [9] describing the same solution method.

1.2. General radar introduction

In this section a general radar introduction will be given to provide some background information that is needed for the remainder of the thesis. This general radar introduction is loosely based on the books of Lynn and Skolnik [10, 11]. A distinction can be made between three detection modes of a radar (see Fig 1.1): monostatic, bistatic and electronic warfare support measures (ESM). ESM is a form of passive radar, while the other two modes are forms of active radar. Passive radar means that there is no dedicated transmitter in the system. For the rest of this thesis the use of active radar is assumed. In the monostatic mode, the transmitter antenna and the receiver antenna are at the same position. In the bistatic mode, the transmitter antenna and the receiver antenna are at a distance from each other and the receiver can only detect a signal emitted from a transmitter source at a different location. In this thesis only radars operating in the monostatic detection mode are considered. To measure the distance, also called range, of a target, the transmitter of the radar

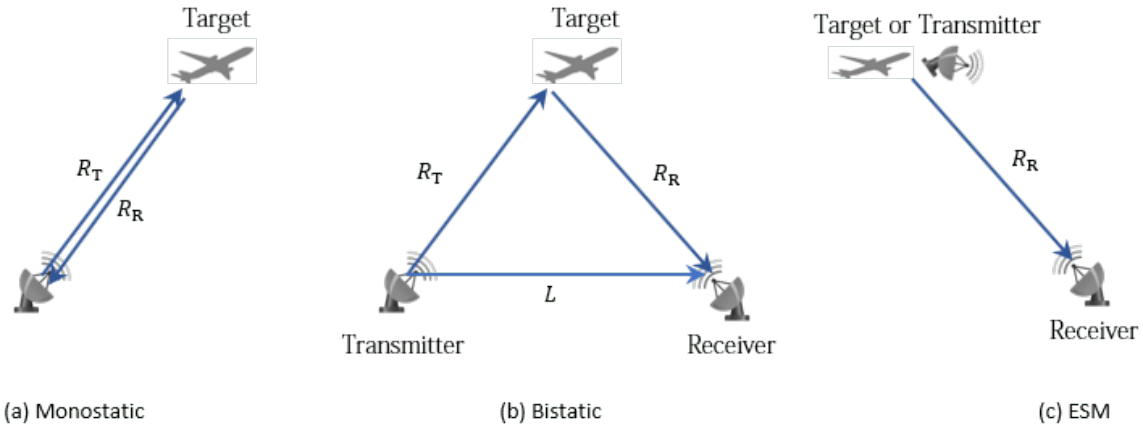

Figure 1.1: Different radar modes. [1]

sends electromagnetic (EM) pulses towards a target. The range of the target can then be determined by measuring the time T_R it took for the pulse to travel to the target and back to the radar. As an EM wave propagates at the speed of light ($c \approx 3.0 \cdot 10^8$[m/s]), the range R of the target is

$$R = \frac{cT_R}{2}. \tag{1.1}$$

The ability of a target to reflect the EM energy sent by the radar can be summarized by the term known as the radar cross section (RCS). It is a characteristic of a specific object and can be considered to be the surface area of the target as seen by the radar. The larger the RCS, the better the target is at returning the EM wave back towards the radar. The RCS of a target depends, among other things, on the size of the target, the shape of the target, the texture of the target and the material the target consists of.

1.3. Radar Resource Management

RRM is the problem of assigning the limited amount of resources of a radar among the different tasks the radar needs to perform. Countless heuristic solutions to this problem have been presented, many of them focusing on scheduling tasks with a fixed resource need (as mentioned in the overviews in [12, 13]. In an overload situation, which is a situation where the available budget is not enough to fully execute all tasks, this inevitably leads to tasks of lower priority being dropped. This is exemplified in Fig. 1.2. In the upper case in this figure it can be seen that task 4 is fully dropped,

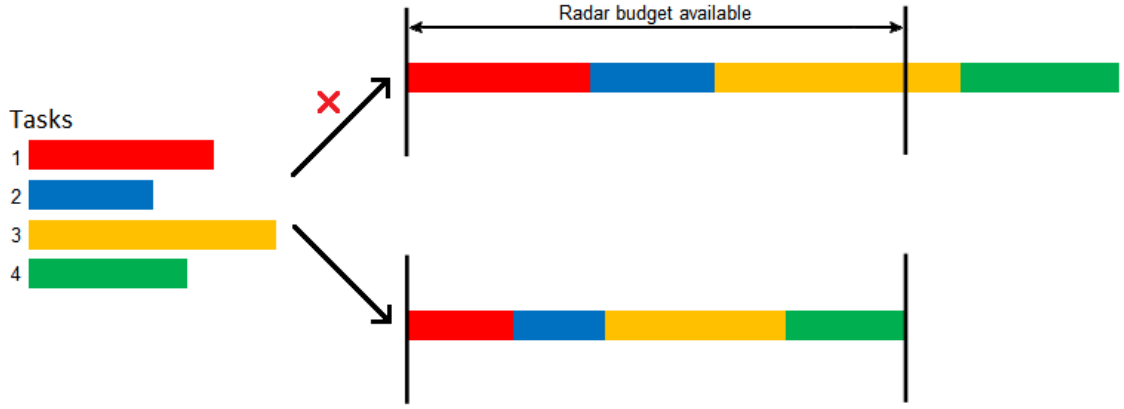

Figure 1.2: Fixed task length vs variable task length.

while task 3 can only be executed in part. If all the tasks were to have the same priority, the decision which task needs to be dropped would potentially be taken at random. Therefore, the full potential of RRM can only be explored once the resource allocation fully depends on the tasks' impact on the mission. To allow all tasks to be considered equally, it is considered that the resource allocation is adjustable and not restricted to specific predefined values, as can be seen in the lower case of the figure. Most of the available RRM approaches focus on single tasks. For example, within a tracking scenario, many approaches try to keep the track quality constant [14, 15]. These approaches solve the RRM problem by applying POMDP solution methods, which was shown to be a good framework for modelling RRM problems. When multiple tasks are considered, this problem becomes more difficult to solve. By design, an MFR system usually operates at its resource limit as it has the ability to quickly move its attention from one target to another. Therefore, resource allocation can be considered to be a balancing act. Consequently, increasing the resources for one task will automatically decrease the resources for all the other ones, which will deteriorate their sensing performance.

1.4. Problem Statement

In this section a high level overview of the RRM problem that will be tackled in the remainder of this thesis will be given. Consider the situation where a single radar is tasked with tracking N different moving objects in a two-dimensional plane and the sensor only has a limited amount of sensor time, which is the time it spends taking observations, available. This leads to the problem of how to allocate the sensor budget, which in our case is the sensor time, between the different objects to be tracked. The goal of this thesis is to provide a solution method that, given such a tracking scenario, provides an approximately optimal distribution of the radar resources over the tracking tasks. It is important that this solution method manages to compute this budget distribution, while taking into account changes that occur during operation. This includes changes to the environment in which the objects are tracked, changes to the number of objects to be tracked and changes in the movement of the objects to be tracked. This leads to a stochastic optimization problem, which can be modelled as a Partially Observable Markov Decision Process (POMDP). In chapter 2 it will be discussed in detail what a POMDP is and how the problem described in this section can be formulated as a POMDP. This thesis builds on the work done in [2], where a policy rollout solution method was introduced to solve the budget distribution problem.

1.5. Thesis Structure

This chapter gave an introduction to the RRM problem and provided background information on radar systems. This section describes how the remainder of this thesis is structured. In chapter 2 background information on POMDPs and target tracking will be provided. In chapter 3 a literature review will be carried out presenting an overview of the available solution methods to POMDPs. Chapter 4 provides a detailed explanation on the proposed solution method and provides an analysis of the results obtained using this method. In chapter 5 an extension to the proposed solution will be explained and analysed. Chapter 6 will summarize the results obtained and draw conclusions. Finally, recommendations will be given for possible related future research.

2

Background information

In this chapter it will be explained what a POMDP is and how this can be used to formulate the RRM problem. It will provide background information on target tracking using a Kalman filter and will explain how the problem can be written as an optimization problem. Furthermore, an existing solution to the formulated RRM problem will be described.

2.1. Markov Decision Processes

Many presented RRM solution methods assume a Markov decision process (MDP) in their RRM solution method (e.g. [16, 17]). An MDP is defined in [18] as the tuple (S, A, T, R), where those 4 symbols relate to the following system information:

- S: Set of states. The state describes the features of a system that are evolving over time.
- A: Set of actions. Actions are the inputs that can be applied to the system.
- T: Transition function. When applying an action $a \in A$ in a state $s \in S$, the state transitions from s to a new state $s' \in S$ with probability $T(s, a, s')$.
- R: Reward function. The reward function specifies the reward for being in a state or for taking an action when in a state.

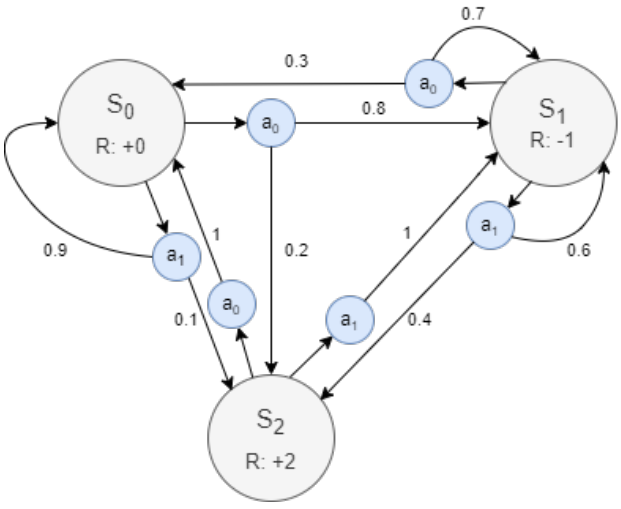

Figure 2.1: Example Markov Decision Process.

A decision process is said to be *Markovian* if the result of taking an action in a state depends on the history only through its current state. An example of such a decision process, where S, A, T and R are given can be seen in Fig. 2.1. The numbers on the arrows indicate the transition probabilities. The solution of such a decision process is called a policy. A policy π defines the action the agent selects in each state. The optimal policy π^* of, for example the MDP in Fig. 2.1 is one that maximizes the discounted reward over some (potentially infinite) time horizon H:

$$\pi^* = \arg\max_{\pi} \mathbb{E}\big[\sum_{k=0}^{H-1} \gamma^k R(s, s')\big], \tag{2.1}$$

where $\gamma \in [0, 1]$ is called the discount factor, that weighs how important rewards now are compared to future rewards. The resulting policy will inform the agent if action a_0 or a_1 should be applied when in each of the states S_0, S_1 and S_2. In the described RRM problem, the problem cannot quite be modelled as an MDP, as it is not possible to fully determine the state, due to e.g. noisy measurements and target maneuverability. Therefore, the decision process is said to be partially observable and can be modelled as a POMDP. POMDPs are generalizations of MDP's and will be discussed in the next section.

2.2. Partially Observable Markov Decision Processes

In contrast to MDP's, POMDPs can be used to describe a decision process in which the agent cannot directly observe the underlying state. This section will provide basic background information and will illustrate how the RRM problem described in section 1.4 can be modelled as a POMDP.

In literature (e.g. in [17, 19]) a POMDP is defined as the tuple (S, A, T, R, Ω, O). In the POMDP configuration that models the RRM problem, those 6 symbols relate to the following system information:

- S: Set of states.
- A: Set of actions.
- T: Transition function.
- R: Reward function.
- Ω: Observations. The observations of the system that can be used to gain more information about the state and make better decisions.
- O: Observation law. The observation law provides the probability distribution of making an observation given the state and the action.

Note that the first 4 items are the same as in the MDP definition in section 2.1. In POMDPs the state is not fully observable. Instead, a probability distribution over the state space is computed. This probability distribution over the state space is often referred to as the belief state. A solution to a POMDP is provided in the form of a policy function π, which maps each belief state to an action that should be taken. When optimizing over a certain time horizon H, the value function when in belief state $\boldsymbol{b}_k$ is given by

$$V_H^{\pi}(\boldsymbol{b}_k) = R(\boldsymbol{b}_k, A^{\pi}(\boldsymbol{b}_k)) + \mathbb{E}\,[V_{H-1}^{\pi}(\boldsymbol{B}_{k+1}^{\pi})|\boldsymbol{B}_k = \boldsymbol{b}_k], \tag{2.2}$$

where $\boldsymbol{b_k}$ is the belief state, $A^{\pi}(\boldsymbol{b}_k))$ is the mapping from the belief state $\boldsymbol{b}_k$ onto the action space following policy π and $\boldsymbol{B}_{k+1}^{\pi}$ is the belief state transition function when following policy π. This value function can be seen as the accumulated reward the agent receives when it is in belief state $\boldsymbol{b}_k$ and following policy π for the next H time-steps. A solution to the POMDP can now be obtained by

maximizing this value function, i.e. some policy π needs to be found that maps the action that leads to the maximum value function for each belief state $\boldsymbol{b}_k$:

$$\pi^* = \arg\max_{\pi} V_H^{\pi}(\boldsymbol{b}_k). \tag{2.3}$$

In chapter 3 an overview of different methods of (approximately) solving POMDPs will be provided. In the next sections, it will be explained how the RRM problem can be formulated as a POMDP.

2.3. Tracking

Since a tracking scenario is assumed in the RRM problem, a tracking filter needs to be chosen. For purely linear scenarios, a Kalman filter is the optimal solution. When a non-linear measurement transformation function or state transition function is assumed, an extended Kalman filter (EKF) or a particle filter are possible solutions. Generally, the tracking algorithm aims at computing the posterior density $p(\boldsymbol{s}_t^n|\boldsymbol{z}_t^n)$ of the object state. For this thesis, an EKF is used to track the targets. In this section it will be described how this EKF will be used in our RRM tracking problem.

2.3.1. Motion Model

There are N targets assumed in the scenario that are moving in a two-dimensional plane. The state of each target at time t can be defined as

$$\boldsymbol{s}_t^n = \begin{bmatrix} x_t^n & y_t^n & \dot{x}_t^n & \dot{y}_t^n \end{bmatrix}^\top, \tag{2.4}$$

where x_t^n, y_t^n and $\dot{x}_t^n, \dot{y}_t^n$ are the position and velocity in x and y direction of a Cartesian coordinate system, respectively. The new target state after a certain time T can be calculated following a state transition function:

$$\boldsymbol{s}_{t+T}^n = f\left(T, \boldsymbol{s}_t^n, \boldsymbol{w}_t^n\right), \tag{2.5}$$

where $\boldsymbol{s}_{t+T}^n$ is the state of target n at time $t+T$ and $\boldsymbol{w}_t^n \in \mathbb{R}^4$ is its maneuverability noise at time t, which is zero-mean and Gaussian. For our RRM problem a constant velocity (CV) model is assumed for the motion of the targets:

$$\boldsymbol{s}_{t+T}^n = \boldsymbol{F}\boldsymbol{s}_t^n + \boldsymbol{w}_t^n = \begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \boldsymbol{s}_t^n + \boldsymbol{w}_t^n, \tag{2.6}$$

where F is the state transition matrix. The process noise covariance matrix for target n is defined as

$$\boldsymbol{Q}_n = \begin{bmatrix} \frac{T^4}{4} & 0 & \frac{T^3}{2} & 0 \\ 0 & \frac{T^4}{4} & 0 & \frac{T^3}{2} \\ \frac{T^3}{2} & 0 & T^2 & 0 \\ 0 & \frac{T^3}{2} & 0 & T^2 \end{bmatrix} \sigma_{w,n}^2, \tag{2.7}$$

where $\sigma_{w,n}^2$ is the maneuverability noise variance.

2.3.2. Measurement Model

The assumed radar sensor takes noisy measurements of the state $\boldsymbol{s}_t^n$ by executing sensor actions $\boldsymbol{a}_t^n$ that are adjustable and influence the accuracy of the observations. At time t, the measurement of target n can be defined by

$$\boldsymbol{z}_t^n = h\left(\boldsymbol{s}_t^n, \boldsymbol{v}_t^n, \boldsymbol{a}_t^n\right), \tag{2.8}$$

where $h(\cdot)$ is the measurement transformation function, $\boldsymbol{v}_t^n$ with variance $\boldsymbol{\sigma}_n^2 = [\sigma_{r,n}^2 \sigma_{\theta,n}^2]^\top$ is the measurement noise, which is Gaussian and zero-mean and $\boldsymbol{a}_t^n$ is the chosen action for target n at time t. For our RRM problem the measurements are made in range (r) and azimuth (θ). The following measurement model is introduced to convert the state from Carthesian coordinates into polar coordinates:

$$h(\boldsymbol{s}_t^n) = \begin{bmatrix} \sqrt{(x_t^n)^2 + (y_t^n)^2} & \text{atan2}(\frac{y_t^n}{x_t^n}) \end{bmatrix}^\top. \tag{2.9}$$

Now, the measurement function can be defined as follows:

$$\boldsymbol{z}_t^n = h(\boldsymbol{s}_t^n) + \boldsymbol{v}_t^n. \tag{2.10}$$

The observation matrix $\boldsymbol{H}$ is the Jacobian of the measurement transformation function $h(\boldsymbol{s}_n)$ defined as (2.9):

$$\boldsymbol{H}_k^n = \left. \frac{\partial h}{\partial \boldsymbol{s}} \right|_{\boldsymbol{s}_k} \tag{2.11}$$

and will be used in the EKF in section 2.3.3. Based on the range of an object, the signal-to-noise ratio (SNR) of a measurement is determined in the same way as was described in [20], following equations by Koch [21]:

$$\text{SNR} = \text{SNR}_0 (\frac{\text{RCS}}{\text{RCS}_0})(\frac{\tau}{\tau_0})(\frac{r}{r_0})^{-4} e^{-2\Delta b}, \tag{2.12}$$

where RCS is the radar cross section, τ is the dwell time of the object and r is the distance between the target and the radar. The dwell time is the time the sensor spends on observing a target. Δb is a measure of relative beam positioning error. As it is assumed that the prediction is accurate enough to steer the beam towards the object, Δb is assumed to be 0, which results in the exponential term vanishing. Note that in this equation τ replaces the beam energy e in the original equations in [21]. In this equation τ is the action the radar takes. Thus, by changing τ the measurement quality for the different targets can be controlled.

Table 2.1: Reference measurement values.

Reference parameter	**Value**
Measurement variance in range ($\sigma_{r,0}^2$)	25 m^2
Measurement variance in angle ($\sigma_{\theta,0}^2$)	2e-3 rad^2
Reference SNR (SNR_0)	1
Reference RCS (RCS_0)	10 m^2
Reference dwell time (τ_0)	1 s
Reference range (r_0)	50 km

In Table 2.1 example values of RCS_0, τ_0 and r_0 as well as other measurement parameters can be found. These reference values will be used in this thesis to calculate the SNR. Finally, after computing the SNR of a target its measurement noise variance can now be calculated as:

$$\sigma_{r,n}^2 = \frac{\sigma_{r,0}^2}{SNR_{n,k}} \tag{2.13}$$

and

$$\sigma_{\theta,n}^2 = \frac{\sigma_{\theta,0}^2}{SNR_{n,k}}. \tag{2.14}$$

Finally, the measurement noise covariance matrix $\boldsymbol{R}$ can be constructed as follows:

$$\boldsymbol{R} = \begin{bmatrix} \sigma_{r,n}^2 & 0 \\ 0 & \sigma_{\theta,n}^2 \end{bmatrix}. \tag{2.15}$$

2.3.3. Extended Kalman filter

An EKF is a filter that combines the predictions made using the motion model with the measurements to minimize the tracking uncertainty. This filtering process consists of two stages. Firstly, during the *predict* stage the next state of a target is predicted based on the assumed motion model:

$$\hat{\boldsymbol{s}}_{k|k-1} = \boldsymbol{F}_k \hat{\boldsymbol{s}}_{k-1|k-1}. \tag{2.16}$$

This state estimate $\hat{\boldsymbol{s}}_{k|k-1}$ is called the a priori state estimate. Secondly, during the *update* stage the state prediction made during the *predict* stage is updated by multiplying the difference between the predicted state $\hat{\boldsymbol{s}}_{k|k-1}$ and the measured state $\tilde{\boldsymbol{z}}_k$ with the Kalman gain $\boldsymbol{K_k}$:

$$\hat{\boldsymbol{s}}_{k|k} = \hat{\boldsymbol{s}}_{k|k-1} + \boldsymbol{K_k}\tilde{\boldsymbol{z}}_k. \tag{2.17}$$

This state estimate is often referred to as the a posteriori state estimate. In Table 2.2 the equations to compute the state estimates during every *predict* and *update* step are listed. In this table $\boldsymbol{P}_{k|k-1}$ is the predicted error covariance matrix, which will be used in the next section to define the cost function.

Table 2.2: Extended Kalman filtering steps.

Predict stage:	
A priori state estimate:	$\hat{\boldsymbol{s}}_{k\|k-1} = \boldsymbol{F}_k \hat{\boldsymbol{s}}_{k-1\|k-1}$
A priori estimated error covariance:	$\boldsymbol{P}_{k\|k-1} = \boldsymbol{F}_k \boldsymbol{P}_{k-1\|k-1} \boldsymbol{F}_k^\top + \boldsymbol{Q}_k$
Update stage:	
Innovation residual:	$\tilde{\boldsymbol{z}}_k = \boldsymbol{z}_k - h(\hat{\boldsymbol{s}}_{k\|k-1})$
Innovation covariance:	$\boldsymbol{S}_k = \boldsymbol{H}_k \boldsymbol{P}_{k\|k-1} \boldsymbol{H}_k^\top + \boldsymbol{R}_k$
Kalman gain:	$\boldsymbol{K}_k = \boldsymbol{P}_{k\|k-1} \boldsymbol{H}_k^\top \boldsymbol{S}_k^{-1}$
A posteriori state estimate:	$\hat{\boldsymbol{s}}_{k\|k} = \hat{\boldsymbol{s}}_{k\|k-1} + \boldsymbol{K}_k \tilde{\boldsymbol{z}}_k$
A posteriori estimated error covariance:	$\boldsymbol{P}_{k\|k} = (\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{H}_k) \boldsymbol{P}_{k\|k-1}$

2.4. Cost function

In contrast to a reward function that is frequently used in literature about stochastic optimization as well as in the literature review in this thesis, for our RRM problem a cost function is used. A cost function is simply a reward function with its sign swapped. The aim of the cost function is to let the radar gather as much information about the targets as possible. Conversely, the solution of the problem should be one that minimizes the uncertainty about the targets. For this reason it was chosen to minimize the first two diagonal terms of the predicted error covariance matrix $\boldsymbol{P}_{k|k-1}$, corresponding to the predicted variance in x and y position of the objects:

$$c(\tau_k^n, T_k^n, \boldsymbol{s}_k^n) = \text{trace}\,(\boldsymbol{E}\boldsymbol{P}_{k|k-1}^n \boldsymbol{E}^\top) + \frac{500}{T_k^n}, \tag{2.18}$$

where

$$E = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}. \tag{2.19}$$

The second term of the cost function, $\frac{500}{T_k^n}$, is a penalty for switching between the different targets too frequently. In this cost function τ_k^n is the dwell time at time-step k of target n and T_k^n is the revisit time at time-step k of target n. This function only comprises of the cost of a single target at a single time-step k. To obtain the total cost over some horizon, the costs of all targets at all time-steps within this horizon need to be summed. Note that this cost function only serves as an example. Different cost functions can be used based on the demands of the user of the radar system.

2.5. Optimization Setup

The RRM problem is solved by finding the revisit times and dwell times $\{T_k^n, \tau_k^n\}$ for $n = 1...N$ and $k = 0...H-1$ that minimize the cost function in 2.18 summed over some horizon H and over n objects. The solution can be considered to be a budget distribution over the tasks, deciding which share of the radar resources each target receives. The fraction that target n receives at time-step k is $\frac{\tau_k^n}{T_k^n}$. As the total fraction of all targets must not exceed B_{max} the minimization problem is constrained. The optimization problem can be defined as follows:

$$\begin{aligned} \min_{\boldsymbol{\tau},\boldsymbol{T}} \quad & \sum_{k=0}^{H-1} \Big(\sum_{n=1}^{N} c(\tau_k^n, T_k^n, s_k^n)\Big) \\ \text{s.t.} \quad & \sum_{n=1}^{N} \frac{\tau_k^n}{T_k^n} \leq B_{max} \text{ for } k = 0 \dots H-1. \end{aligned} \tag{2.20}$$

Solving this problem comes down to minimizing the cost of the N objects summed over the time horizon H. The result of this optimization problem provides the actions of the radar system:

$$\boldsymbol{\tau} = \begin{bmatrix} \tau_1^1 & \tau_2^1 & \dots & \tau_H^1 \\ \tau_1^2 & \tau_2^2 & \dots & \tau_H^2 \\ \vdots & \vdots & \ddots & \vdots \\ \tau_1^N & \tau_2^N & \dots & \tau_H^N \end{bmatrix} \text{ and } \boldsymbol{T} = \begin{bmatrix} T_1^1 & T_2^1 & \dots & T_H^1 \\ T_1^2 & T_2^2 & \dots & T_H^2 \\ \vdots & \vdots & \ddots & \vdots \\ T_1^N & T_2^N & \dots & T_H^N \end{bmatrix}, \tag{2.21}$$

where τ_j^i and T_{k+j}^i correspond to the dwell time and the revisit time of the i-th object and the $(k+j)$-th time-step respectively. Note that in traditional control problems, the actions or inputs to the system usually directly affect the state of the system. In the control problem described in this chapter the inputs to the system do not affect the state, i.e. the actions do not influence the movement the target makes. Instead, the actions in this RRM problem affect the quality of the sensor measurements.

2.6. AODB algorithm

An existing solution method to the specified RRM problem is the approximately optimal dynamic budget balancing (AODB) algorithm introduced in [2]. This section will describe how the solution method provided in that paper is used to solve the RRM problem. Firstly, the problem is split into a sub-problem per target using Lagrangian Relaxation. Then, policy rollout is used to find $\{\boldsymbol{\tau}, \boldsymbol{T}\}$ for each of these sub-problems.

2.6.1. Lagrangian Relaxation

Using Lagrangian Relaxation the constraints of the optimization problem described in 2.20 can be brought into the objective function. The optimization function now becomes:

$$\max_{\lambda} \quad \Big(\underbrace{\min_{\tau,T} \quad \sum_{t=k}^{k+H} \Big(\sum_{n=1}^{N} c(\tau_t^n, T_t^n, \boldsymbol{s}_t^n) + \lambda \frac{\tau_t^n}{T_t^n} \Big)}_{\text{sum of independent minimization problems}} - \lambda B_{max} \Big). \tag{2.22}$$

In this optimization problem it can be seen that the minimization problem consists of the sum of N minimization problems which are now, for a fixed iteration of λ, independent of each other as they are no longer linked to each other by the constraints. As a result, the problem can be rewritten into N sub-problems for the N different objects that need to be tracked:

$$\min_{\boldsymbol{\tau},\boldsymbol{T}} \quad \sum_{t=k}^{k+H} c(\tau_t^n, T_t^n, s_t^n) + \lambda \frac{\tau_t^n}{T_t^n}. \tag{2.23}$$

These sub-problems are independent of each other and are solved by performing policy rollout.

2.6.2. Policy Rollout

Policy rollout is a Monte Carlo sampling based technique. Firstly, the action space is discretized, generating a finite number of $\{\tau, T\}$-pairs as trial actions for each of the targets. A number of samples of horizon length H of the future are generated for each $\{\tau, T\}$-pair. Each of these samples is considered a rollout. For the first action of the rollout τ and T are applied as inputs. After that, for the remainder of the horizon length, the action provided by the base policy is used. In this algorithm, the base policy was assumed to repeat τ and T as inputs for the full horizon length. For a $\{\tau, T\}$-pair, the average cost over its rollouts is computed using 2.23. Finally, the actions that resulted in the lowest average cost is the solution to a sub-problem. The values of τ and T that are found for each of the sub-problems are combined to determine if the available radar budget B_{max} is not exceeded:

$$|\sum_{n=1}^{N} \frac{\tau_t^n}{T_t^n} - B_{max}| \leq \varepsilon, \tag{2.24}$$

where ε is some constraint tolerance parameter. Until this constraint is satisfied, policy rollout is repeated with a new value of λ, which is determined using the subgradient method described in Algorithm 1, where l indicates the iteration index:

Algorithm 1: Subgradient method for finding λ.

Step 1 Set initial $\lambda = \lambda_0$
Step 2 Generate policy rollout solution using λ
Step 3 Compute subgradient to be $\mu_l^\lambda = |\sum_{n=1}^{N} \frac{\tau_n^t}{T_n^t} - B_{max}|$
Step 4
if $\mu_l^\lambda \leq \varepsilon$ **then**
  return λ
else
  $\lambda_{l+1} \leftarrow \max\{0, \lambda_l + \gamma_l \mu_l^\lambda\}$
  $l \leftarrow l + 1$
  Go back to step 2
end

An overview of the AODB algorithm is provided in Fig. 2.2. In this figure θ_k^n refers to the resulting budget $\{\tau_k^n, T_k^n\}$ of object n at time-step k. While the AODB algorithm reported good results in terms of realized cost, the computation time can be quite large. This stems from the fact that multiple rollouts are needed to make proper use of the sampling in this method. Moreover, a sufficiently low discretization step size in the action space is required to return good results.

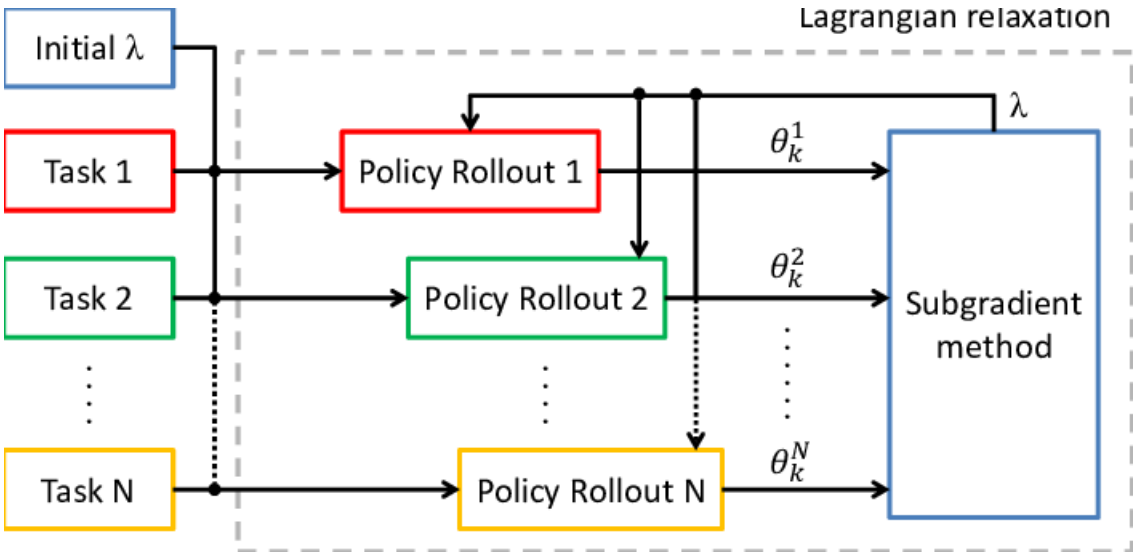

Figure 2.2: AODB algorithm block scheme [2].

2.7. Conclusion

In this chapter it has been explained what a POMDP is and how this framework can be used to formulate our RRM problem. It was then described how this POMDP formulation could be solved with the AODB algorithm using Lagrangian relaxation and policy rollout. The main goal in this thesis is to build on the work done in [2] and construct an algorithm that is less computationally intense, while still providing good results compared to the AODB algorithm. In chapter 3 different POMDP solution methods will be explored.

3

POMDP solutions in literature

This section will discuss different solutions methods that can be found in literature for solving POMDP problems. The solution methods are mostly those that followed from [22–24]. As described in section 2.2, these solution methods should provide a policy that maps an action to a belief state, such that the value function is maximized. In order words, for each belief state the action needs to be found that maximizes its value function:

$$A^*(\boldsymbol{b}_k) = \underset{\boldsymbol{a}_k \in \mathcal{A}}{\arg\max}\, R(\boldsymbol{b}_k, A^\pi(\boldsymbol{b}_k)) + \mathbb{E}\,[V^\pi_{H-1}(\boldsymbol{B}^\pi_{k+1})|\boldsymbol{B}_k = \boldsymbol{b}_k], \tag{3.1}$$

where A^* is the optimal mapping from the belief space to the action space and $\mathcal{A}$ is the action space. However, with growing state and action space, the optimal solution of a POMDP becomes increasingly intractable. Therefore, real-time approaches inevitably require approximate POMDP solution methods to be applied. Following the framework presented in [3, 23], these approximate solution methods can be divided into four classes:

- Policy Function Approximation methods,
- Cost Function Approximation methods,
- Value Function Approximation methods,
- Direct Lookahead Approximation methods.

It is shown graphically in Fig. 3.1 which part of the optimal policy function the first three of these classes approximate. In the next sections, each of these approximate solution methods will be discussed.

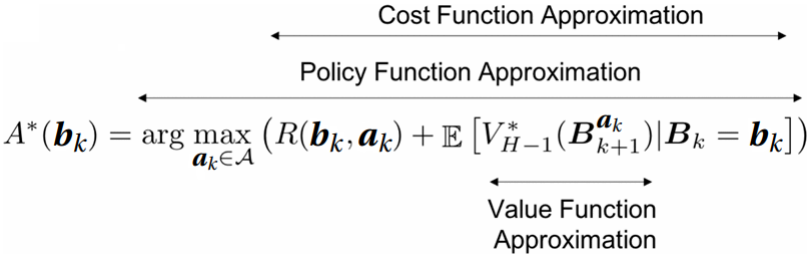

Figure 3.1: Different function approximation classes for the optimal policy function. [3]

3.1. Policy function approximations

In policy function approximations (PFA's) the policy function is parameterized in a way that the policy function depends on some parameter vector $\boldsymbol{\theta}$. The chosen action now follows directly from the policy function that we approximate:

$$A^{f,\boldsymbol{\theta}}(\boldsymbol{b}_k) = f(\boldsymbol{b}_k;\boldsymbol{\theta}). \tag{3.2}$$

For example in portfolio optimization this policy function could be to sell a certain stock if the price goes above a threshold θ_{sell} and to buy this stock if it goes below a threshold θ_{buy}. The approach of PFA is for example used in [25] and [15]. For a solution method involving PFA to be applied in our RRM problem one would need to find some suitable (parameterized) heuristic function. It seems difficult to come up with functions that could fully exploit the information known about the objects and give solutions that yield good results over a given time rather than a single time instance. The implementations in [25] and [15] were simplified such that the functions used can no longer be extended to the multi-target tracking scenario considered in this thesis.

3.2. Cost function approximations

In cost function approximations (CFA's) the cost function needs to be parameterized based on some parameter vector $\boldsymbol{\theta}$. The policy in this case is to take the action that results in the highest value of the reward function:

$$A^{\pi,\boldsymbol{\theta}} = \underset{\boldsymbol{a}_k \in \mathcal{A}}{\arg\max}[\tilde{r}^{\pi}(\boldsymbol{b}_k, \boldsymbol{a}_k;\boldsymbol{\theta})]. \tag{3.3}$$

This class of POMDP solution methods is discussed in detail in [26]. For the RRM problem, it is considered important to evaluate how the actions taken now affect the future. For example, to deal with changes to the system that are known in advance. Generally, in CFA solution methods the impact of a decision now on the future is not considered.

3.3. Value function approximations

By using Value Function Approximations (VFA's), the value function term from equation 3.1, which encapsulates the future rewards can be replaced by an approximation. The equation now becomes:

$$A(\boldsymbol{b}_k) = \underset{\boldsymbol{a}_k \in \mathcal{A}}{\arg\max} R(\boldsymbol{b}_k, A^{\pi}(\boldsymbol{b}_k)) + \mathbb{E}\,[\tilde{V}^{\pi}_{H-1}(\boldsymbol{B}^{\pi}_{k+1})|\boldsymbol{B}_k = \boldsymbol{b}_k]. \tag{3.4}$$

The objective becomes to find a good approximation $\tilde{V}^{\pi}_{H-1}(\boldsymbol{B}_{k+1})$ of the value function. Approximating value functions is a large part in the fields of approximate dynamic programming [27] and Reinforcement Learning [28]. There is a lot of literature available where POMDP solutions belonging to the VFA class are used, e.g. [22, 24, 29–31]. It seems challenging to use a VFA solution method for our RRM problem. The value of all belief states would need to be approximated. In the RRM problem considered in this thesis, the state of the system consists of the two-dimensional position and velocity of the objects. Each of these state elements can assume a lot of different values. Furthermore, there could be any reasonable number of targets to be tracked. Each of these reasons contribute to a very large belief state space. It seems infeasible to approximate the value functions for all belief state spaces using VFA-based solution methods.

3.4. Direct Lookahead Approximations

The last of the four classes of POMDP solution established in [23] is the class of Direct Lookahead Approximations (DLA). Here an approximate lookahead model of the system needs to be made. Using this lookahead model the future evolution and reward of the system can be approximated following some actions that we take now. In this manner, the reward of taking a sequence of actions

Table 3.1: Evaluation of the classes of POMDP solution methods.

Characteristic	PFA	CFA	VFA	DLA
Feasible to find solution	+	+	-	+
Performance (future predicting ability)	-	-	+	+

can be evaluated. Finally, the action needs to be found for which the future reward is the largest. This action can then be applied to the system.

3.5. Evaluation

In Table 3.1 the discussed classes of POMDP solution methods are compared with respect to their ability to be applied to the RRM problem. As stated earlier, the RRM problem that is considered in this thesis has a state space that is too large for VFA solution methods to be implemented. PFA solution methods do not appear to be suited enough to provide a good performance by taking into account all information known to the system. In CFA solution methods the impact that taking actions has on the future is not considered, while this is a goal for our RRM solution. Based on these reasons, in the remainder of this chapter different DLA solution methods will be investigated further.

3.5.1. Rollout policies

Rollout policies predict the future in a state following all policies $\tilde{\pi} \in \tilde{\Pi}$, where $\tilde{\Pi}$ is a restricted set of policies. Afterwards, the rewards obtained by following each of the policies can be computed. Then, the policy that led to the highest reward is applied. This method is applied in [2] in combination with Monte Carlo sampling, where for each policy multiple samples are evaluated and averaged. This was described in detail in section 2.6.2.

3.5.2. Monte Carlo tree-search

Another approach is to use a Monte Carlo tree-search (MCTS). In this solution method the goal is to model the entire system by a decision tree and find optimal decisions to take at each decision node. Each decision node represents a belief state in the system. The following process must be repeated for a sufficient number of times:

1. Select an action out of a decision node (which represents a state).
2. Expand the tree if the resulting observation results in a node not already in the tree.
3. Perform rollout policy to determine the value of the node that was reached.
4. Go backwards through the tree to update the value of being at each node.

The problem is that each belief state in the system needs to be reached and at each of these belief states all actions must be evaluated. In [23] it is suggested that MCTS is feasible for problems only when there are a few dozen actions per state. Consequently, this solution method is useful only for problems where the state space and the action space are small.

3.5.3. Deterministic Lookahead

Deterministic Lookahead methods use an approximate lookahead model of the system to predict the effect that actions taken now have on the future. These methods are often referred to as Model Predictive Control (MPC). The future evolution of the system for some sequence of chosen actions is predicted using the lookahead model. A cost function is then used to compute the resulting cost of taking these actions at each future time-step until some horizon. The sequence of actions resulting

in the lowest value for the cost function is then considered to be the solution of the POMPD at that time-step. This method can in fact also be considered a hybrid CFA solution method using a deterministic lookahead, negating the downside of CFA solution methods, which is that they generally do not take into account the future effect of actions. An example of such a solution method for an energy storage problem can be found in [32]. In contrast to computing different rollouts using Monte Carlo sampling now the most likely future evolution based on the underlying model is considered, rather than the future evolution of some randomized samples.

3.6. Conclusion

In this chapter different POMDP solution methods are categorized in 4 classes. These classes were then explained and their applicability to the RRM problem was compared. It was found in section 3.5 that a DLA solution method would be most suited to solve the RRM problem. Subsequently, different DLA solution methods were discussed. MCTS was determined to be unfit for the RRM problem as the state space and action space must be small to apply this solution method. While policy rollout methods showed promising results, the computation time was quite large. The deterministic lookahead approach, often referred to as model predictive control (MPC), has been shown to be an approximation of policy rollout [33]. Due to its ability to provide non-myopic solutions and due to the goal to lower the computation time compared to policy rollout, the possibility of employing an MPC solution method to the RRM problem will be explored in this thesis. As MPC can be seen as an approximation of policy rollout, it will be compared to the Policy Rollout method implementation discussed in [2]. Based on their performances in terms of computational time and cost in different scenarios it will be investigated if MPC could be a viable alternative to policy rollout methods for RRM problems. In chapter 4 the MPC solution method to the RRM problem will be described and evaluated.

4

Proposed approach

The proposed POMDP solution method for the RRM problem in this thesis is model predictive control (MPC). MPC is a receding horizon approach meaning that the prediction horizon is shifted forward after every iteration. The basic operating principle can be summarized as follows, where H indicates the prediction horizon, which in this thesis is always assumed to be equal to the control horizon:

1. Minimize the cost function $c(\boldsymbol{a}_t, \boldsymbol{s}_t)$ over prediction horizon:

$$\boldsymbol{a}_t = \arg\min_{\boldsymbol{a}_t} \sum_{t=k}^{k+H} c(\boldsymbol{a}_t, \boldsymbol{s}_t), \tag{4.1}$$

   where $\boldsymbol{a}_t$ are the actions taken at time t and $\boldsymbol{s}_t$ is the state at time t. The output $\boldsymbol{a}_t$ of the minimization problem gives the approximately optimal actions to take for the upcoming H time-steps.

2. Apply only the actions corresponding to the upcoming time-step of the computed action sequence, i.e. $\boldsymbol{a}_k$.

3. Repeat 1 and 2 at next time-step $k+1$.

In the next section it will be explained how the MPC framework is used to solve the RRM problem defined in chapter 2.

4.1. Model Predictive Control

Recall the optimization problem of N different sub-problems obtained using Lagrangian relaxation:

$$\min_{\boldsymbol{\tau},\boldsymbol{T}} \quad \sum_{t=k}^{k+H} c(\tau_t^n, T_t^n, s_t^n) + \lambda \frac{\tau_t^n}{T_t^n}. \tag{4.2}$$

Each of these minimization problems are solved using MPC for some value of the Lagrange multiplier λ. Next, it is evaluated if the imposed constraint on the available budget is approximately met. A budget distribution needs to be found where the total assigned budget is close to B_{max}. If the total assigned budget is too large, the determined actions cannot be applied. If the total assigned budget is too small, the available radar resources are not sufficiently exploited. Therefore, a value

of λ must be found for which (4.3) is met, where ε is some small number indicating the tolerance of the constraint:

$$|\sum_{n=1}^{N}(\frac{\tau_t^n}{T_t^n}) - B_{max}| \leq \varepsilon. \tag{4.3}$$

It will be discussed in section 4.2 how this λ is found using golden section search.

4.2. Golden Section Search

Recall that the goal is to find the Lagrange multiplier λ such that 4.3 is met. In Fig. 4.1 values of $\sum_{n=1}^{N}(\frac{\tau_n^t}{T_n^t}) - B_{max}$ for different Lagrange multipliers λ are plotted. The absolute value is plotted for different values of the Lagrange multiplier in Fig. 4.2.

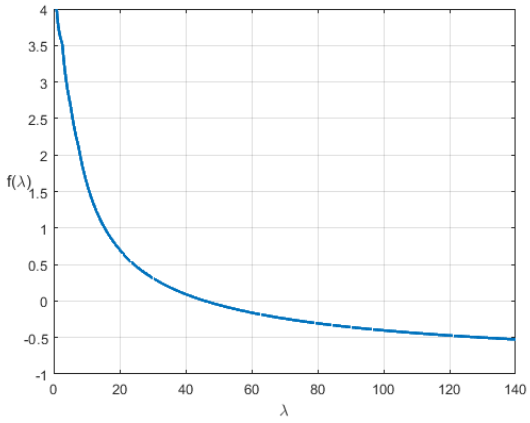

Figure 4.1: Difference between available budget and budget needed resulting from optimization.

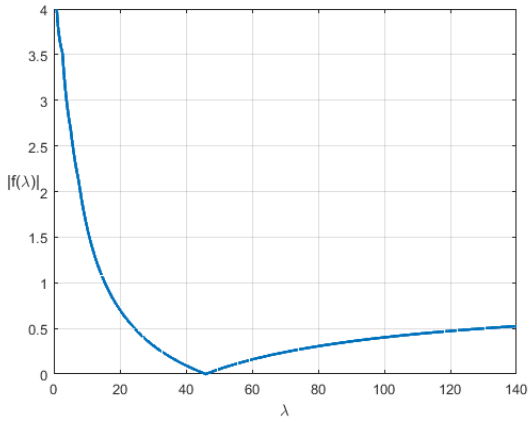

Figure 4.2: Absolute value of difference between available budget and budget needed resulting from optimization.

From this figure it can be seen that this forms a unimodal function, i.e. a function monotonically decreasing for $x \leq x_{min}$ and monotonically increasing for $x \geq x_{min}$. Based on this, it is known that its minimum can be found using golden section search [34]. The standard golden section search method, as described in for example [35], is extended due to the fact that initially the lower and upper bounds of λ are unknown. To find these bounds, λ is increased after every time-step k. As it is known that the value of λ must ensure that (4.3) is met, our initial values for the upper and lower bounds are those values of λ_k and λ_{k+1} for which there is a change in sign when going from $f(\lambda_k)$ to $f(\lambda_{k+1})$. Here a function evaluation $f(\lambda)$ is defined as follows:

$$f(\lambda) = \sum_{n=1}^{N}\frac{\tau_t^n}{T_t^n} - B_{max}, \tag{4.4}$$

where τ_t^n and T_t^n result from solving the sub-problems in (2.23) using λ. These lower and upper bounds will be referred to as x_L and x_U. Once these bounds are found the next step to find λ is to perform function evaluations of values of λ between x_U and x_L. An efficient way of choosing these intermediate points is using the golden ratio conjugate ($r = \frac{\sqrt{5}-1}{2} \approx 0.618$). Initially, the two intermediate points $x_1 = x_L + r(x_U - x_L)$ and $x_2 = x_U - r(x_U - x_L)$ are evaluated. Then, based on the values $f(x_1)$ and $f(x_2)$ at those points a new intermediate point is chosen and x_L or x_U is shifted to x_1 or x_2 respectively after which the function is evaluated at the newly chosen point. This procedure is repeated until a point is found that meets the criterion set in (4.3). The whole search method is summarized in Algorithm 2.

Algorithm 2: Finding the Lagrange multiplier using golden section search

Step 1 Starting from $\lambda = 0$, increase λ and compute f(λ) until f(λ) becomes negative
Step 2 This λ becomes x_U while the previous λ becomes x_L
$x_1 = x_U - r(x_U - x_L)$
$x_2 = x_L + r(x_U - x_L)$
Compute f(x_1) and f(x_2)
Step 3;
while $f(x_1) \leq \varepsilon \cap f(x_2) \leq \varepsilon$ **do**
  if $f(x_1) \leq f(x_2)$ **then**
    $x_U = x_2$
    $x_2 = x_1$
    $x_1 = x_U - r(x_U - x_L)$
    Compute $f(x_1)$
  else
    $x_L = x_1$
    $x_1 = x_2$
    $x_2 = x_L + r(x_U - x_L)$
    Compute $f(x_2)$
  end
end

As the value of λ_t that results from this process is in general close to that at the next budget update interval λ_{t+T}, it is better to not initialize λ at 0 again. Instead, the initial value of λ can be set to the previously determined λ to reduce the number of function evaluations. As this initial λ might already be larger than the optimal λ, it must first be checked if $f(\lambda)$ is increasing when λ increases to determine the search direction of the first step of the algorithm.

4.3. Scheduling conflict

Consider the situation where 2 objects are tracked and the optimization problem results in the following values for τ and T, which is visually represented in figure 4.3:

Table 4.1: Example optimized actions.

$\tau^1[s]$	$\tau^2[s]$	$T^1[s]$	$T^2[s]$
1	1.5	2	3

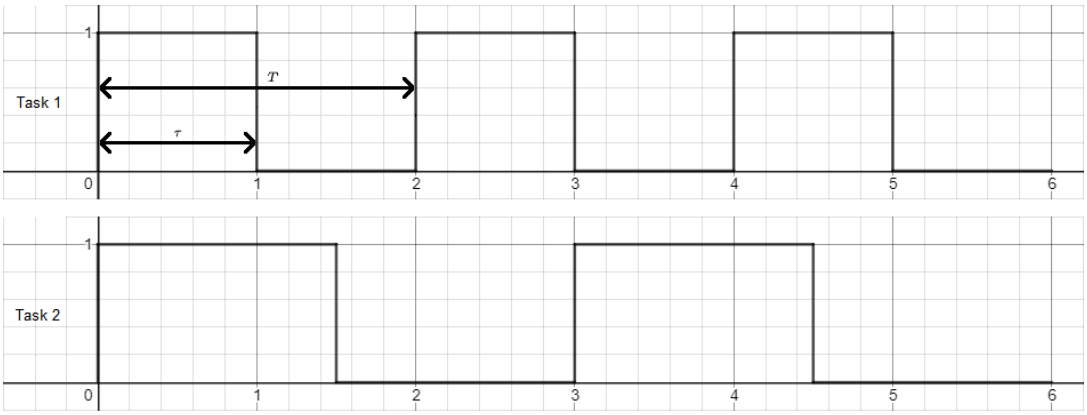

Figure 4.3: Example budget distribution that leads to scheduling issues.

Assuming that $B_{max} = 1$, with these values the constraint imposed in (4.3) is satisfied. It can, however, clearly be seen from the figure that the tasks cannot be executed while maintaining the computed values for $\boldsymbol{\tau}$ and $\boldsymbol{T}$, considering that the radar can only take measurements of a single object at once. Evidently, to ensure that the matrices $\boldsymbol{\tau}$ and $\boldsymbol{T}$ that result from an optimization step can be directly translated into actions for the radar, some more advanced constraints are needed. To avoid such scheduling issues, for the remainder of this thesis it is assumed that the revisit time is equal for all the tasks. It can easily be confirmed that when the revisit time for all objects are the same, no scheduling issues will arise as long as the constraint listed in (4.3) is satisfied.
Furthermore, as it is assumed that the budget update interval is kept constant, for some values of the revisit time it may be the case that not all objects that are scheduled have been executed when a new budget update is issued, leading to some tasks being dropped. To prevent this, the value of the revisit time of the tasks needs to be a divisor of the budget update interval. For instance for a budget update interval of 5 s, $\frac{5}{T}$ should return a natural number: $\frac{5}{T} \in \mathbb{N}$.

4.4. Solution overview

Finally, it is summarized in this section using Fig. 4.4 how the different parts of the solution methods are interconnected for a time-step in which a budget update takes place. The inputs are the object parameters and the initial λ. Based on this, the sub-problems for the N different targets can be

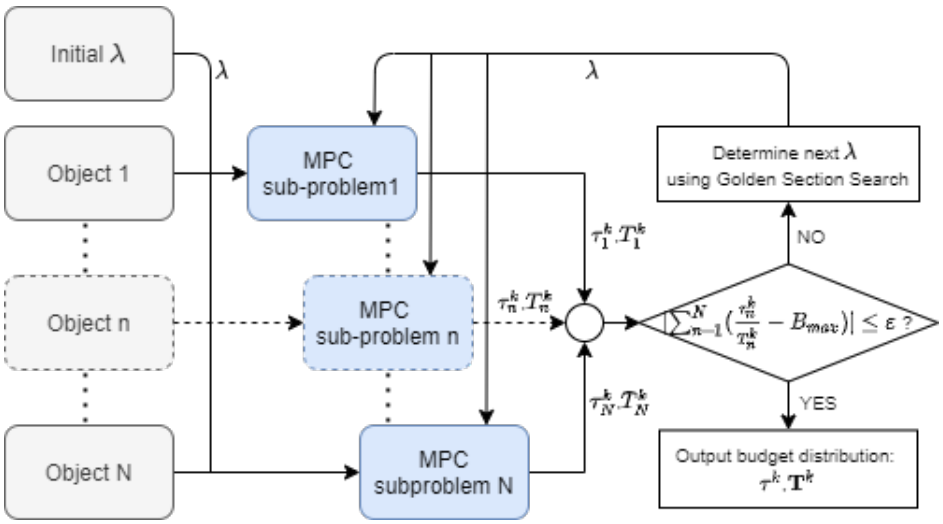

Figure 4.4: Overview of solution method.

solved using MPC. With these found solutions it is checked if the budget constraint in (4.3) is met. If this is the case, the solution $\{\boldsymbol{\tau}^k, \boldsymbol{T}^k\}$ is found. The entries of $\{\boldsymbol{\tau}^k, \boldsymbol{T}^k\}$ corresponding to the next time-step are then applied to the radar system. If this is not the case, the golden section search algorithm is used to find new values for λ until the budget constraint is met. At the next budget update interval, the described process in this section is repeated. Comparing the MPC solution method with the policy rollout solution method introduced in [2], there are a couple of key differences. Firstly, in the case of policy rollout the action space is discretized, while for the MPC solution method the action space can be continuous. Moreover, there is a fundamental difference in the way the future is predicted in order to minimize over the action space. Using policy rollout, for every action the future is predicted by sampling the random variables, such as maneuverability and measurement noise, a number of times, after which the cost is averaged over these samples. In the MPC solution method, for these random variables the most likely value is chosen based on the underlying model, and the cost of only one future sample needs to be computed. Lastly, in the policy rollout method, some base policy needs to be assumed and you can practically only optimize the first action to take

of the horizon. In the MPC method, you can compute different actions for each measurement step. Note that theoretically the same could be done using the policy rollout method, but the action space and consequently the computation time would increase exponentially with the number of different actions that you optimize at once.

4.5. Results and Simulations

In the remainder of this chapter, the proposed solution method will be compared to the method of policy rollout introduced in [2]. Using MATLAB simulations, both solution methods will be tasked with computing the budget distributions in four different dynamic tracking scenarios. Furthermore, the importance of Lagrangian relaxation and the method of finding the Lagrange multiplier will be investigated. For the reference measurement parameters the values listed in Table 2.1 will be used.

4.6. Simulation Scenario A

Simulation scenario A consists of targets moving in a straight line at a constant velocity, as can be seen in Fig. 4.5 for 5 targets. Scenarios with varying numbers of targets moving in a straight line at a constant velocity are used in the following sections for the following purposes:

- Compare the effect that certain events have on the budget distribution using MPC and policy rollout (subsection 4.6.1),
- compare the execution time and realized cost using MPC and policy rollout (subsections 4.6.2 and 4.6.3),
- investigate the importance of Lagrangian relaxation in the solution method (subsection 4.6.4),
- compare the implementation of the golden section search algorithm with the subgradient method for finding λ (subsection 4.6.5).

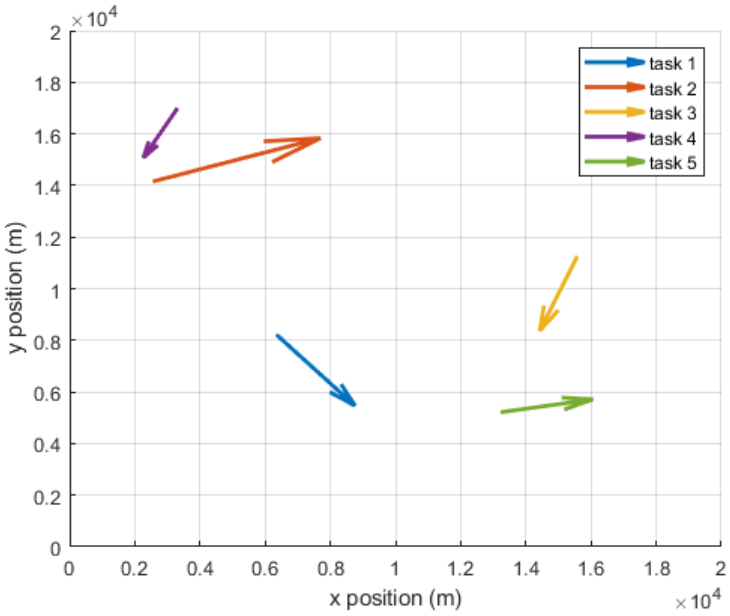

Figure 4.5: Trajectories of the 5 objects to be tracked for a scenario of 100 seconds.

4.6.1. Budget distribution

In this example situation, some events took place during the simulation time so that their effect on the budget distribution could be evaluated:

- **t = 20 s**: A new object (task 5) needs to be tracked.
- **t = 60 s**: Total available budget decreases from 1.0 to 0.9.
- **t = 90 s**: Maneuverability of task 1 increases.

Table 4.2: Simulation parameters scenario A.

Reference parameter	**Value**
Maneuverability noise variance (σ_w^2)	2.5 $(\frac{m}{s^2})^2$
Radar cross section (RCS)	10 m^2
Prediction horizon length (H)	15
Number of rollouts	10
Simulation update interval	5 s
Budget precision (ε)	0.002

For the simulation the parameters listed in Table 4.2 were used. Fig. 4.6 and 4.7 show the budget distributions for the example scenario shown in Fig. 4.5 using the MPC and the policy rollout algorithm, respectively. It can be seen that objects located further away from the radar are allocated a

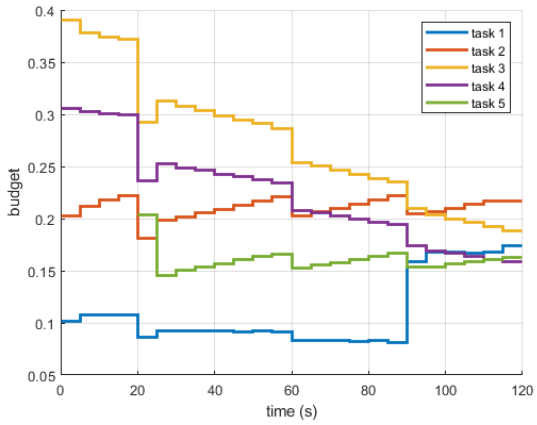

Figure 4.6: Budget allocation of the 5 objects over the simulation time obtained using MPC.

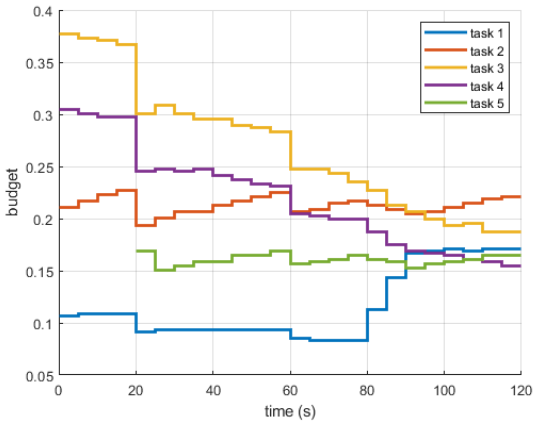

Figure 4.7: Budget allocation of the 5 objects over the simulation time obtained using Policy Rollout.

larger amount of the budget than those closer to the radar. This is due to the way the cost function is constructed. Measurements of objects further away will have a smaller signal-to-noise ratio and this is compensated for by increasing the budget available for tracking these objects. This behaviour is specific to the cost function choice and might not always be desired, so given the demands of the user, a different cost function can be constructed. Furthermore, the figure reflects the changes to the system at the set time-steps. At t = 20 s, a new object needs to be tracked and some budget is made available for this task. At t = 60 s, the total available budget decreases and the budgets of all tasks decrease. At t = 90 s, the maneuverability noise variance of the first task increases, resulting in the need to take better measurements of this object and therefore increasing the budget of task 1. When comparing the budget distributions from the policy rollout and MPC, it can be seen that the

budgets are mostly the same. The main difference is seen at the change in maneuverability at t = 90 s. It can be seen that the effect of this change shows only after 90 s in the case of MPC, while in the case of policy rollout the effect of this change shows already at 80 s. This is due to the fact that in the policy rollout optimization the same action is chosen for the full prediction horizon, while for MPC this is not necessarily the case. As a horizon of 15 s is used, at the budget update that takes place at 80 s this change in maneuverability already needs to be taken into account, and thus the actions are already slightly influenced by this change.

4.6.2. Execution Time Comparison

A goal of employing MPC was to lower the computational complexity compared to the policy rollout algorithm. This is evaluated by comparing the execution time of an average budget update of both algorithms. Here scenarios with linearly moving targets similar to scenario A are used with varying numbers of moving objects. To compare the execution time as well as the realized cost in the next section, a prediction horizon of 5 s is assumed. In Fig. 4.8 the results from running each simulation 3 times and averaging the execution times are shown. It can be seen that the execution times when using MPC are significantly lower than when policy rollout is used.

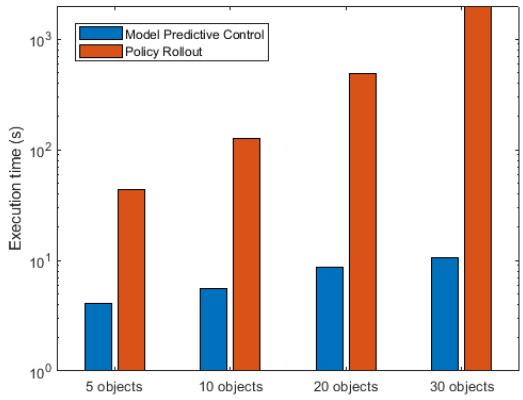

Figure 4.8: Comparison between execution times of policy rollout and MPC algorithms.

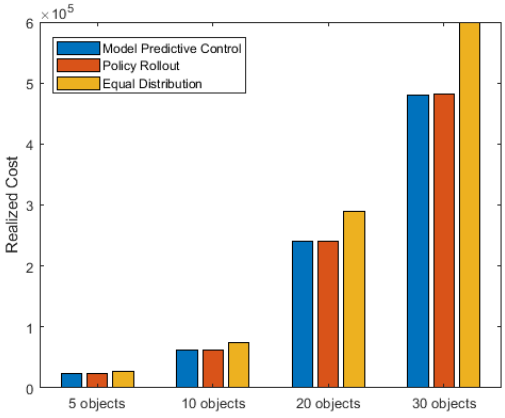

Figure 4.9: Comparison between realized costs of equal distribution, policy rollout and MPC.

4.6.3. Realized Cost Comparison

It is important that the performance of the resulting budget distribution is not degraded compared to the policy rollout case. To investigate this their realized costs were compared using the same scenarios as for the execution time comparison. The realized cost is defined as the sum of the evaluated cost function at every time-step during the simulation. In Fig. 4.9 the results from running each simulation 3 times and averaging the realised costs are shown. For comparison, the realised cost of using an equal distribution, i.e. allocating the same budget to all targets at every time-step, is also included. From the figure it can be seen that MPC and Policy Rollout have similar performances when looking at realized costs, while both outperform an equal distribution scheme.

4.6.4. Lagrangian Relaxation

Next, the importance of using Lagrangian Relaxation was evaluated. To do this, again simulations were ran using different numbers of objects. To compute the budget distributions for these tasks, two solution methods were used. The first solution method was the one described in this chapter, while the second solution method used MPC, but did not split the problem into sub-problems us-

ing Lagrangian relaxation. Instead, a single large optimization problem was solved at every budget update. The simulations using both algorithms were ran using 6 different values for the prediction horizon. In Fig. 4.10-4.13 the results are shown. Here it can be seen that for small values for the horizon and for the number of tasks, using Lagrangian Relaxation increased the computation time needed for the budget update. However, when the number of objects or the prediction horizon increases, the added value of Lagrangian Relaxation becomes clear. For tracking more than 20 objects, Lagrangian Relaxation greatly reduces the computation time needed for the budget update.

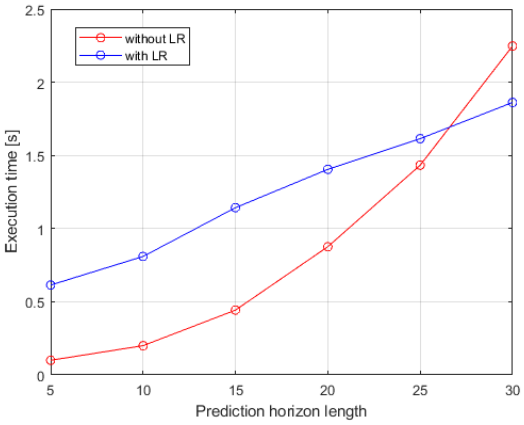

Figure 4.10: Execution time for distributing the budget between 5 objects with and without using LR.

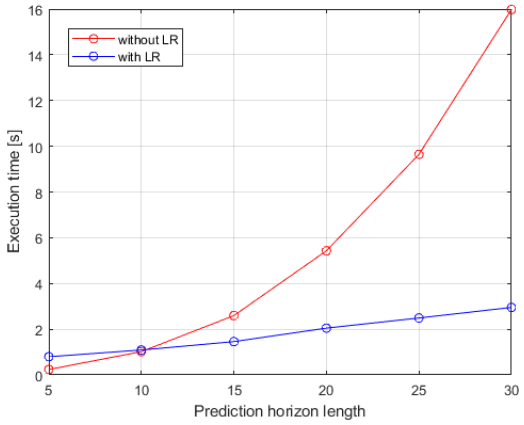

Figure 4.11: Execution time for distributing the budget between 10 objects with and without using LR.

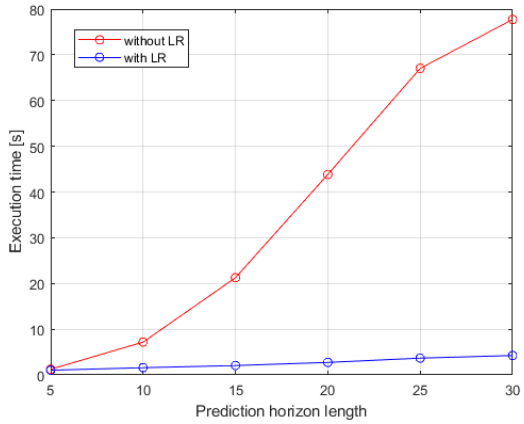

Figure 4.12: Execution time for distributing the budget between 20 objects with and without using LR.

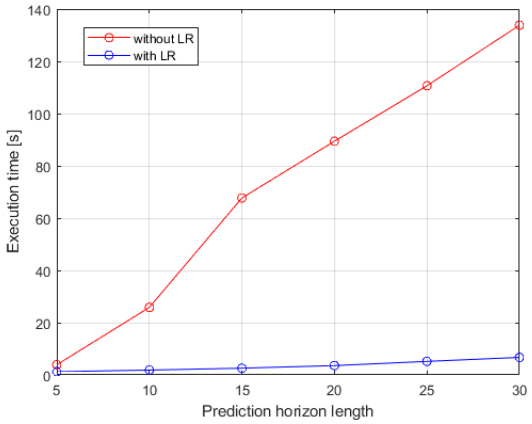

Figure 4.13: Execution time for distributing the budget between 30 objects with and without using LR.

4.6.5. Converging to the Lagrange multiplier

In this section the performance of the golden section search algorithm was investigated. To do this, it was compared to the subgradient method that was implemented as described in Algorithm 1. For both algorithms it was evaluated how many function evaluations $f(\lambda)$ were needed until convergence to some allowed tolerance ε took place. The number of function evaluations until convergence was averaged over 20 budget updates. The results can be seen in Fig. 4.14. For small numbers of targets, both methods show similar performance. When the amount of tasks is increased, the

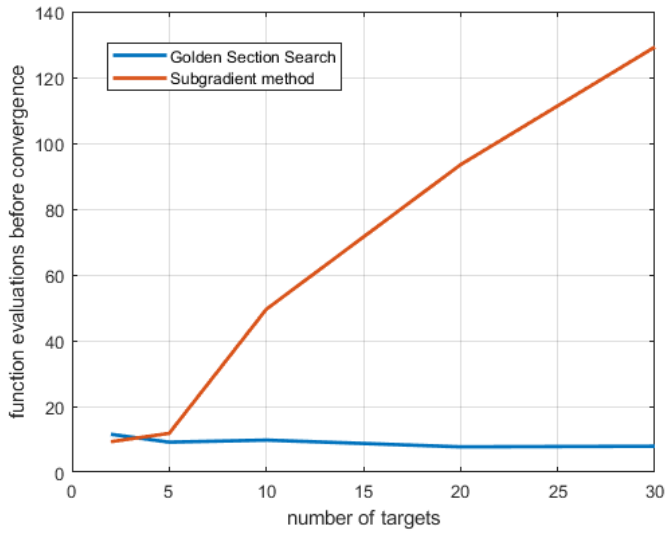

Figure 4.14: Golden section search vs subgradient method comparison.

required function evaluations of the subgradient method increases, while for the golden section search it stays nearly constant. Another downside of the subgradient method is the need to manually select a step size. The number of function evaluations needed depends on this step size and an optimal step size differs per problem (e.g. as the number of targets go up, typically a larger step size is needed). For the subgradient method in this simulation, the initial step size was set to be nearly optimal for all the different numbers of objects through trial and error.

4.7. Simulation Scenario B

Simulation scenario B consists of a static and a moving target. The moving target is first moving straight before making a turn towards the radar sensor and then continuing a linear trajectory. This trajectory is shown in Fig. 4.15. For the simulation the parameters listed in Table 4.3 were used.

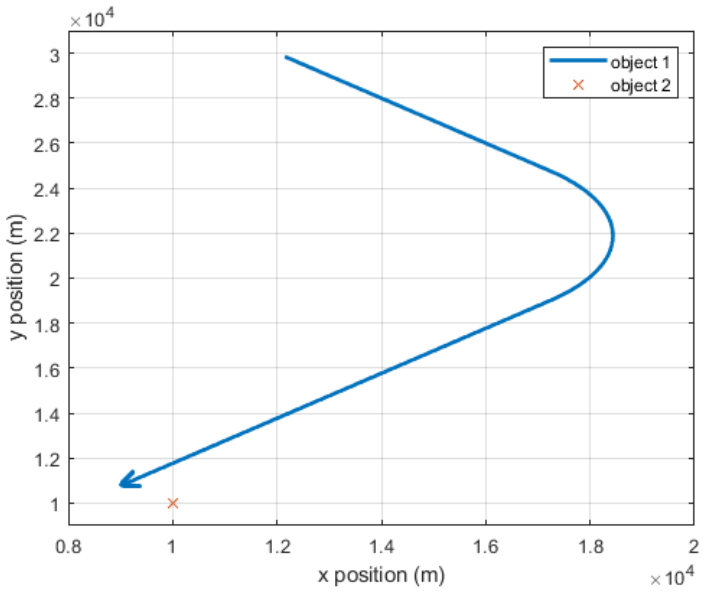

Figure 4.15: Trajectories of the objects of scenario B.

Table 4.3: Simulation parameters scenario B.

Reference parameter	**Value**
Maneuverability noise variance object 1(σ^2_{w1})	2.5 $(\frac{m}{s^2})^2$
Maneuverability noise variance object 2(σ^2_{w2})	0.1 $(\frac{m}{s^2})^2$
Radar cross section (RCS)	10 m^2
Prediction horizon length (H)	5
Number of rollouts	10
Simulation update interval	5 s
Budget precision (ε)	0.002

Fig. 4.16 and 4.17 show the resulting budget distributions of this scenario for MPC and policy rollout respectively. In this case the budget distributions again are very similar. Furthermore, the resulting realised costs over the simulation time are again very close to each other.

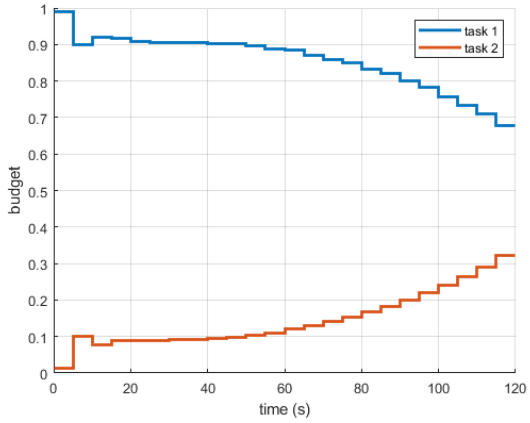

Figure 4.16: Budget allocation of the 2 objects described in scenario B using MPC.

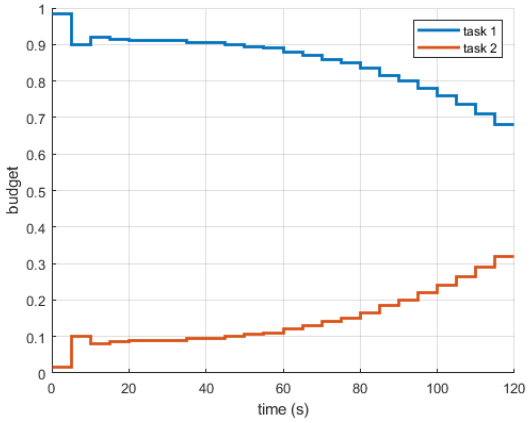

Figure 4.17: Budget allocation of the 2 objects described in scenario B using policy rollout.

4.7.1. Simulation Scenario C

Table 4.4: Simulation parameters scenario C.

Reference parameter	**Value**
Maneuverability noise variance object 1 (σ^2_{w1})	2.5 $(\frac{m}{s^2})^2$
Maneuverability noise variance object 2 (σ^2_{w2})	2.5 $(\frac{m}{s^2})^2$
Radar cross section (RCS)	10 m^2
Prediction horizon length (H)	5
Number of rollouts	10
Simulation update interval	5 s
Budget precision (ε)	0.002

Simulation scenario C consists again of two targets. The first target makes an unexpected maneuver at t = 15 s. In the scenario an area

$$\begin{aligned} 15000\ [\text{m}] \leq x_k^n \leq 20000\ [\text{m}] \\ 15000\ [\text{m}] \leq y_k^n \leq 20000\ [\text{m}] \end{aligned} \tag{4.5}$$

exists in which the quality of the measurements is negatively affected due to e.g. weather conditions. If an object enters this area at least 80% of the budget is needed or else the radar loses track of the target, which is known to the radar system. This trajectory can be seen in Fig. 4.18. The second target

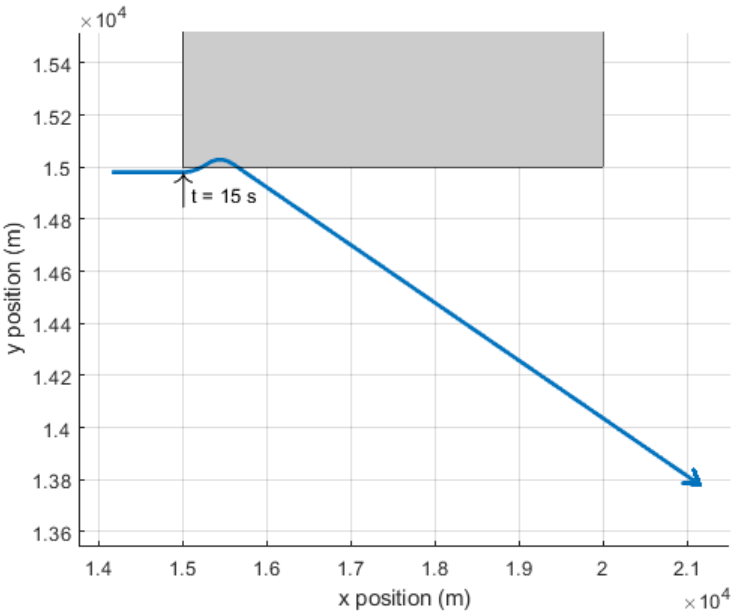

Figure 4.18: Trajectory of the objects from scenario C.

makes the same maneuver but in the negative x and y quadrant, where there is no such grey area. For the simulation the parameters listed in Table 4.4 were used. The resulting budget distributions for MPC and policy rollout can be seen in Fig. 4.19 and 4.20. From these figures it can be seen that policy rollout adapts earlier to the unexpected maneuver, as in some of the rollouts object 1 maneuvers into the grey area between t = 15 s and t = 20 s, while the MPC approach at that point still assumes that the target continues moving in the same direction, avoiding the grey area. As a result, in the MPC case the track would be lost in this scenario. This downside of MPC only assuming a perfect system model can be negated by using a more robust MPC scheme, similar to [36], where different disturbances in the state are considered. However, this will lead to a more computationally intense control law.

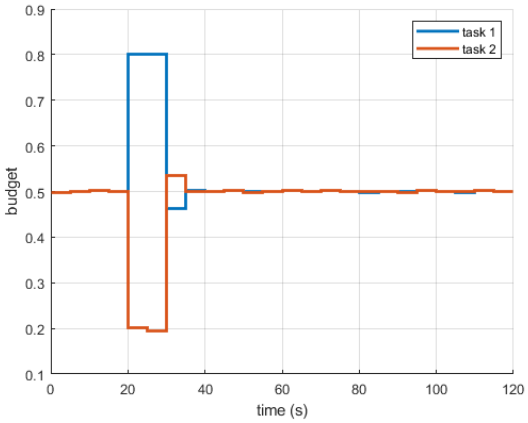

Figure 4.19: Budget allocation of the 2 objects described in scenario C using MPC.

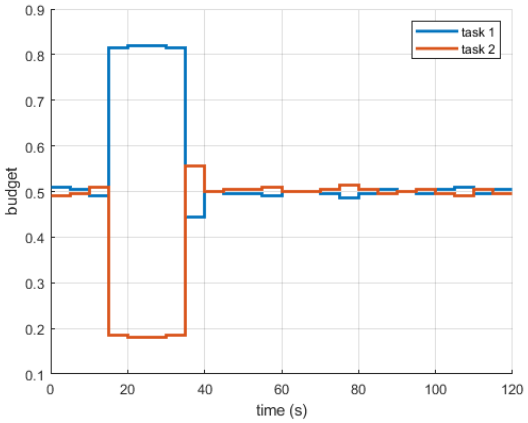

Figure 4.20: Budget allocation of the 2 objects described in scenario C using policy rollout.

4.8. Simulation scenario D

In the final simulation of this chapter the importance of the prediction horizon H is investigated. This horizon refers to how far ahead the lookahead model predicts at every optimization step. Events taking place in the future can only be taken into account if the prediction horizon is sufficiently large, such that the event occurs within the prediction window. To illustrate this the scenario in Fig. 4.21 was constructed.

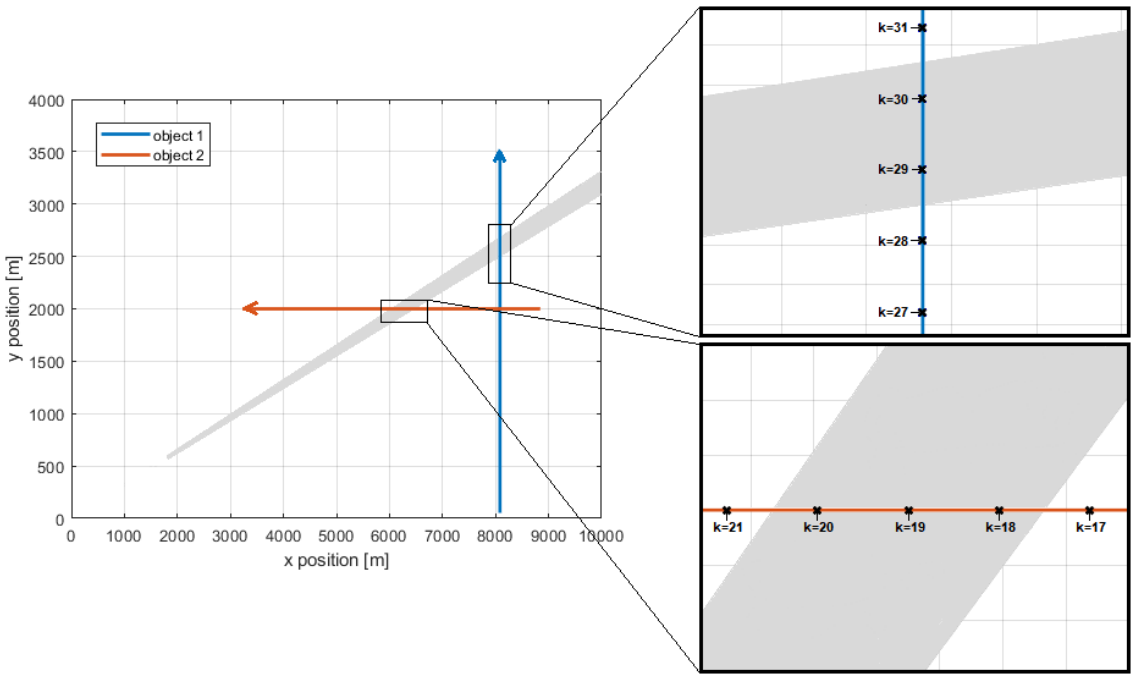

Figure 4.21: Trajectory of the objects from scenario D.

In this scenario there is a specific region where, similar as in scenario C, again at least 80% of the budget is needed to not lose track of the target. This region is defined as

$$0.3 \text{ [rad]} \leq \theta \leq 0.32 \text{ [rad]} \tag{4.6}$$

and can for example be caused by some object blocking this azimuth interval, negatively affecting the measurements. In the figure it can be seen that both objects traverse the grey area. In this simulation T is fixed to 1 s so every time-step will be 1 s. Object 1 is in this area during the measurements taken at time-steps 29 and 30, while object 2 traverses this area during time-steps 18, 19 and 20. The budget updates take place at every 5 time-steps, so at k=0, 5, 10, etc. At the budget update at time-step 15, at least a prediction horizon of 3 steps ahead is needed to take into account the expected deteriorated measurements ahead and assign 80% of the budget to tracking object 2. Similarly, at

Table 4.5: Simulation parameters scenario D.

Reference parameter	**Value**
Maneuverability noise variance object 1 (σ^2_{w1})	2.5 $(\frac{m}{s^2})^2$
Maneuverability noise variance object 2 (σ^2_{w2})	2.5 $(\frac{m}{s^2})^2$
Radar cross section (RCS)	10 m^2
Simulation update interval	5 s
Revisit time (T)	1 s
Budget precision (ε)	0.002

the budget update at time-step 25 at least a prediction horizon of 4 is needed for the optimization to take into account that at least 80% of the budget is needed for object 1 at time-steps 29 and 30. The budget distributions for the described trajectory were simulated for prediction horizon values of 1, 2, 3 and 4 using the parameters listed in Table 4.5. The resulting budget distributions can be seen in Fig. 4.22 to 4.25. From Fig. 4.22 and 4.23 it can be seen that for prediction horizons of 1 and 2 the budget distribution did not adapt to either of the two objects entering the grey area. This is a result of this event not taking place within the prediction horizon.

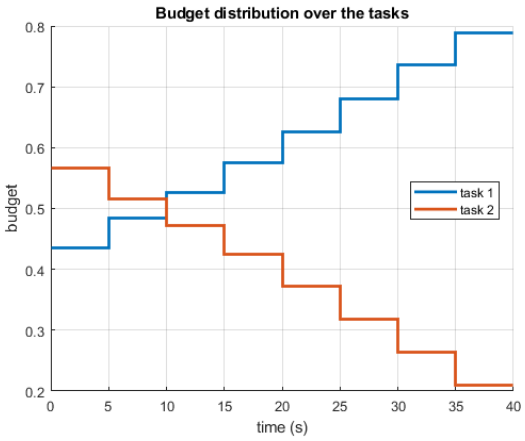

Figure 4.22: Budget distribution of the two objects with a prediction horizon of 1.

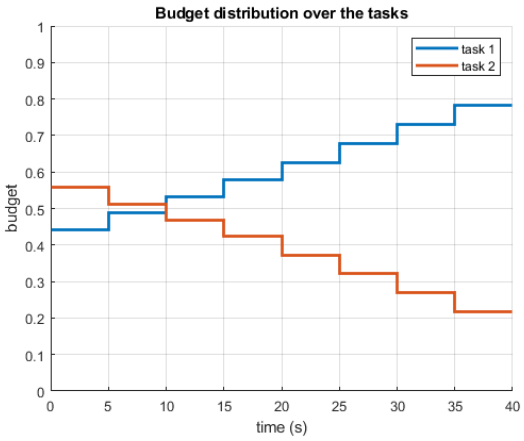

Figure 4.23: Budget distribution of the two objects with a prediction horizon of 2.

In Fig. 4.24 it can be seen that the budget distribution adapts only to object 2 entering the grey area, as the prediction horizon is not large enough to predict object 1 entering the grey area at time-step 29. In the simulation with prediction horizon of 4, the prediction horizon is large enough to predict and adapt to both objects entering the grey area. Consequently, with a prediction horizon of 1 and 2 both tracks would be lost, with a prediction horizon of 3 only the first track would be lost, while with a prediction horizon of (at least) 4, the radar can continue to track both objects.

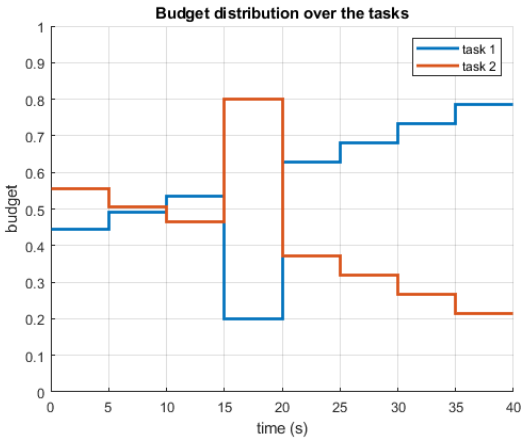

Figure 4.24: Budget distribution of the two objects with a prediction horizon of 3.

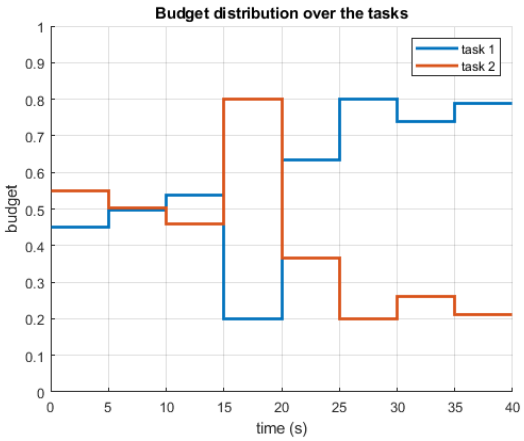

Figure 4.25: Budget distribution of the two objects with a prediction horizon of 4.

4.9. Conclusion

In this chapter the proposed solution method using MPC to the RRM problem has been described. This solution method was then compared to the existing method of policy rollout using four dynamic tracking scenarios. It was shown that the proposed method greatly reduced the computation time needed to provide a very similar budget distribution. The next chapter will provide an extension to the proposed solution method. An IMM filter will be used to improve the tracking performance while the targets are maneuvering.

5

Approach extended with IMM filter

Until now an EKF using a constant velocity (CV) model has been used to track the objects. This greatly limits the tracking capability when tracking objects whose movement does not closely resemble the CV model, resulting in large tracking errors. One way of improving the tracking capability in such cases is by using an Interacting Multiple Model (IMM) filter. Now, instead of by a filter using a single CV model, the objects are tracked using multiple models. In this chapter it is first explained how this IMM filter is implemented for the tracking of targets, following the theory described in [37]. Then, it is explained how the IMM filter is integrated in the MPC optimization problem. Next, the behaviour of the solution method extended with the IMM filter is evaluated. It is investigated if the tracking error is in fact reduced in scenarios where the CV model does not approximate the movement of the objects. Lastly, it is looked at how the integration of the IMM filter affects the budget distribution in a scenario where one of the objects makes a maneuver.

5.1. IMM filter

In theory, one could implement an IMM filter with as many different models describing different types of object movements as desired. Increasing the number of models will increase the number of filters that need to be updated and will let the mixing step, as will become clear later in this section, take more operations to compute. Both these factors will attribute to an increase in the computational load. As the goal here is to investigate how the tracking performance can be increased by using a more advanced filter than an extended Kalman filter and how this will behave in combination with a budget distribution algorithm, it was chosen to implement an IMM filter using only two models: a CV model and a constant turn (CT) model. For the CV model, the same model as described in section 2.3.1 is used:

$$\boldsymbol{s}_{k+1}^{CV} = \begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \dot{x}_{k+1} \\ \dot{y}_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & T_k & 0 \\ 0 & 1 & 0 & T_k \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \boldsymbol{s}_k + \boldsymbol{w}_k, \tag{5.1}$$

where $\boldsymbol{w}_k$ is the maneuverability noise of the target at time-step k. For the CT model, the following model is used to update the state:

$$\boldsymbol{s}_{k+1}^{CT} = \begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \dot{x}_{k+1} \\ \dot{y}_{k+1} \\ \omega_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & \sin(\omega_k T_k)/\omega_k & -(1-\cos(\omega_k T_k))/\omega_k & 0 \\ 0 & 1 & (1-\cos(\omega_k T_k))/\omega_k & \sin(\omega_k T_k)/\omega_k & 0 \\ 0 & 0 & \cos(\omega_k T_k) & -\sin(\omega_k T_k) & 0 \\ 0 & 0 & \sin(\omega_k T_k) & \cos(\omega_k T_k) & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \boldsymbol{s}_k + \boldsymbol{w}_k, \tag{5.2}$$

where the angular velocity ω_k is appended to the state and $\boldsymbol{w}_k$ is again the maneuverability noise of the target at time-step k. Now that the motion model is non-linear in the case of this CT model, the state transition matrix $\boldsymbol{F}$ is now the Jacobian of the state transition function

$$\boldsymbol{F}_k = \left.\frac{\partial f}{\partial \boldsymbol{s}}\right|_{\boldsymbol{s}_k^n}. \tag{5.3}$$

At every time-step a *predict* step is carried out using both models. Then, based on the update step, the likelihood of model j at time-step k can be calculated as follows:

$$L_k^j = \frac{1}{\sqrt{(2\pi)^2 \det(\boldsymbol{S}_k^j)}} \exp \frac{-(\tilde{\boldsymbol{z}}_k^j)^\top (\boldsymbol{S}_k^j)^{-1} \tilde{\boldsymbol{z}}_k^j}{2}, \tag{5.4}$$

where $\boldsymbol{S}_k^j$ is the innovation covariance of model j at time-step k:

$$\boldsymbol{S}_k^j = \boldsymbol{H}_k^j \boldsymbol{P}_{k|k-1} (\boldsymbol{H}_k^j)^\top + \boldsymbol{R}_k^j \tag{5.5}$$

and $\tilde{\boldsymbol{z}}_k$ is the innovation residual of model j at time-step k:

$$\tilde{\boldsymbol{z}}_k^j = \boldsymbol{z}_k - h(\hat{\boldsymbol{s}}_{k|k-1}^j). \tag{5.6}$$

Next, the probability of the target movement following model j can be computed as shown below:

$$p_k^j = \sum_{i=1}^{2} \boldsymbol{P}_{tr}(j,i) \frac{L_k^i p_{k-1}^i}{L_k^1 p_{k-1}^1 + L_k^2 p_{k-1}^2}, \tag{5.7}$$

where

$$\boldsymbol{P}_{tr} = \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{21} \end{bmatrix} = \begin{bmatrix} 0.98 & 0.02 \\ 0.02 & 0.98 \end{bmatrix} \tag{5.8}$$

is the transition probability matrix. This matrix can be seen as a weighing of the previously established model probabilities and the model likelihoods that are calculated using (5.4). Finally, once the model probabilities are known, the predicted state and predicted error covariance matrix can be computed by mixing the predicted state vectors and predicted error covariance matrices based on the model probabilities:

$$\hat{\boldsymbol{s}}_k = \sum_{j=1}^{2} p_k^j \hat{\boldsymbol{s}}_k^j \tag{5.9}$$

and

$$\hat{\boldsymbol{P}}_{k|k} = \sum_{j=1}^{2} p_k^j \hat{\boldsymbol{P}}_{k|k}^j (\hat{\boldsymbol{s}}_k - \hat{\boldsymbol{s}}_k^j)(\hat{\boldsymbol{s}}_k - \hat{\boldsymbol{s}}_k^j)^\top. \tag{5.10}$$

Table 5.1: IMM filtering steps.

Predict stage:	
A priori state estimate:	$\hat{\boldsymbol{s}}^j_{k\|k-1} = \boldsymbol{F}_k \hat{\boldsymbol{s}}^j_{k-1\|k-1}$
A priori estimated error covariance:	$\boldsymbol{P}_{k\|k-1}(j) = \boldsymbol{F}^j_k \boldsymbol{P}^j_{k-1\|k-1} (\boldsymbol{F}^j_k)^\top + \boldsymbol{Q}^j_k$
Update stage:	
Innovation residual:	$\tilde{\boldsymbol{z}}^j_k = \boldsymbol{z}_k - h(\hat{\boldsymbol{s}}^j_{k\|k-1})$
Innovation covariance:	$\boldsymbol{S}^j_k = \boldsymbol{H}^j_k \boldsymbol{P}^j_{k\|k-1} (\boldsymbol{H}^j_k)^\top + \boldsymbol{R}^j_k$
Kalman gain:	$\boldsymbol{K}^j_k = \boldsymbol{P}^j_{k\|k-1} (\boldsymbol{H}^j_k)^\top (\boldsymbol{S}^j_k)^{-1}$
A posteriori state estimate:	$\hat{\boldsymbol{s}}^j_{k\|k} = \hat{\boldsymbol{s}}^j_{k\|k-1} + \boldsymbol{K}^j_k (\boldsymbol{z}_k - \tilde{\boldsymbol{z}}^j_k)$
A posteriori estimated error covariance:	$\boldsymbol{P}^j_{k\|k} = (\boldsymbol{I} - \boldsymbol{K}^j_k \boldsymbol{H}^j_k) \boldsymbol{P}^j_{k\|k-1}$
Model Probabilities:	
Model likelihood:	$L^j_k = \frac{1}{\sqrt{(2\pi)^2 \det(\boldsymbol{S}^j_k)}} \exp \frac{-(\tilde{\boldsymbol{z}}^j_k)^\top (\boldsymbol{S}^j_k)^{-1} \tilde{\boldsymbol{z}}^j_k}{2}$
Model probability:	$p^j_k = \sum_{i=1}^2 P_{tr}(j,i) \frac{L^i_k p^i_{k-1}}{L^1_k p^1_{k-1} + L^2_k p^2_{k-1}}$
Mixing stage:	
State estimate:	$\hat{\boldsymbol{s}}_k = \sum_{j=1}^2 p^j_k \hat{\boldsymbol{s}}^j_k$
Error ovariance estimate:	$\hat{\boldsymbol{P}}_{k\|k} = \sum_{j=1}^2 p^j_k [\hat{\boldsymbol{P}}^j_{k\|k} + (\hat{\boldsymbol{s}}_k - \hat{\boldsymbol{s}}^j_k)(\hat{\boldsymbol{s}}_k - \hat{\boldsymbol{s}}^j_k)^\top]$

The IMM filtering steps and equations are summarized in Table 5.1. In these equations the following process noise covariance matrices are used for the CV and CT models:

$$\boldsymbol{Q}^{CV} = \begin{bmatrix} 0.1 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 \\ 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0.1 \end{bmatrix} \sigma^2_{w,CV}, \quad \boldsymbol{Q}^{CT} = \begin{bmatrix} 0.1 & 0 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 & 0 \\ 0 & 0 & 0.1 & 0 & 0 \\ 0 & 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0 & 0.002 \end{bmatrix} \sigma^2_{w,CT}. \tag{5.11}$$

An overview of IMM filtering steps is shown in Fig. 5.1. This figure follows the evolution of the predicted state of a target. Note that the predicted error covariance will evolve following a similar path. Here $\hat{\boldsymbol{s}}^{ij}_{k+1|k}$ following from the transition functions is the prediction using model i converted into the dimensions of model j. The prediction $\hat{\boldsymbol{s}}^{12}_{k+1|k}$ is $\hat{\boldsymbol{s}}^1_{k+1|k}$ augmented with a 0 entry and $\hat{\boldsymbol{s}}^{21}_{k+1|k}$ takes the first 4 elements of $\hat{\boldsymbol{s}}^2_{k+1|k}$.
Recall the solution method overview provided in Fig. 4.4. The integration of the IMM filter results in an additional input parameter for each of the objects. At the time of the budget update, it is for each object determined which of the models used in the IMM filter has the largest model probability. This model will in turn be used in the MPC sub-problem of that object to predict its future state evolution and find the budget distribution.

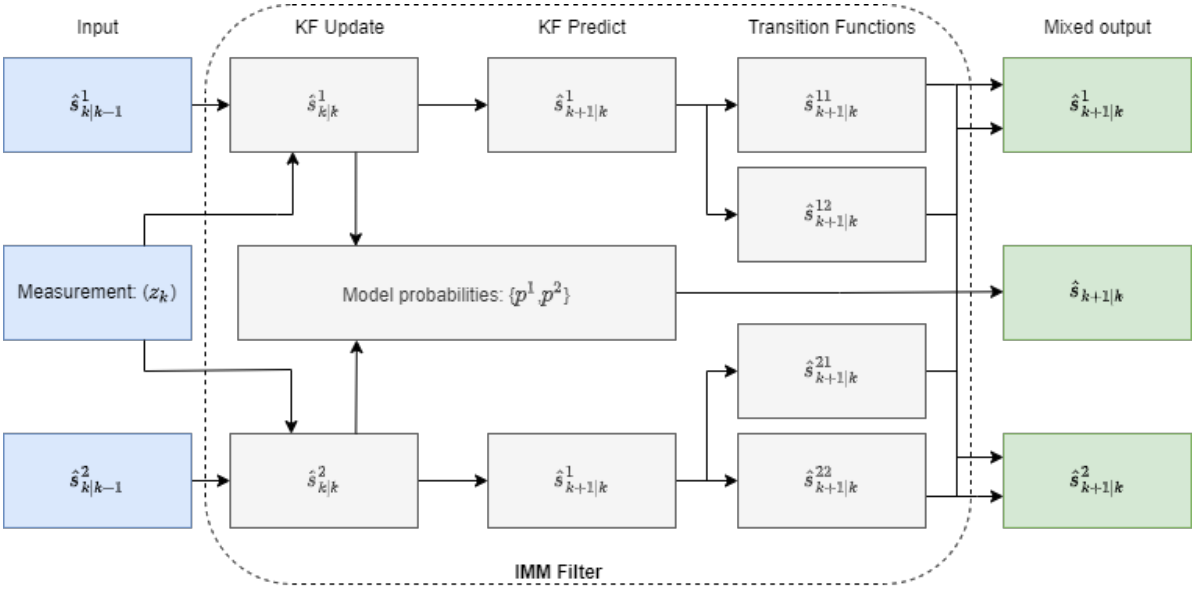

Figure 5.1: Interconnection of the IMM filtering steps.

5.2. Results

The objective of replacing the single model extended Kalman filter used in chapter 4 with an IMM filter was to make the solution method more generally applicable, so that it provides good results also in cases where targets do not follow a CV model. To investigate if this is indeed the case a number of simulations were run. Consider the scenario shown in Fig. 5.2 where two objects are tracked. Object 2 moves along a straight line at a constant velocity, while object 1 performs two maneuvers:

- A turn between t = 55 s and t = 89 s of -5 rad/s.
- A turn between t = 145 s and t = 179 s of 5 rad/s.

The simulation parameters used in this section can be seen in Table 5.2.

Table 5.2: Simulation parameters IMM.

Reference parameter	**Value**
Maneuverability noise variance CV model ($\sigma^2_{w,CV}$)	2.5 $(\frac{m}{s^2})^2$
Maneuverability noise variance CT model ($\sigma^2_{w,CT}$)	25 $(\frac{m}{s^2})^2$
Radar cross section (RCS)	10 m^2
Prediction horizon length (H)	5
Number of rollouts	10
Simulation update interval	5 s
Budget precision (ε)	0.002

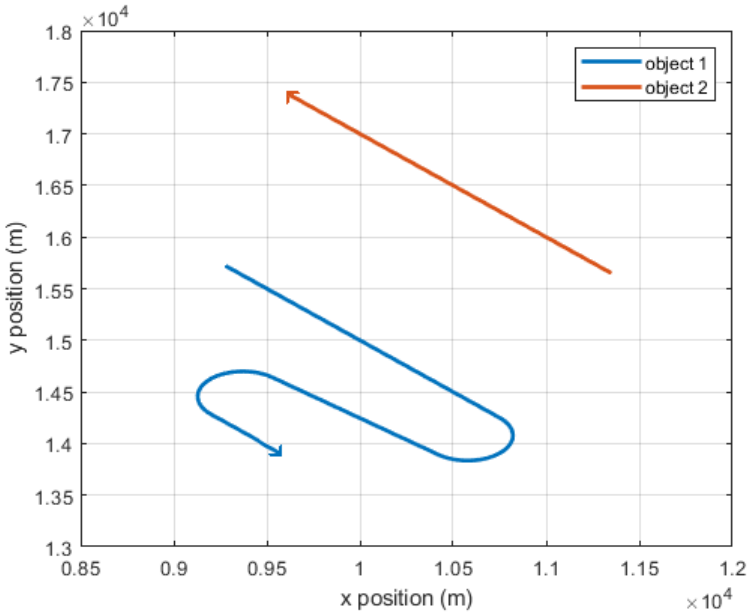

Figure 5.2: Trajectory of the 2 objects.

5.2.1. Model switching

The first thing to look into is to see if the IMM filter correctly estimates what movement the object is currently making. As described in section 5.1, at every time-step the IMM filter considers both models to hold with a certain likelihood. Consider object 1 following the trajectory described in section 5.2, it is investigated how the likelihoods of the two models are evolving over time. This is shown in Fig. 5.3. It can be seen that for times where the object is moving along a straight line the CV model has a large likelihood, while at times where the object is making a turn, the CT model has a large likelihood.

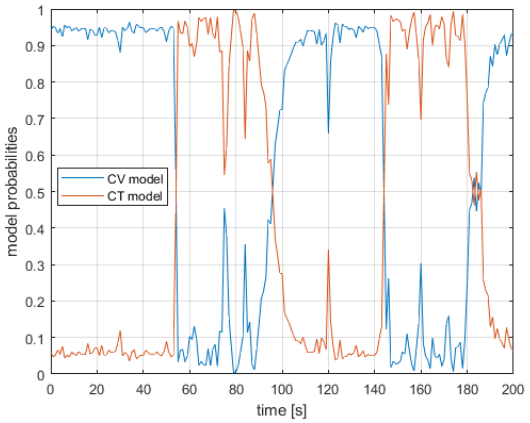

Figure 5.3: Model probabilities of object 1.

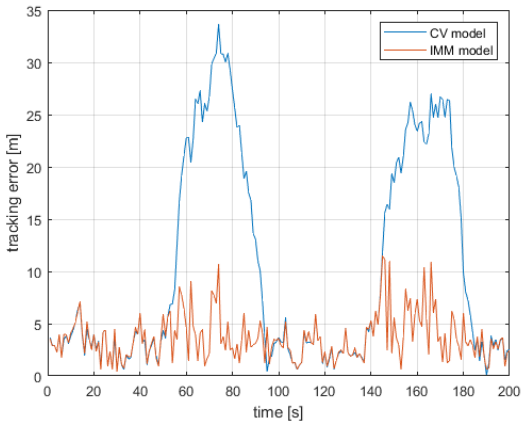

Figure 5.4: Tracking error using an IMM filter and a single model filter.

5.2.2. Tracking error

Secondly, it is investigated if employing an IMM filter results in a better tracking of targets that are maneuvering. To see if this is indeed the case, the first object following the trajectory described in

section 5.2 is tracked using two filters: A single model filter using a CV model and an IMM filter using both a CV model and a CT model. For both filters the tracking error at every time-step k is computed:

$$\mathrm{TE}_k = \sqrt{(x_k - \hat{x}_k)^2 + (y_k - \hat{y}_k)^2}, \tag{5.12}$$

where x_k and y_k are the ground truth x and y positions of the object and $\hat{x}$ and $\hat{y}$ are the estimated x and y positions of the filter. The tracking error of both simulations can be seen in Fig. 5.4. As expected, it can be seen that when the object is making a turn, the CV model filter fails to make a good prediction of the object. Using the IMM filter, the tracking error increases only slightly when the object is making a turn.

5.2.3. Budget distribution

Lastly, it is evaluated how the budget distribution is affected by the implementation of the IMM filter. When the object is making a turn, the uncertainty in its position increases. This is reflected by an increase in the maneuverability noise of the object in the CT model. In Fig. 5.5 and 5.6 the budget distributions of the scenario described in section 5.2 are shown. It can be seen that in the

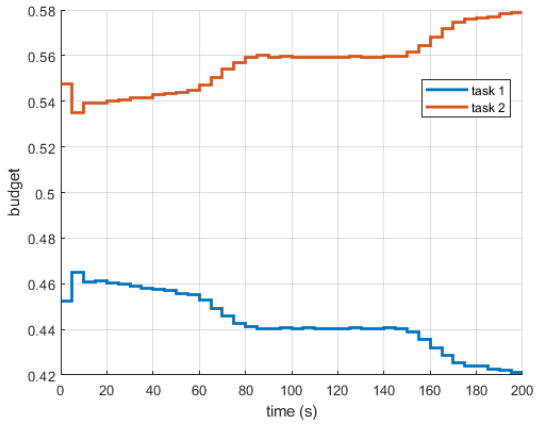

Figure 5.5: Budget distribution using a single model filter.

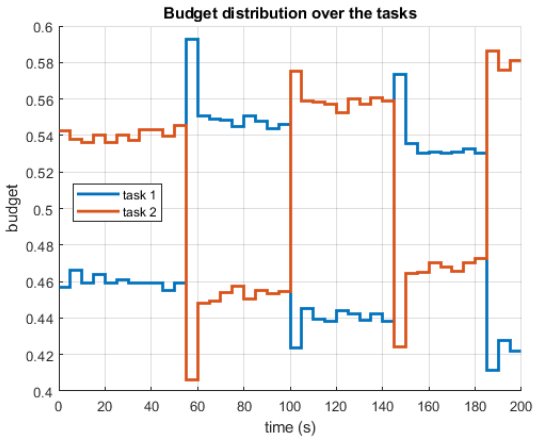

Figure 5.6: Budget distribution using an IMM filter.

single model Kalman filter case, the budget of tracking object 1 decreases, as the object is moving closer towards the radar. In the case of the IMM filter, the budget of tracking object 1 increases when the object is making a turn, as the maneuverability noise of the target is greater when making a turn. When object 1 is moving along a straight line, the budget distribution is roughly the same as in the single model filter case.

5.3. Conclusion

To make the proposed approach of using MPC to solve the RRM problem more generally applicable, an IMM tracking filter was employed. It was shown that using an IMM filter greatly improved the tracking of targets that do not follow a CV model. Furthermore, using this IMM implementation a larger amount of the radar resources could be allocated to those objects that have a large likelihood of making a turn.

6

Conclusion

6.1. Conclusion

This thesis introduces a novel algorithmic solution to the radar resource management (RRM) problem in multi-target tracking. Using this method, the problem is decoupled into sub-problems using Lagrangian relaxation after which these sub-problems are solved using model predictive control (MPC). The Lagrange multiplier is found iteratively using the golden section search method.

This novel solution method provides an alternative to the existing method of using policy rollout in combination with Lagrangian relaxation to solve the RRM problem. Four different dynamic tracking scenarios are considered to evaluate the performance of this approach in terms of computational efficiency and realised costs in comparison to policy rollout.

In the first scenario objects moving at a constant velocity in a straight line are tracked. It could be seen that MPC performs similarly compared to policy rollout in terms of realised cost and that MPC achieves a similar budget distribution within a much smaller execution time. It was shown that the usage of Lagrangian relaxation greatly helped in reducing the computation time, while the cost remained the same. Finally, the replacement of the subgradient method with golden section search to find the Lagrange multiplier reduces the number of function evaluations needed, especially when there are a lot of objects to track.

In the second scenario two objects are tracked of which one is stationary and the other one is moving in a straight line at a constant velocity with a turn halfway through the trajectory. It could again be seen in this scenario that MPC performs similarly compared to policy rollout in terms of realised cost and that MPC achieves a similar budget distribution within a much smaller execution time.

In the third scenario two objects are tracked of which one is stationary and the other one makes an unexpected maneuver into an area in which the quality of measurements is significantly lowered. In this case it could be seen that using MPC results in the track being lost, while by using policy rollout the target could continue to be tracked.

In the fourth scenario the objects cross some area, where the quality of measurements is significantly lowered. It was investigated what the importance of the prediction horizon is for the budget distribution. It could be shown that a sufficiently large prediction horizon needs to be chosen to adapt in time to events happening in the future.

To make the approach more generally applicable, an IMM tracking filter was employed. In contrast to a standard extended kalman filter, an IMM filter can track objects using multiple different motion models. It was shown that using an IMM filter greatly helped in reducing the tracking errors of the object in scenarios where the movements of the objects are not following a constant velocity model. Furthermore, using this IMM implementation a larger amount of the radar resources could be allocated to those objects that have a large likelihood of making a turn.

From the results in this thesis it can be concluded that MPC provides a computationally less intense alternative to policy rollout, while producing similar budget distributions and realised costs. This is a result of the MPC approach being an approximation of policy rollout: in MPC only the most likely future evolution is considered, while in policy rollout a number of future evolutions is considered and the actions are chosen based on the average of these evolutions. The drawback of this approximation can be seen in scenarios where small perturbations to the state evolution can lead to major differences in the cost, such as in the third scenario. In those scenarios the most likely future evolution does not properly approximate all future evolutions.

6.2. Recommendations

For optimizing the Lagrange multiplier λ, the golden section search was used in this thesis. While this showed a significant reduction in computation time compared to the subgradient method, it could be improved upon further by using more advanced non-derivative based optimization techniques such as Brent's method [34].

Furthermore, in this thesis the scenario was considered where only a single radar was tracking the different objects. To improve tracking performance, the number of radars can be increased. Then, the solution method needs to be extended to optimize the joint tracking performance of the multi-radar network, like it has been done for the policy rollout method in [38].

Recall the issue when scheduling the tasks while freely choosing $\boldsymbol{\tau}$ and $\boldsymbol{T}$, only taking into account that the maximum available budget is not exceeded. This issue was resolved by only choosing between certain values of $\boldsymbol{T}$. This assumption limits the performance of the tracking, as this drastically reduces the action space. It is worth investigating the use of more advanced scheduling techniques, such that the value of $\boldsymbol{T}$ is not limited only to a small number of values.

Lastly, an example cost function was presented in this thesis. In practice, this cost function could be any function depending on the needs of the radar system user.

Bibliography

[1] "Matlab sensor fusion and tracking toolbox." `https://www.mathworks.com/help/pdf_doc/fusion/fusion_ug.pdf`, R2021a. The MathWorks, Natick, MA, USA.

[2] M. Schöpe, H. Driessen, and A. Yarovoy, "Multi-task sensor resource balancing using lagrangian relaxation and policy rollout," 07 2020.

[3] A. Charlish, K. Bell, and C. Kreucher, "Implementing perception-action cycles using stochastic optimization," in *2020 IEEE Radar Conference (RadarConf20)*, pp. 1–6, 2020.

[4] A. Charlish, F. Hoffmann, and I. Schlangen, "The development from adaptive to cognitive radar resource management," *IEEE Aerospace and Electronic Systems Magazine*, vol. 35, 06 2020.

[5] S. Haykin, "Cognitive Radar: a Way of the Future," *IEEE Signal Processing Magazine*, vol. 23, pp. 30–40, Jan 2006.

[6] M. Bockmair, C. Fischer, M. Letsche-Nuesseler, C. Neumann, M. Schikorr, and M. Steck, "Cognitive Radar Principles for Defence and Security Applications," *IEEE Aerospace and Electronic Systems Magazine*, vol. 34, pp. 20–29, Dec 2019.

[7] R. Klemm, H. Griffiths, and W. Koch, *Novel Radar Techniques and Applications, Volume 2 - Waveform Diversity and Cognitive Radar, and Target Tracking and Data Fusion*. Scitech Publishing, 2017.

[8] S. Brüggenwirth, M. Warnke, S. Wagner, and K. Barth, "Cognitive radar for classification," *IEEE Aerospace and Electronic Systems Magazine*, vol. 34, pp. 30–38, Dec 2019.

[9] T. de Boer, M. I. Schöpe, and H. Driessen, "Radar resource management for multi-target tracking using model predictive control," *Journal of Advances in Information Fusion*, 2021, Submitted for Publication.

[10] P. A. Lynn, *The Radar Equation*, pp. 11–30. London: Macmillan Education UK, 1987.

[11] M. I. Skolnik, *Introduction to Radar Systems /2nd Edition/*. New York: McGraw Hill Book Co., 2 ed., 1980.

[12] A. O. Hero and D. Cochran, "Sensor Management: Past, Present, and Future," *IEEE Sensors Journal*, vol. 11, pp. 3064–3075, Dec 2011.

[13] P. W. Moo and Z. Ding, *Adaptive Radar Resource Management*. London: Academic Press, 1st ed., 2015.

[14] A. Charlish and F. Hoffmann, "Anticipation in Cognitive Radar using Stochastic Control," in *2015 IEEE Radar Conference (RadarCon)*, pp. 1692–1697, May 2015.

[15] V. Krishnamurthy, "POMDP Sensor Scheduling with Adaptive Sampling," in *17th International Conference on Information Fusion (FUSION)*, pp. 1–7, July 2014.

[16] D. A. Castanon, "Approximate Dynamic Programming for Sensor Management," in *Proceedings of the 36th IEEE Conference on Decision and Control*, vol. 2, pp. 1202–1207 vol.2, Dec 1997.

[17] E. K. P. Chong, C. M. Kreucher, and A. O. Hero, "Partially Observable Markov Decision Process Approximations for Adaptive Sensing," *Discrete Event Dynamic Systems*, vol. 19, pp. 377–422, Sep 2009.

[18] M. Otterlo and M. Wiering, "Reinforcement learning and markov decision processes," *Reinforcement Learning: State of the Art*, pp. 3–42, 01 2012.

[19] M. T. J. Spaan, *Partially Observable Markov Decision Processes*, pp. 387–414. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012.

[20] M. I. Schöpe, H. Driessen, and A. Yarovoy, "A Constrained POMDP Formulation and Algorithmic Solution for Radar Resource Management in Multi-Target Tracking," *Journal of Advances in Information Fusion*, 2021, Accepted for Publication.

[21] W. Koch, "Adaptive parameter control for phased-array tracking," in *Signal and Data Processing of Small Targets 1999* (O. E. Drummond, ed.), vol. 3809, pp. 444 – 455, International Society for Optics and Photonics, SPIE, 1999.

[22] S. Ross, J. Pineau, S. Paquet, and B. Chaib-draa, "Online planning algorithms for pomdps," *The journal of artificial intelligence research*, vol. 32 2, pp. 663–704, 2008.

[23] W. B. Powell, "A unified framework for stochastic optimization," *European Journal of Operational Research*, vol. 275, no. 3, pp. 795 – 821, 2019.

[24] E. Chong, C. Kreucher, and A. Hero, "Partially observable markov decision process approximations for adaptive sensing," *Discrete Event Dynamic Systems*, vol. 19, pp. 377–422, 09 2009.

[25] T. Brehard, P. Coquelin, E. Duflos, and P. Vanheeghe, "Optimal policies search for sensor management : Application to the esa radar," in *2008 11th International Conference on Information Fusion*, pp. 1–8, 2008.

[26] R. T. Perkins and W. B. Powell, "Stochastic optimization with parametric cost function approximations," 2017.

[27] W. Powell, "Approximate dynamic programming: Solving the curses of dimensionality," 08 2011.

[28] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: A Bradford Book, 2018.

[29] C. Topliff, W. Melvin, and D. Williams, "Application of pomdps to cognitive radar," in *2019 53rd Asilomar Conference on Signals, Systems, and Computers*, pp. 1662–1666, 2019.

[30] Y. He and E. Chong, "Sensor scheduling for target tracking: A monte carlo sampling approach," *Digital Signal Processing*, vol. 16, pp. 533–545, 09 2006.

[31] J. Pineau, G. Gordon, and S. Thrun, "Point-based value iteration: An anytime algorithm for pomdps," in *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1025 – 1032, August 2003.

[32] S. Ghadimi, R. Perkins, and W. Powell, "Reinforcement learning via parametric cost function approximation for multistage stochastic programming," Submitted.

[33] D. P. Bertsekas, "Dynamic programming and suboptimal control: A survey from adp to mpc*," *European Journal of Control*, vol. 11, no. 4, pp. 310–334, 2005.

[34] R. Brent, *Algorithms for minimization without derivatives*. Prentice-Hall, 1973.

[35] S. C. Chapra, *Applied Numerical Methods with MATLAB for Engineers and Scientists*. McGraw-Hill Science/Engineering/Math, 2006.

[36] P. Scokaert and D. Mayne, "Min-max feedback model predictive control for constrained linear systems," *IEEE Trans. Autom. Control.*, vol. 43, pp. 1136–1142, 1998.

[37] E. Mazor, A. Averbuch, Y. Bar-Shalom, and J. Dayan, "Interacting multiple model methods in target tracking: a survey," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 34, no. 1, pp. 103–123, 1998.

[38] B. van der Werk, "Approximately optimal radar resource management for multi-sensor multi-target tracking," Master's thesis. TU Delft, Delft, The Netherlands, May 2021.