# Approximately Optimal Resource Management for Multi-Function Radar

Algorithmic Solutions Using a Generic Framework

Max Ian Schöpe

# Approximately Optimal Resource Management for Multi-Function Radar

Algorithmic Solutions Using a Generic Framework

# Approximately Optimal Resource Management for Multi-Function Radar

Algorithmic Solutions Using a Generic Framework

**Dissertation**

for the purpose of obtaining the degree of doctor
at Delft University of Technology
by the authority of the Rector Magnificus, Prof. dr. ir. T.H.J.J. van der Hagen,
chair of the Board for Doctorates
to be defended publicly on
Friday 12 November 2021 at 10:00 o'clock

by

**Max Ian SCHÖPE**

Master of Science in Electrical Engineering,
Delft University of Technology, The Netherlands
born in Kassel, Germany

This work is part of the research programme "Integrated Cooperative Automated Vehicles" (i-CAVE), subproject "Communication" (P4), with project number 14895, which is partly financed by the Netherlands Organisation for Scientific Research (NWO).

*Keywords:* Radar Resource Management, Lagrangian Relaxation, Partially Observable Markov Decision Process, Policy Rollout.
*Printed by:* Ipskamp Drukkers B.V., Enschede, The Netherlands.
*Front & Back:* Design by Kalliopi Papangelopoulou.

ISBN 978-94-6384-263-1

An electronic version of this dissertation is available at
`http://repository.tudelft.nl/`.

Author e-mail: max.schope@protonmail.com

To my family and friends, for their support and encouragement

# Contents

# List of Acronyms

| | |
|---|---|
| AODB | Approximately Optimal Dynamic Budget Balancing |
| BP | Base Policy |
| CPA | Closest Point of Approach |
| CR | Cognitive Radar |
| DBF | Digital Beamforming |
| EKF | Extended Kalman Filter |
| FMCW | Frequency-Modulated Continuous-Wave |
| GPS | Global Positioning System |
| GSS | Golden Section Search |
| JDL | Joint Directors of Laboratories |
| KF | Kalman Filter |
| LD | Lagrangian Dual |
| LDP | Lagrangian Dual Problem |
| LR | Lagrangian Relaxation |
| LTI | Linear Time-Invariant |
| MDP | Markov Decision Process |
| MFR | Multi-Function Radar |
| ML | Machine Learning |
| MPC | Model Predictive Control |
| OODA | Observe-Orient-Decide-Act |
| OSB | Optimal Steady-State Budget Balancing |
| PBVI | Point-Based Value Iteration |
| PDF | Probability Density Function |
| POMDP | Partially Observable Markov Decision Process |
| PP | Primal Problem |
| PR | Policy Rollout |
| Q-RAM | Quality of Service Resource Allocation Method |
| RCS | Radar Cross Section |
| RL | Reinforcement Learning |
| RRM | Radar Resource Management |
| SNR | Signal-to-Noise Ratio |
| VI | Value Iteration |

# Summary

Recent advances in Multi-Function Radar (MFR) systems led to an increase in their degrees of freedom. As a result, modern MFR systems are capable of adjusting many parameters during runtime. An automatic adaptation of the radar system to changing situations, like weather conditions, interference, or target maneuvers, is often mentioned in the context of MFR and is usually called Radar Resource Management (RRM). This thesis aims at developing a generic framework and approximately optimal algorithmic solutions for solving RRM problems. This is achieved by formulating the sensor tasks as Partially Observable Markov Decision Processes (POMDPs). Although the focus is on MFR, the approach is not limited to such sensor systems and has broader applicability.

In Chapter 2, a first step is taken by investigating Lagrangian Relaxation (LR) and the subgradient method for optimally distributing the sensor resources to the different tasks in a multi-target tracking scenario. A constrained optimization problem is formulated. Using LR, the constraints can be included in the cost function. In a time-invariant scenario, it is shown that the proposed Optimal Steady-State Budget Balancing (OSB) algorithm will lead to balanced budgets based on track parameters like maneuverability and measurement uncertainty. The time-invariant scenario is a special case of general tracking scenarios, and the presented solution can be seen as the optimal POMDP solution in that case. Since real-world applications quickly lead to time-varying scenarios, it is demonstrated how the approach can be extended to such cases. Finally, the proposed method is compared with other budget assignment strategies.

Subsequently, the tracking tasks are explicitly formulated as POMDPs, and the novel Approximately Optimal Dynamic Budget Balancing (AODB) algorithm is proposed in Chapter 3. The algorithm applies a combination of LR and Policy Rollout (PR). PR is a Monte Carlo sampling method for POMDPs to find the expected future cost. Due to its generic architecture, the framework can be applied to different radar or sensor systems and cost functions. In a time-invariant scenario, the algorithm calculates a solution close to the optimal steady-state solution, as presented in Chapter 2. This is shown through simulations of a two-dimensional tracking scenario. Moreover, it is demonstrated how the algorithm dynamically allocates the sensor time budgets to the tasks in a changing environment using a non-myopic fashion. Finally, the algorithm's performance is compared with different resource allocation techniques.

Based on the previous results, Chapter 4 conducts a detailed investigation of the computational load of the AODB algorithm. It is shown how the choice of several input parameters influences computational performance. Additionally, Model Predictive Control (MPC) is applied in the same framework as an alternative POMDP solution method. Compared to stochastic optimization methods such as PR, the computational load is dramatically reduced while the resource allocation results are similar. This is shown through simulations of dynamic multi-target tracking scenarios in which the cost and computational load of different approaches are compared.

So far, this thesis has used tracking scenarios to demonstrate the validity of the proposed algorithms. Chapter 5 shows how to apply the proposed framework and algorithmic solution to a multi-target joint tracking and classification scenario. It is shown that tracking and classification can be considered in a single task type. Furthermore, it is shown how the task resource allocations can be jointly optimized using a single carefully formulated cost function based on the task threat variance. Multiple two-dimensional radar scenarios demonstrate how sensor resources are allocated depending on the current knowledge of the target position and class.

Chapter 6 extends the single-sensor approach shown in the previous chapters to multiple sensors and demonstrates the usefulness of the proposed algorithm in two different multi-sensor multi-target tracking scenarios. The first scenario considers a generic surveillance situation. An approximately optimal approach based on the previously proposed algorithm is formulated assuming a central processor. Subsequently, a distributed implementation is introduced that converges to the same results as the centralized implementation and requires less computational resources. The performance of the proposed approach for both centralized and distributed implementation is demonstrated through dynamic tracking scenarios. The second scenario focuses explicitly on an automotive application. The proposed generic framework and algorithmic solution are used to allocate scarce resources across multiple mobile sensor nodes. A central system manages the nodes' transmission and shares sensing data with other sensor nodes if this improves the overall track accuracy. The proposed method allocates time and frequency resources. Through simulation of a typical traffic situation, the validity of the approach is demonstrated.

This thesis shows that the application of the proposed novel generic framework and algorithmic solution increases the performance w.r.t. heuristic solutions. Furthermore, it is demonstrated that the proposed framework allows the user to exchange elements such as cost function or POMDP solution method to adjust it to specific needs. The proposed method can be applied in many different areas involving different types of sensors. Possible applications include automotive scenarios, such as autonomous driving or traffic monitoring, (maritime) surveillance, and air traffic control.

# Samenvatting

Recente ontwikkelingen in Multi-Function Radar (MFR)-systemen hebben geleid tot een toename van hun vrijheidsgraden. Als gevolg hiervan zijn moderne MFR-systemen in staat om tijdens de uitvoeringstijd veel parameters aan te passen. Een automatische aanpassing van het radarsysteem aan veranderende situaties, zoals weersomstandigheden, interferentie of targetmanoeuvres, wordt vaak gebruikt in de context van MFR en wordt meestal Radar Resource Management (RRM) genoemd. Dit proefschrift heeft tot doel het opstellen van een generiek raamwerk alsmede nagenoeg optimale algoritmische oplossingen ten behoeve van het oplossen van RRM-problemen. Dit wordt bereikt door de sensortaken te formuleren als Partially Observable Markov Decision Processes (POMDP's). Hoewel de focus ligt op MFR, is de aanpak niet beperkt tot dergelijke sensorsystemen en heeft deze een bredere toepasbaarheid.

In Hoofdstuk 2 wordt een eerste stap gezet door Lagrangian Relaxation (LR) en de subgradiëntmethode te onderzoeken om de sensorresources optimaal te verdelen over de verschillende taken in een multi-targets trackingscenario. Er wordt een optimalisatieprobleem met randvoorwaarden geformuleerd. Met behulp van LR kunnen de randvoorwaarden worden opgenomen in de kostenfunctie. In een tijdsinvariant scenario wordt aangetoond dat het voorgestelde Optimal Steady-State Budget Balancing (OSB)-algoritme zal leiden tot evenwichtige budgetten op basis van trackparameters zoals manoeuvreerbaarheid en meetonzekerheid. Het tijdsinvariante scenario is een speciaal geval van algemene trackingscenario's en de gepresenteerde oplossing kan in dat geval worden gezien als de optimale POMDP-oplossing. Aangezien toepassingen in de echte wereld snel leiden tot in de tijd variërende scenario's, wordt gedemonstreerd hoe de aanpak kan worden uitgebreid naar dergelijke gevallen. Ten slotte wordt de voorgestelde methode vergeleken met andere budgettoewijzingsstrategieën.

Vervolgens worden de trackingtaken expliciet geformuleerd als POMDP's en wordt het nieuwe Approximately Optimal Dynamic Budget Balancing (AODB)-algoritme voorgesteld in Hoofdstuk 3. Het algoritme past een combinatie van LR en Policy Rollout (PR) toe. PR is een Monte Carlo-steekproefmethode voor POMDP's om de verwachte toekomstige kosten te vinden. Door de generieke architectuur kan het raamwerk worden toegepast op verschillende radar- of sensorsystemen en kostenfuncties. In een tijdsinvariant scenario berekent het algoritme een oplossing die dicht bij de optimale stationaire oplossing ligt, zoals gepresenteerd in Hoofdstuk 2. Dit wordt aangetoond met behulp van simulaties van een tweedimensionaal trackingscenario. Bovendien wordt gedemonstreerd hoe het algoritme de sensortijdbudgetten dynamisch toewijst aan de taken in een veranderende omgeving op een niet-bijziende manier. Ten slotte worden de prestaties van het algoritme vergeleken met verschillende technieken voor het toewijzen van sensorresources.

Gebaseerd op de eerdere resultaten, voert Hoofdstuk 4 een gedetailleerd onderzoek uit naar de rekenbelasting van het AODB-algoritme. Er wordt getoond hoe de keuze van verschillende invoerparameters de rekenprestaties beïnvloedt. Bovendien wordt Model Predictive Control (MPC) toegepast in hetzelfde raamwerk als een alternatieve POMDP-oplossingsmethode. Vergeleken met stochastische optimalisatiemethoden zoals PR, wordt de rekenbelasting drastisch verminderd, terwijl de resultaten van de resourcetoewijzing vergelijkbaar zijn. Dit wordt aangetoond door simulaties van dynamische multi-target trackingscenario's waarin de kosten en rekenbelasting van verschillende benaderingen worden vergeleken.

Tot nu toe heeft dit proefschrift trackingscenario's gebruikt om de validiteit van de voorgestelde algoritmen aan te tonen. Hoofdstuk 5 laat zien hoe het voorgestelde raamwerk en de algoritmische oplossing kunnen worden toegepast op een scenario voor gezamenlijke tracking en classificatie van meerdere targets. Aangetoond wordt dat tracking en classificatie in één taaktype kunnen worden beschouwd. Verder wordt getoond hoe de toewijzingen van resourcen aan de taken gezamenlijk kunnen worden geoptimaliseerd met behulp van een enkele zorgvuldig geformuleerde kostenfunctie op basis van de variantie van taakbedreigingen. Meerdere tweedimensionale radarscenario's laten zien hoe sensorresources worden toegewezen, afhankelijk van de huidige kennis van de targetpositie en -klasse.

Hoofdstuk 6 breidt de benadering met één sensor uit de vorige hoofdstukken uit naar meerdere sensoren en demonstreert het nut van het voorgestelde algoritme in twee verschillende multi-target tracking scenario's met meerdere sensoren. Het eerste scenario gaat uit van een generieke surveillancesituatie. Een nagenoeg optimale benadering op basis van het eerder voorgestelde algoritme wordt geformuleerd, uitgaande van een centrale processor. Vervolgens wordt een gedistribueerde implementatie geïntroduceerd die convergeert naar dezelfde resultaten als de gecentraliseerde implementatie en die minder rekenkracht vereist. De prestaties van de voorgestelde aanpak voor zowel de gecentraliseerde als de gedistribueerde implementatie worden aangetoond door middel van dynamische trackingscenario's. Het tweede scenario richt zich expliciet op een toepassing in de automobielindustrie. Het voorgestelde generieke raamwerk en de algoritmische oplossing worden gebruikt om de schaarse sensorenresources toe te wijzen aan meerdere mobiele sensorknooppunten. Een centraal systeem beheert de transmissie van de knooppunten en deelt detectiegegevens met andere sensorknooppunten als dit de algehele tracknauwkeurigheid verbetert. De voorgestelde methode wijst tijd- en frequentieresources toe. Door simulatie van een typische verkeerssituatie wordt de validiteit van de aanpak aangetoond.

Dit proefschrift laat zien dat de toepassing van het voorgestelde nieuwe generieke raamwerk en de algoritmische oplossing de prestaties verhoogt met betrekking tot heuristische oplossingen. Verder wordt aangetoond dat het voorgestelde raamwerk de gebruiker in staat stelt om elementen zoals de kostenfunctie of de POMDP-oplossingsmethode uit te wisselen om het aan specifieke behoeften aan te passen. De voorgestelde methode kan op veel verschillende gebieden worden toegepast met verschillende soorten sensoren. Mogelijke toepassingen zijn automotive scenario's, zoals autonoom rijden of verkeersmonitoring, (maritieme) surveillance en luchtverkeersleiding.

# Preface

When starting as a PhD candidate at TU Delft, I had no idea what was lying ahead of me. As I primarily focused on electromagnetics and radio frequency theory during my studies, I had little knowledge of the optimization techniques and methods commonly used in Radar Resource Management and related fields. The first year was very tough, but thanks to the great support from my supervisors and colleagues, I slowly got more and more into the topic. Additionally, I significantly improved my skills as a critical researcher. My research gained more momentum towards the end, and I finally started to grasp all the crucial parts of my problem.

It was a long personal journey with many ups and downs. However, I learned a lot of things about myself, especially how I can best cope with challenging problems and organize myself. I am glad to finally write this preface, as it means that I have reached the end of the PhD journey. This dissertation summarizes my research results. I hope that my contribution will inspire other researchers to develop new ideas and approaches.

*Max Ian Schöpe*
*Delft, October 2021*

# 1 Introduction

*"There is a theory which states that if ever anyone discovers exactly what the Universe is for and why it is here, it will instantly disappear and be replaced by something even more bizarre and inexplicable. There is another theory which states that this has already happened."*

Douglas Adams in *The Restaurant at the End of the Universe*

## 1.1. Why Radar Resource Management?

Although the usage of electromagnetic waves for object detection has been studied at least since the beginning of the 20th century, the first fully operational radar systems were only developed before and during the Second World War, [1]. For example, the British Royal Air Force used the Chain Home radar system from 1938 to detect and track potentially threatening aircraft approaching the British coast. Using this radar, only the range and the angle of an object could be determined with limited accuracy. It was up to the experience of the operator to determine the exact amount of objects, their headings, as well as the amount of threat that they posed [2]. After the Second World War, radar systems became more and more complex and were applied in many other fields than military scenarios, e.g., for meteorology, air traffic control, ground and material inspections, and, most recently, in the automotive industry. Many of these advances have been made possible by improved transceivers and antennas, but also by the use of computers and advanced signal and data processing that took over an increasing amount of the operator's tasks [3].

Due to various technological improvements, the degrees of freedom of radar systems have increased significantly in recent decades [4]. The most notable examples of such improvements are the rise of phased-array antennas, Digital Beamforming (DBF) on transmit and receive, as well as digital waveform generation. This has led to a shift in radar systems from highly specialized systems that focus mostly on a single application towards so-called Multi-Function Radar (MFR) systems that are able to execute multiple functions jointly [5]. Among those functions are surveillance-related functions, such as object detection, tracking and classification. As a result, modern MFR systems are capable of adjusting many parameters during run-time. An automatic adaptation of the radar system to changing situations, like weather conditions, interference, or target maneuvers, is often mentioned in the context of MFR and is usually called Radar Resource Management (RRM). An illustration of such an MFR system for which RRM could be beneficial is shown in Figure 1.1. RRM is frequently considered within the broader context of so-called Cognitive Radar (CR) [6–10].

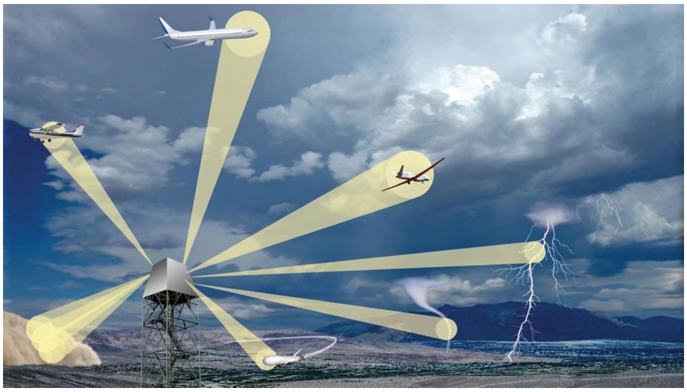


Figure 1.1: Artist depiction of an MFR system [11].

To clarify the usefulness of RRM in modern radar systems, some general assumptions about a possible MFR system are given below:

- The radar system has a phased array antenna and is able to move its beam into another direction within a matter of milliseconds.
- The system is supposed to handle multiple tasks such as detection, tracking and classification for various objects and (weather) phenomena.
- There are limited sensing resources available. The constraints affect the choice of sensing time, center frequency, and bandwidth or energy usage per task. If the resource allocations are decided independently per task, the MFR system often ends up in an overload situation, where more resources are requested than the sensor has available.
- Decisions about the current situation need to be made quickly, which only allows a high-level influence of a human operator.

MFR systems can be controlled on different levels taking into account different kinds of data. This has often been represented using the Joint Directors of Laboratories (JDL) data fusion model. Figure 1.2 shows such a model as block scheme from [6]. Similar representations are used in [12, 13]. On Level 0, the signal of the radar system is directly controlled by adjusting the waveform. This is a very low level of control that changes in relatively small time intervals (smaller 10 ms). Level 1 is related to the measurements, which can consist of multiple signals. Those signals have to be scheduled in the available time frame. Typically, the updating interval is in the order of a few 100 ms. The next level, Level 2, is related to the different objects. This part includes tracking and classification actions. On this level, the control is updated in the order of seconds. Level 3 deals with the management of the current overall situation. This means that the underlying tasks are given priorities based on the current circumstances, and the resources are allocated accordingly. The updates based on the situation are taking place in intervals of tens of seconds. The final Level 4 is related to the radar mission. Here, the general actions are defined which need to be taken to achieve the desired mission goals. This management stage is updated in intervals of 100 s or more.

The RRM methods developed in this thesis primarily deal with management tasks related to Levels 2 and 3. They assign resources to the different object-related tasks based on uncertainty and threat. However, the considered framework can also be used to extend the approach to include optimizing measurement- and waveform-related parameters. The benefit of applying RRM to optimize the waveform selection of radar systems has been discussed in the literature, e.g., in [14]. On a high level, the considered RRM algorithms in this thesis can be represented as a feedback loop as shown in Figure 1.3. This is sometimes referred to as an Observe–Orient–Decide–Act (OODA) loop [15]. Based on the predicted future situation, the priorities of the different tasks are determined using a cost function, and resources are allocated accordingly. Using the allocated resources, measurements of the environment are taken, leading to new estimations of the situation and corresponding covariances. From those estimations, new predictions can be made.

## 1.2. What is Radar Resource Management?

There are many ways of how to define RRM. Therefore, a couple of assumptions are summarized here to understand what is considered as RRM in this thesis and to understand the

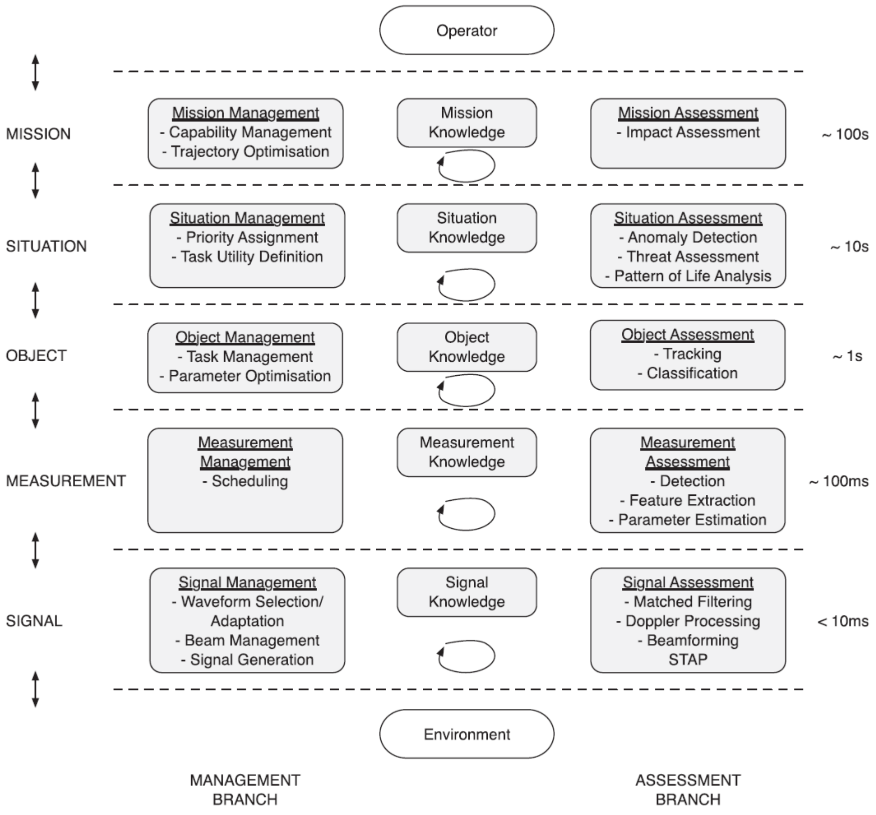


Figure 1.2: JDL data fusion model for MFR system [6].

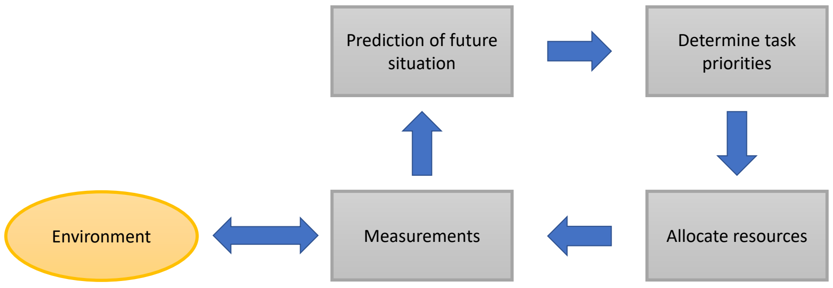


Figure 1.3: High-level feedback loop of an RRM algorithm.

goals of this research:

- Generally, RRM is supposed to maximize the performance of the sensor system from the user perspective. Therefore the idea is to find the optimal resource allocation.
- RRM techniques could be applied:
  - offline as a benchmark solution to test the performance of other RRM approaches or for designing radar systems. The computational effort can be high in this case, as there is no need for quick results.
  - online in a real system. This means that the resource allocations must be calculated fast and in real-time, while the implementation cost should stay affordable.
- An RRM algorithm needs to deliver a valid solution considering:
  - various sensor types with different characteristics.
  - sensor systems consisting of multiple sensors.
  - multiple, possibly very different sensor tasks.
  - various scenarios and user wishes.
- RRM should be able to take the expected future scenario into account when optimizing the sensor resources.

Accordingly, this thesis aims to develop a flexible and generic framework that can adapt to all the mentioned problem characteristics and find an objectively optimal formulation and solution of the problem. The goal is to find potential solutions for real radar applications, and the focus will thus be on an approximately optimal approach to allow a more straightforward future implementation in a real system with limited computational resources.

The advantages of a truly generic optimal approach would be the following:

- **Applicability**: the solution could be applied in a variety of fields that use different types of sensors. Additionally, different task types could be optimized jointly.
- **Optimality**: The results would be (approximately) optimal w.r.t. the chosen cost function.
- **User-friendliness**: The user would have to define the cost function only. Since this is no easy task, additional guidelines or tools might be necessary to help the user with this choice.
- **Simplicity**: The formulation of the problem and solution would be simple, which would make it easy for the user to judge the output.
- **Adaptivity**: Part of the solution (such as the optimization algorithm) could be replaced with other ones to suit the need of each application.

Compared to already available solutions, a generic optimal solution could tremendously decrease the development cost of new sensor systems since it could be reused for different applications and sensors. In addition to that, it could be applied in any scenario and environment with the appropriate user-chosen cost function.

Although this thesis focuses on MFR, the approach is not limited to such sensor systems and has broader applicability. One could think about, e.g., sonar or lidar applications, as well as video surveillance. Situations, where these types of sensors are applied include automotive scenarios, such as autonomous driving or traffic monitoring, (maritime) surveillance, and air traffic control. Even for applications without sensors, the proposed management framework could potentially be applied. Examples are the management of energy networks in cities, the management of resources in communication networks or even applications in finance or economy, where the assignment of specific resources leads to different performances. Many applications that can be formulated as a feedback loop could be solved with a similar framework.

## 1.3. High-Level Overview of Existing Approaches

Much of the research on RRM (see, e.g., the overview by Hero and Cochran in [4] or by Moo and Ding in [5]) focuses on a single task, e.g., keeping a constant track quality even under target maneuvers. This usually means managing the time budget spent on a particular task. However, MFR systems are commonly operating at their sensor time and energy budget limit. In such cases, increasing the budget for one task means simultaneously decreasing the budget of the others, inevitably deteriorating their performance. In this thesis, part of the RRM problem is therefore seen as a budget or resource balancing act over multiple individual tasks.

Heuristic solutions have been presented in the past (see, for instance, the overview in [16]), some relying on assigning task priorities and priority-based scheduling. Applying heuristics too early in the design leads to complicated solutions, e.g., nested if-then-else rules. It is not easy to understand what problem is solved within those approaches and whether or not and in what sense the solution is optimal. This usually does not lead to a reusable generic algorithm. In addition, a priority-based scheduler usually does not balance the budget over all tasks but simply schedules the jobs in order of priority (as, e.g., applied in [17] and [18]). When the timeline is fully occupied, it often leaves a set of tasks with the lowest priorities that together do not fit anymore. These approaches do not consider decreasing the time budgets of individual tasks. Furthermore, the determination of the levels of priorities and the rules for assigning them is often not easy and prone to heuristics.

Additionally, most heuristic approaches are myopic, which means that they optimize the sensor resources for the subsequent timestep only. While this might be sufficient for many applications, a generic approach would benefit from making decisions based on the expected future situation. Suppose enough knowledge is available to predict how the observed environment will develop. In that case, the RRM algorithm could consider that in an earlier stage and avoid sudden resource allocation changes due to “unexpected” situations. Such an approach is called non-myopic (see, e.g., [19]).

Another aspect of multi-task RRM is the difficulty of allocating resources to different kinds of tasks, such as detection, tracking, and classification. Many available approaches combine searching for new targets and tracking known targets. Commonly, tracking and searching are defined in separate problems that are rather heuristically combined into a single measure (see, e.g., [20–23]). Other methods try to develop RRM algorithms for tracking and classification by defining a risk or threat measure that directly depends on track and class parameters (see, e.g., [16, 24, 25]). Kreucher and Hero have presented a very general approach in [26] where they use the uncertainty in the joint multitarget probability density as the objective function for assigning resources. It is stated that this can theoretically be done with detection, tracking, and classification tasks at once, but it is only demonstrated for detection and tracking tasks. Defining a relevant threat or risk measure directly related to the underlying task uncertainties seems to be the most useful and straightforward solution and is also investigated in this thesis. Similarly, Charlish et al. also present high-level RRM approaches that theoretically include many aspects that this research focuses on. Nevertheless, a detailed practical implementation of an algorithm that solves the RRM problem for different task types using these general frameworks has not been demonstrated so far. The available solutions usually do not explicitly mention the assumed simulation parameters, making it difficult to correctly comprehend and interpret the results.

To conclude, different available problem formulations and proposed solutions vary w.r.t. the specifications of the models, actions, and task states involved and the cost function. While the concept of general optimal approaches already exists in literature, it has never been fully developed and presented. The descriptions of such an RRM solution stayed on a very high level. For the first time, a practical and detailed implementation of an algorithmic solution using such a generic framework is presented in this thesis, with many examples containing a variety of different explicit parameter values.

## 1.4. Research Objectives and Approach

The research objectives can be summarized in the following six main parts:

1. The development of methods to trade-off multiple tracking tasks. The assumption is that the sensor is operating at its resource limit, and the available resources need to be balanced. Contrarily to simple scheduling approaches where the resource need is calculated independently per task, the resource limit for all tasks needs to be included in the resource allocation procedure.
2. Addressing the uncertainty in the measurements and object movements using stochastic control methods to implement non-myopic optimization.
3. Solving the problem non-myopically by taking into account the expected future.
4. The development of basic cost functions that serve as an example.
5. Showing that the RRM solution works jointly for tracking and classification.
6. Extending the developed algorithm for multiple sensors.

In this thesis, the problem is treated as an optimal stochastic control problem which relies on an explicit formulation of

- the inference problem that the radar has to solve in terms of dynamic and measurement models,

- the control actions that the sensor has available, which reflect the degrees-of-freedom of the MFR mentioned earlier,
- a cost function that reflects the system performance that the user would like to optimize.

To the author's best knowledge, an overall solution to the RRM problem using this approach has not been presented so far. It has been suggested that a truly optimal solution could possibly lead to a significant improvement of the performance of adaptive sensors [27], but that still needs to be illustrated. However, even if the performance would not improve much over heuristic solutions that are carefully tuned to each application, a reusable generic framework will reduce the design effort of RRM solutions. Consequently, such a framework would reduce the development cost and time and aid in understanding the system behavior.

In this thesis, the RRM problem is considered a multi-task time budget-constrained control problem, where the individual tasks are different tracking tasks. Our chosen problem formulation directly leads to the assumption of a constrained Partially Observable Markov Decision Process (POMDP). The presented examples explicitly deal with tracking and classification scenarios, while other tasks such as searching, for instance, are not covered. However, the chosen POMDP framework is suitable for other sensor tasks as well.

## 1.5. Novelties and Main Results

- In this thesis, the Optimal Steady-State Budget Balancing (OSB) algorithm is introduced. It is shown how using it led to the optimal balancing of sensor budgets in a Linear Time-Invariant (LTI) setting. It applied LR to distribute the resources over the different tasks.
- Subsequently, generic dynamical problems by utilizing the POMDP framework are considered. This thesis introduces the Approximately Optimal Dynamic Budget Balancing (AODB) algorithm derived from the OSB algorithm to solve dynamical problems. This novel algorithm uses a cost function based on the predicted error-covariance of the Extended Kalman Filter (EKF). It was shown that the results of this generic approach are approximately optimal w.r.t. the steady-state error-covariance of a one-dimensional Kalman Filter (KF). The RRM problem was solved non-myopically using an online Monte Carlo technique called Policy Rollout (PR), which stochastically predicts the future. The AODB algorithm was applied to a dynamic radar tracking scenario to emphasize its practical value in variable problem settings. Such a practical analysis of a comparable RRM algorithm has never been presented before.
- In addition to that, a comparison of the performance of the AODB algorithm to several other resource allocation techniques was conducted. It was shown that applying the AODB always led to the lowest cost.
- Furthermore, the computational load of the AODB algorithm was investigated in detail in a practical setting, and a computationally more efficient implementation was presented. This is the first time that such a detailed analysis of a comparable RRM algorithm has been published.

- Additionally, it was shown that the proposed method could be used for multiple task types, such as tracking and classification. Although these different tasks might require different measurement actions if considered separately, the proposed solution achieves a novel solution that suits both task types.

- Finally, the framework was also extended to deal with sensor networks. For the first time, it was shown in a practical setting how sensor actions can be optimized for multiple tasks in a non-myopic fashion while balancing the resources rather than applying a sensor selection.

## 1.6. Outline of the Thesis

The remainder of this thesis is structured as follows:

**Chapter 2** presents the first step towards a generic RRM framework. It introduces the fundamental problem formulation that is used throughout the thesis. It then presents a novel myopic RRM approach that allocates the sensor resources by applying LR and the subgradient method, called OSB algorithm. It is shown that the proposed algorithm can balance the resources in a simple LTI tracking scenario using a cost function based on the tracking accuracy.

**Chapter 3** expands the balancing approach of Chapter 2 by assuming an underlying POMDP framework for the target states. This chapter introduces the so-called AODB algorithm that solves the POMDPs non-myopically by applying PR. The algorithm's performance is shown through two-dimensional tracking scenarios assuming different sensor parameters and a more complete radar scenario.

**Chapter 4** investigates the computational load of the algorithm proposed in Chapter 4 and gives recommendations of which input parameters to use in practice. Furthermore, it introduces an alternative implementation of the algorithm, which uses Model Predictive Control (MPC) instead of PR. It is shown that this approach significantly reduces computational load.

**Chapter 5** demonstrates that the proposed framework can be used for joint tracking and classification of multiple targets. It is shown that proper modeling and a suitable cost function formulation allow it to consider both tracking and classification through a single sensor task type and a single cost function. This is shown through multiple two-dimensional simulation scenarios.

**Chapter 6** extends the previously proposed single sensor approach to a sensor network with multiple sensors. It demonstrates that the proposed framework also works in such a case and demonstrates it through multiple two-dimensional simulation scenarios. Additionally, it is shown how the implementation of the algorithm can be adjusted to allow the problem to be solved in a distributed fashion for independent sensor nodes at different locations.

**Chapter 7** contains the conclusions and gives recommendations for possible future research.

# 2 Balancing of Radar Resources for Multi-Target Tracking

*Based on the findings from the literature review, this chapter is the first step towards a generic solution of the RRM problem and introduces the basic problem formulation that is used throughout this thesis. It presents a non-myopic RRM solution method for an LTI scenario using LR and the subgradient method. Through simple one-dimensional tracking scenarios, it is demonstrated how these techniques can be used to optimize the sensor resources considering a cost function related to the tracking accuracy.*

Parts of this chapter have been published in:

M. I. Schöpe, H. Driessen, and A. Yarovoy, "Optimal Balancing of Multi-Function Radar Budget for Multi-Target Tracking Using Lagrangian Relaxation", in *Proceedings of the 22nd International Conference on Information Fusion (FUSION)*, Ottawa, Ontario, Canada, 2019.

## 2.1. Optimal Multi-Task Radar Resource Management

As already mentioned in Chapter 1, the approach here is not to schedule tasks with fixed resource demands into the sensor timeline. This thesis aims to find optimal RRM solution methods that find the best global resource allocation for all tasks and fit into the total available budget. The problem can therefore be formulated as a constrained optimization problem.

Optimization problems that involve constraints are usually more difficult to solve than unconstrained problems. The constraints can be divided into equality and inequality constraints, as shown in the general problem stated in (2.1) where the first two constraints are equality constraints and the last one is an inequality constraint:

$$\begin{aligned} \underset{\boldsymbol{x}}{\text{minimize}} \quad & f(\boldsymbol{x}) \\ \text{subject to} \quad & \boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{A} \\ & \boldsymbol{h}(\boldsymbol{x}) \geq \boldsymbol{B} \\ & \boldsymbol{i}(\boldsymbol{x}) < \boldsymbol{C}. \end{aligned} \tag{2.1}$$

Depending on the type of constraint, different solution methods can be chosen. If the optimization problem only involves equality constraints, LR can be used to convert it to an unconstrained optimization problem that can be solved iteratively. The solution will converge to the equality constraint. When one or more inequality constraints need to be considered, the solution is generally more difficult to obtain. Other solution methods such as, e.g., linear, non-linear, or quadratic programming need to be applied. These techniques have been covered extensively in the literature, and more information about the different optimization methods can be found e.g. in [28–31]. When this kind of optimization is applied to a dynamic system, it falls into the domain of optimal control. An introduction and overview can, e.g., be found in [32–34].

It has been shown that LR is a beneficial technique to solve constrained optimization problems related to RRM; see, for instance, the approaches by Wintenby and Krishnamurthy in [35], White and Williams in [36], or Castañón in [37]. Alternatively, the Quality of Service Resource Allocation Method (Q-RAM) could be used. Q-RAM requires an action space discretization while LR allows the sub-problems to be solved analytically. Nevertheless, these methods are conceptually very similar. Some interesting approaches using Q-RAM have been shown by Irci et al. in [38] and Charlish et al. in [39] and [40].

This chapter will focus on the solution of RRM problems when the cost can easily be calculated through a relatively simple function. It is structured as follows. The assumed general RRM problem is formulated in Section 2.2 and Section 2.3 describes how a cost function can be formulated and what the general requirements are. The specific tracking problem treated in this chapter is defined in Section 2.4 as a constrained optimization problem. Subsequently, Section 2.5 introduces our proposed solution of the RRM problem, using LR and the subgradient method. Furthermore, the results are illustrated by time-invariant and time-variant tracking scenarios in Section 2.6, using the position uncertainty from the predicted steady-state Kalman error covariance matrix as cost function. Finally, the chapter ends with general conclusions in Section 2.7.

## 2.2. General Radar Resource Management Problem Definition

This section introduces the general optimization problem that will be considered in the following chapters.

### 2.2.1. Motion Model

At every moment in time $t$, each target considered within this model can be characterized by a state based on its position and velocity in $x$ and $y$ direction within a two-dimensional Cartesian coordinate system. For target $n$ this state is defined as

$$\boldsymbol{s}_t^n = \left[x_t^n \quad y_t^n \quad \dot{x}_t^n \quad \dot{y}_t^n\right]^T, \tag{2.2}$$

where $x_t^n$, $y_t^n$ and $\dot{x}_t^n$, $\dot{y}_t^n$ are the position and velocity of target $n$ in $x$ and $y$, respectively. The future target state at time $t + \Delta t$ can be calculated following a function

$$\boldsymbol{s}_{t+\Delta t}^n = f_{\Delta t}\left(\boldsymbol{s}_t^n, \boldsymbol{w}_t^n\right), \tag{2.3}$$

where $s_{t+\Delta t}^n$ is the next following state at time $t + \Delta t$ and $\boldsymbol{w}_t^n \in \mathbb{R}^4$ is the maneuverability noise for target $n$ at time $t$. The state evolution equation (2.3) directly defines the evolution Probability Density Function (PDF) which is given as

$$p\left(\boldsymbol{s}_{t+\Delta t}^n | \boldsymbol{s}_t^n\right). \tag{2.4}$$

### 2.2.2. Measurement Model

A sensor is assumed that is taking noisy observations of the state $\boldsymbol{s}_t^n$ with sensor action $\boldsymbol{a}_t^n \in \mathbb{R}^m$, where $m$ is the amount of adjustable action parameters. A measurement $\boldsymbol{z}_t^n$ of target $n$ at time $t$ can be characterized by using the measurement function $\mathfrak{h}$ as

$$\boldsymbol{z}_t^n = \mathfrak{h}\left(\boldsymbol{s}_t^n, \boldsymbol{v}_t^n, \boldsymbol{a}_t^n\right), \tag{2.5}$$

where $\boldsymbol{v}_t^n \in \mathbb{R}^q$ is the measurement noise for target $n$ and $q$ is the amount of measurement parameters. The measurement equation (2.5) directly defines the measurement PDF which can be written as

$$p\left(\boldsymbol{z}_t^n | \boldsymbol{s}_t^n, \boldsymbol{a}_t^n\right). \tag{2.6}$$

### 2.2.3. Tracking Algorithm

For the tracking scenarios considered in this chapter, a tracking algorithm should be chosen that aims at computing the posterior density. For linear systems, a KF can be adopted as an exact solution. For non-linear systems, possible algorithms are an extended KF (EKF) or a particle filter, for example.

### 2.2.4. Budget Optimization Problem

As mentioned in Chapter 1, the radar sensor is assumed to have a limited maximum budget $\Theta_{max}$ of any kind. For action $\boldsymbol{a}_t^n$ that is executed for each task $n$, a certain amount of budget (e.g. time or energy allocations) is required. In an overload situation, the current tasks require more of the total budget than is available. Thus, the available budget has to be distributed over the tasks to minimize a cost (e.g., related to the uncertainty of the current situation).

At time $t$, the optimization problem for $N$ different tasks can be written as

$$\begin{aligned} \underset{\boldsymbol{a}_t}{\text{minimize}} \quad & \sum_{n=1}^{N} c(\boldsymbol{a}_t^n, \boldsymbol{s}_t^n) \\ \text{subject to} \quad & \sum_{n=1}^{N} \Theta_t^n(\boldsymbol{a}_t^n) \leq \Theta_{max}, \end{aligned} \tag{2.7}$$

where $\Theta_t^n \in [0, 1]$ is the budget for task $n$ at time $t$, $c(\cdot)$ is the used cost function and $\Theta_{max} \in [0, 1]$ is the maximum available budget (0: no budget assigned, 1: all budget assigned). If all resources are supposed to be optimized, the maximum budget $\Theta_{max}$, will be 1 but in some cases also values smaller than 1 are possible, e.g., if resources have to be reserved for very important other tasks.

## 2.3. The Cost Function

When applying such an RRM approach, the final performance of the sensor system will be determined by the cost function, which is preferred over a heuristic approach. However, it introduces the explicit formulation of such a cost function in the application of the framework. The definition of an operationally relevant cost function is essential to benefit from these techniques efficiently but is not the focus of this thesis. An example of an operationally more relevant cost function has been discussed by Katsilieris et al. [41]. However, it should be noted that their cost function formulation is designed for a specific application and might not lead to the expected results in other situations.

Sometimes it has been suggested that generic measures of performance, such as the Information Gain or the Renyi divergence applied to the posterior density of the full state, could be applied (see, e.g., [20, 42]). Given the very different natures of radar scenarios, such generic measures are more reliable and comparable. For instance, if the cost is measured in the Information Gain, this formulation is directly clear and valid as it is not connected to a specific application. This research is based on the conviction that one single cost function will not meet the desires of different users in different applications with different sensors, targets, and environments.

The development of specific cost functions is essential and will be a development task in itself that will require close cooperation with potential users. In heuristic solutions, a specific cost function is often already built-in in the approach and cannot easily be adjusted. This is one of the reasons why the primary focus in this thesis is on developing a generic framework for RRM, which allows the user to decide on the preferred cost function. Therefore, the development of such user-specific cost functions is out of the scope, but it will be shown how different cost functions affect the results.

## 2.4. Radar Resource Management Tracking Problem Definition

In this section, the tracking problem under consideration is described in detail. The chosen parameters to be optimized are the revisit time $T$ and the dwell time $\tau$. In the following, time will be referred to in time steps $k_n$ which are separeted by the revisit time $T_n$ for target $n$. It is assumed that a certain number of targets are in the observable area around our radar system and are already being tracked. The targets are moving according to a linear dynamical system. When discrete time steps $k$ are considered, the next state for target $n$ can be predicted as

$$\boldsymbol{s}^n_{k_n+1} = \boldsymbol{F}(T_n) \cdot \boldsymbol{s}^n_{k_n} + \boldsymbol{w}^n_{k_n}, \tag{2.8}$$

where $\boldsymbol{s}^n$ is the state of the target and $\boldsymbol{F}(T_n) \in \mathbb{R}^{4\times4}$ is the according state transition matrix which is based on the revisit time $T_n$. Moreover, $\boldsymbol{w}^n_{k_n} \in \mathbb{R}^4$ is the zero-mean Gaussian maneuverability noise for target $n$, whose covariance is defined as

$$\begin{aligned} E\left(\boldsymbol{w}^n_{k_n}(\boldsymbol{w}^n_k)^T\right) &= \begin{bmatrix} \frac{(T_n)^2}{2} & 0 \\ 0 & \frac{(T_n)^2}{2} \\ T_n & 0 \\ 0 & T_n \end{bmatrix} \begin{bmatrix} \frac{(T_n)^2}{2} & 0 & T_n & 0 \\ 0 & \frac{(T_n)^2}{2} & 0 & T_n \end{bmatrix} \sigma^2_{w,n} \\ &= \begin{bmatrix} \frac{(T_n)^4}{4} & 0 & \frac{(T_n)^3}{2} & 0 \\ 0 & \frac{(T_n)^4}{4} & 0 & \frac{(T_n)^3}{2} \\ \frac{(T_n)^3}{2} & 0 & (T_n)^2 & 0 \\ 0 & \frac{(T_n)^3}{2} & 0 & (T_n)^2 \end{bmatrix} \sigma^2_{w,n} \end{aligned} \tag{2.9}$$

with $\sigma^2_{w,n}$ being the maneuverability noise variance.

All measurements are considered to be well separated, so there are no association problems. In this chapter, it is assumed that the relationship between the Cartesian target state and the measurements is linear. In that case, the measurement function in (2.5) can be replaced by a matrix. For target $n$ at time $k_n$, the measurement can therefore be described as

$$\boldsymbol{z}^n_{k_n} = \boldsymbol{H} \cdot \boldsymbol{s}^s_{k_n} + \boldsymbol{v}^n_{k_n}, \tag{2.10}$$

where $\boldsymbol{H} \in \mathbb{R}^{2\times4}$ is the measurement matrix and $\boldsymbol{v}^n_{k_n} \in \mathbb{R}^2$ the zero-mean Gaussian measurement noise. It is assumed that the standard deviation of the latter depends on the dwell time $\tau$ as

$$\sigma_{v,n} = \frac{\sigma_{0,n}}{\sqrt{\tau_n}}, \tag{2.11}$$

where $\sigma_{0,n}$ is a predefined basic measurement noise standard deviation for target $n$. This relation of dwell time and measurement standard deviation is slightly different as presented in [43] where the square root was not applied. It has been shown that the formulation in (2.11) is more realistic when considering a Signal-to-Noise Ratio (SNR) based on the dwell time (see Chapter 3 for more information).

For the actual tracking, a KF can be applied, for instance. The problem that is being solved in this chapter is how to optimally assign dwell times and revisit intervals to the different tracks to achieve the best result according to some cost. The problem is formulated as an optimization problem to find the minimum of a cost function $c(T_n, \tau_n)$, constrained by a maximum available budget for all tracking tasks.

The general optimization problem can therefore be described as

$$\begin{aligned} \underset{\boldsymbol{T},\boldsymbol{\tau}}{\text{minimize}} \quad & \sum_{n=1}^{N} c(T_n, \tau_n) \\ \text{subject to} \quad & \sum_{n=1}^{N} \frac{\tau_n}{T_n} \leq B_{max}, \end{aligned} \tag{2.12}$$

where $N \in \mathbb{Z}^+$ is the amount of tasks (or the amount of targets to be tracked), $\boldsymbol{T} = [T_1, ..., T_N]^T \in \mathbb{R}^N$ are the revisit intervals and $\boldsymbol{\tau} = [\tau_1, ..., \tau_N]^T \in \mathbb{R}^N$ the dwell times for all $N$ targets, $c(T, \tau)$ is the chosen cost function and $B_{max} \in [0, 1]$ is the maximum time budget for all tasks combined.

The time budget is defined as the ratio of dwell time and revisit interval. Therefore, this number represents the fraction of the revisit interval that is being used by the dwell time per task. The idea of the global constraint $B_{max}$ is to limit the total time budget of all tasks to a value between 0 (no sensor time used) and 1 (all sensor time used).

## 2.5. Proposed Solution of the RRM Tracking Problem

Our solution approach uses the LR technique. Following this approach, the original optimization problem, or Primal Problem (PP), can be relaxed by adding the constraints as penalty terms to the cost function, which results in the so-called Lagrangian Dual (LD). The optimization problem of finding the maximum of the LD over the Lagrange multiplier (also referred to as dual variable), is called LD Problem (LDP) and can be expressed as

$$Z_D = \max_{\lambda} \left( \min_{\boldsymbol{T},\boldsymbol{\tau}} \left( \sum_{n=1}^{N} \left( c(T_n, \tau_n) + \lambda \cdot \frac{\tau_n}{T_n} \right) \right) - \lambda \cdot B_{max} \right). \tag{2.13}$$

where $\lambda \in \mathbb{R}$ is the Lagrange multiplier for the budget constraint.

It needs to be noted that in general cases when only a weak duality can be assumed, the solution of the LDP is a lower bound to the PP. Generally, there is still a so-called duality gap between the optimal PP and the optimal LDP solution. Only when strong duality can be assumed, the duality gap is zero, and the solution of the LDP is equal to the solution of the PP. More details about LR and duality can be found in Appendix A.

It can easily be seen that the resulting LDP in (2.13) is just a sum of $N$ sub-optimization problems, one for each task. Therefore, this problem does not have to be solved for all tasks jointly but can be decoupled into the individual tasks $n$. Accordingly, (2.13) is split up into $N$ easier to solve optimization problems. It is important to realize that $\lambda$ is a single multiplier for the sum of all sub-optimization problems. Therefore, it is considered the outer optimization problem, solved after optimizing the parameters $T_n$ and $\tau_n$ per task $n$. Since those sub-optimization problems can still be quite complicated, they are going to be solved in different stages through an iterative process with steps $l \in \mathbb{Z}^+$. First, an initial value for the Lagrange multiplier $\lambda^l$ is chosen.

The LD function for each target $i$ is solved with the current Lagrange multiplier value, as shown in (2.14):

$$\{T_n^l, \tau_n^l\} = \underset{T_n, \tau_n}{\arg\min} \left( c(T_n, \tau_n) + \lambda^l \left( \frac{\tau_n}{T_n} \right) \right). \tag{2.14}$$

The total budget $B_{max}$ is omitted here, because it is a constant with respect to $\boldsymbol{T}^l$ and $\boldsymbol{\tau}^l$ and does therefore not change the position of the minimum in the Lagrangian. The current optimal values $\boldsymbol{T}^l = [T_1^l, ..., T_N^l]^T$ and $\boldsymbol{\tau}^l = [\tau_1^l, ..., \tau_N^l]^T$ are then used to find the next Lagrange multiplier $\lambda^{l+1}$. This is done by the use of the subgradient method, as explained in detail in Appendix A. The subgradient for Lagrange multiplier $\lambda^l$ is chosen as

$$s_\lambda^l = \sum_{n=1}^{N} \frac{\tau_n}{T_n} - B_{max}. \tag{2.15}$$

The Lagrangian multiplier is then updated with a chosen step size $\zeta^l$. Therefore the new Lagrangian multipliers for the next iteration are calculated as shown in (2.16):

$$\lambda^{l+1} = max\{0, \lambda^l + \zeta^l s_\lambda^l\}. \tag{2.16}$$

The initial multiplier value $\lambda^0$ has to be suitably chosen. Since the budget constraint is an inequality constraint, the value of its Lagrange multiplier can only be positive. With the new value $\lambda^{l+1}$, the process is started again until the desired precision of the solution is reached.

## 2.6. One-Dimensional Tracking Scenario

In this chapter, a one-dimensional tracking scenario is considered as an example. To conduct the tracking, the KF is used according to the state and measurement definitions mentioned in (2.2), (2.8) and (2.10). In this section, a possible solution to such a scenario is discussed.

### 2.6.1. Steady-State of the Tracking Filter Error-Covariance

If the targets are known already, and the KFs are assumed to be staying in a steady-state, the predicted error-covariance matrix of the KF could be used as the cost function. Being in steady-state means that the measurement uncertainties and maneuverabilities per target are constant, and therefore an optimal time-invariant KF error-covariance can be computed. It can be shown that the α-β-filter is an example for a steady-state KF [44]. The predicted error-covariance matrix is minimized in order to find the revisit intervals $\boldsymbol{T}$ and the dwell times $\boldsymbol{\tau}$.

For this scenario, it is assumed that the objects follow a linear dynamical system as described in (2.8) and (2.10). The state matrix in the one-dimensional case is therefore defined as

$$\boldsymbol{s} = [p, \dot{p}]^T, \tag{2.17}$$

where $p$ is the position of the target and $\dot{p}$ is the velocity of the target.

A method to calculate the error covariance for a steady-state KF has been introduced by Kalata in [45] with an extension by Gray and Murray in [46]. In their work, the steady-state equations for an $\alpha$-$\beta$-filter are calculated based on a tracking index $\Lambda$, which for a target $n$ is defined as

$$\begin{aligned}\Lambda_n &\propto \frac{\text{position maneuverability uncertainty of target n}}{\text{position measurement uncertainty of target n}} \\ &\mathrel{\hat{=}} \frac{T_n^2 \sigma_{w,n}}{\sigma_{v,n}},\end{aligned} \tag{2.18}$$

where $\sigma_{v,n}$ is the standard deviation of the measurement noise (see (2.11)) and $\sigma_{w,n}$ is the standard deviation of the maneuverability noise for target $n$. Through the tracking index, the filter parameters $\alpha$ and $\beta$ can be calculated. To simplify the calculations, an extra damping parameter has been introduced in [46], which is defined as

$$\begin{aligned}r_n &= \sqrt{1-\alpha_n} \\ &= \frac{4+\Lambda_n-\sqrt{8\Lambda_n+\Lambda_n^2}}{4}.\end{aligned} \tag{2.19}$$

From this, $\alpha$ can be calculated as

$$\begin{aligned}\alpha_n &= 1-r_n^2 \\ &= 1-\left(\frac{4+\Lambda_n-\sqrt{8\Lambda_n+\Lambda_n^2}}{4}\right)\end{aligned} \tag{2.20}$$

and $\beta$ as

$$\beta_n = 2(2-\alpha_n)-4\sqrt{1-\alpha_n}. \tag{2.21}$$

Based on (2.20) and (2.21), the filtered covariance matrix can be formed, as shown by Kalata in [45]. The corresponding matrix is shown in (2.22).

$$\begin{aligned}\boldsymbol{P}_{k_n|k_n}(T_n,\tau_n,\sigma_{0,n},\sigma_{w,n}) &= \begin{bmatrix}\sigma_{p,n}^2 & \sigma_{pv,n}^2 \\ \sigma_{pv,n}^2 & \sigma_{v,n}^2\end{bmatrix} \\ &= \begin{bmatrix}\alpha_n\sigma_{v,n}^2 & \frac{\beta_n}{T_n}\sigma_{n,n}^2 \\ \frac{\beta_n}{T_n}\sigma_{v,n}^2 & \frac{(2\alpha_n-\beta_n)\beta_n}{2(1-\alpha_n)T_n^2}\sigma_{v,n}^2\end{bmatrix}.\end{aligned} \tag{2.22}$$

Since the interest here lies in creating a cost function based on the prediction of the error covariance matrix, the approach in [45] is followed and the cost function for target $n$ is defined according to the KF prediction equations as

$$\begin{aligned}&\boldsymbol{P}_{k_n|k_n-1}(T_n,\tau_n,\sigma_{0,n},\sigma_{w,n}) \\ &\quad= \boldsymbol{F}(T_n)\boldsymbol{P}_{k_n|k_n,n}(T_n,\tau_n,\sigma_{0,n},\sigma_{w,n})\boldsymbol{F}^T(T_n)+\boldsymbol{\Psi}_n\boldsymbol{\Psi}_n^T\sigma_{w,n}^2 \\ &\quad= \begin{bmatrix}1 & T_n \\ 0 & 1\end{bmatrix}\begin{bmatrix}\alpha_n\sigma_{v,n}^2 & \frac{\beta_n}{T_n}\sigma_{v,n}^2 \\ \frac{\beta_n}{T_n}\sigma_{v,n}^2 & \frac{(2\alpha_n-\beta_n)\beta_n}{2(1-\alpha_n)T_n^2}\sigma_{v,n}^2\end{bmatrix}\begin{bmatrix}1 & 0 \\ T_n & 1\end{bmatrix} \\ &\qquad+\begin{bmatrix}\frac{T_n^2}{2} \\ T_n\end{bmatrix}\begin{bmatrix}\frac{T_n^2}{2} & T_n\end{bmatrix}\sigma_{w,n}^2,\end{aligned} \tag{2.23}$$

where $\boldsymbol{F}(T_n)$ is the dynamic matrix for target $n$.

As a simple cost function for target $n$, the first element of the predicted error-covariance matrix will be used, corresponding to the error covariance in range. The cost function definition is slightly different from the one presented in [43]. In contrast to that previous formulation, an additional term is added here that penalizes very small values of $T$ to avoid choosing parameters that are impractical in an actual application:

$$c_{1,n}(T_n, \tau_n) = \begin{bmatrix} 1 & 0 \end{bmatrix} \boldsymbol{P}_{k_n|k_n-1}(T_n, \tau_n, \sigma_{0,n}, \sigma_{w,n}) \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \frac{1000}{(T_n)^2}. \tag{2.24}$$

Cost function $c_1$ is convex for positive values of $T$ and $\tau$. Therefore, the algorithm achieves approximate optimality in this case. The cost function will lead to a budget distribution based only on the different targets' maneuverability and measurement uncertainty.

## 2.6.2. Time-Invariant Problem

In a time-invariant scenario, the KFs for all objects are assumed to be in a steady-state, which means that both the measurement uncertainties and the maneuverabilities are constant. This leads to a constant predicted error-covariance matrix. A single solution can therefore be calculated that is valid for every moment in time.

To illustrate that the LR approach leads to an automatic budget distribution, a simulation is conducted. The budgets are calculated according to the method mentioned in Section 2.5, without creating an explicit schedule of the task. Three targets are assumed, starting at different positions from the radar, which is positioned at the origin of the coordinate system. The state of the objects is defined as $\boldsymbol{s} = [p, \dot{p}]^T$ in a Cartesian coordinate system where $p$ is the position of the target in meters while $\dot{p}$ is its velocity in meters per second. For the KF, the dynamic matrix for target $n$ is defined as $\boldsymbol{F}(T_n) = [1, T_n; 0, 1]$, while the measurement matrix is defined as $\boldsymbol{H} = [1, 0]$. As cost function, $c_1$ is used as defined in (2.24). The targets have different measurement uncertainties and maneuverabilities to point out that our approach balances the budgets according to those uncertainties while still taking the constraints into account. A total budget $B_{max}$ of 1 is assumed, which corresponds to using all available sensor time for tracking. The LR step size is set to a constant value. The features of the tracked targets are summarized in Table 2.1, while the general simulation parameters are shown in Table 2.2.

Table 2.1: Initial target parameters for time-invariant scenario simulation.

| **Target** | **Position [m]** | **Velocity [m s$^{-1}$]** | **Measurement variance [m$^2$]** | **Maneuverability [m$^2$ s$^{-4}$]** |
|---|---|---|---|---|
| 1 | -1000 | 10 | 25 | 25 |
| 2 | 2000 | 20 | 25 | 250 |
| 3 | 1000 | -30 | 300 | 25 |

The simulation results are shown in Figure 2.1 and in Table 2.3. It can be seen that the LR indeed converges to constant budget values, which add up to a total of 1. In every

Table 2.2: Simulation parameters for time-invariant scenario.

| **Parameter** | **Value** |
|---|---|
| Amount of targets ($N$): | 3 |
| Maximum budget ($B_{max}$): | 1 |
| Initial Lagrangian multiplier ($\lambda_0$): | 10000 |
| Step size for LR ($\zeta$): | 100 |
| Precision of subgradient solution: | 0.01 |
| Cost function: | $c_1$ |

time step of the simulation, the Lagrange multipliers are adjusted, leading to the Lagrangian approaching the value of the cost function. The process stops after 411 iterations when the subgradient of the constraint reaches 0 with the desired precision of 0.01. Since the cost function is only based on the measurement uncertainty and maneuverability, the actual state of the targets has no direct impact on the result. Obviously, both the differences in maneuverability (see the budget difference between tasks 1 and 2) and the differences in measurement uncertainty (see the budget difference between tasks 1 and 3) lead to quite different sensor budgets for the different tasks. This solution is not trivial, as it can not simply be achieved analytically.

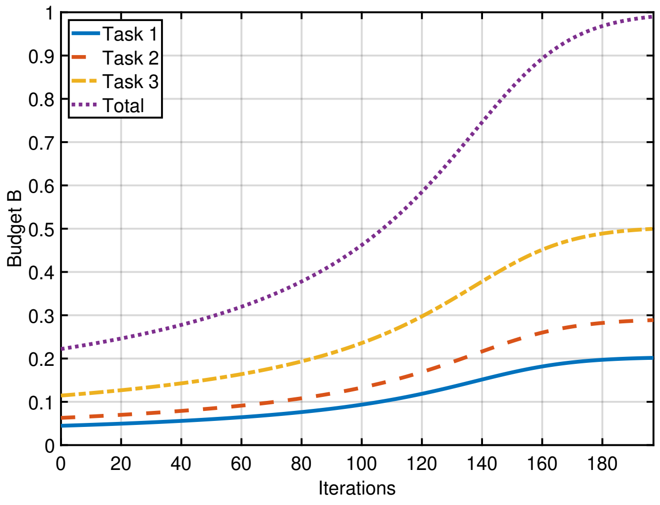


Figure 2.1: Simulation results of time-invariant budget allocation using cost function $c_1$ for three tracked targets.

### 2.6.3. Time-Variant Problem

In this time-variant scenario, it is assumed that the state of the targets influences the cost. The position and velocity will influence the solutions, which will therefore not be valid for the whole future, as assumed in the time-invariant scenario. For that reason, it needs to be

Table 2.3: Simulation results of time-invariant budget allocation after convergence using cost function $c_1$ for three tracked tasks.

| **Target** | **Revisit interval** $T$ [s] | **Dwell time** $\tau$ [s] | **Budget** |
|---|---|---|---|
| 1 | 1.79 | 0.36 | 0.20 |
| 2 | 1.27 | 0.37 | 0.29 |
| 3 | 1.58 | 0.79 | 0.50 |

updated in certain intervals in which the filters can be assumed to be nearly in a steady-state.

The formulation of the state vector $\boldsymbol{s}$, the dynamic matrix $\boldsymbol{F}$ and the measurement matrix $\boldsymbol{H}$ are the same as in the time-invariant scenario. Since cost function $c_1$ is only depending on the uncertainty of the measurement and the maneuverability, it will assign resources to targets only according to the uncertainty of their states. In a real application, this will for instance lead to paying more attention to targets that are far away than to closer ones. This is not a very useful cost function formulation, because the threat of an object is directly related to its state. It is therefore obvious that is is very important to carefully formulate the cost function according to the mission needs. For illustration purposes, the above mentioned cost function is extended by a heuristic threat factor $\theta_t(\boldsymbol{s})$. This threat factor is based on the threat formulation as used for example by Katsilieris, Driessen and Yarovoy in [47] and is related to the Closest Point of Approach (CPA). Since the examples presented in this chapter are one-dimensional, only the time to reach the CPA is considered. The CPA is equivalent to the radar location in our case. To convert the time into threat, the same sigmoid function as suggested in [47] is used, with an additional offset of $+0.1$, to avoid a factor of 0. The following parameters for the sigmoid function are applied: $t_1 = 10\,\mathrm{s}$, $t_{0.5} = 20\,\mathrm{s}$ and $t_0 = 30\,\mathrm{s}$. This is by all means not the best cost function. Its purpose is to point out how important it is to define a proper cost function and to illustrate the impact of an extra heuristic factor. The cost function $c_2$ for target $n$ is therefore defined as

$$c_{2,i}(T_n, \tau_n) = \left( \begin{bmatrix} 1 & 0 \end{bmatrix} \boldsymbol{P}_{k_n|k_n-1}(T_n, \tau_n, \sigma_{0,n}, \sigma_{w,n}) \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \frac{1000}{(T_n)^2} \right) \theta_t(\boldsymbol{s}_n). \qquad (2.25)$$

The budgets for all targets are updated in a fixed time interval $\beta_t$. During this time interval, measurements are conducted according to the calculated dwell times $\boldsymbol{\tau}$ and revisit intervals $\boldsymbol{T}$. Separate KFs are used to track the objects accordingly. All $\boldsymbol{T}$ and $\boldsymbol{\tau}$ stay constant until a new update of the budgets is performed by the use of LR. Two separate simulations with $\beta_t = 5\,\mathrm{s}$ and $\beta_t = 10\,\mathrm{s}$ are conducted. It is assumed that the targets are constantly tracked without track drops or reinitializations.

The LR budget algorithm is fed with the predicted positions given by the KF based on noisy measurements. The features of the simulated targets are the same as in the previous simulation; see Table 2.1. All other simulation-related values are shown in Table 2.4 and the trajectories of the targets are shown in Figure 2.2.

The simulation results are shown in Figures 2.3 and 2.4. It can be seen that our LR approach leads to changing budgets over time according to the uncertainty and the threat. When an object is expected to reach the radar position comparatively quickly, it gets much

Table 2.4: Simulation parameters for time-variant scenario.

| **Parameter** | **Value** |
|---|---|
| Amount of targets ($N$): | 3 |
| Maximum budget ($B_{max}$): | 1 |
| Total simulation time ($t_{max}$): | 100 s |
| Simulation step size ($t_{step}$): | 0.1 s |
| Budget update interval ($\beta_t$): | 5 s and 10 s |
| Initial Lagrangian multiplier ($\lambda_0$): | 10000 |
| Step size for LR ($\zeta$): | 1 |
| Precision of subgradient solution: | 0.01 |
| Cost function: | $c_2$ |

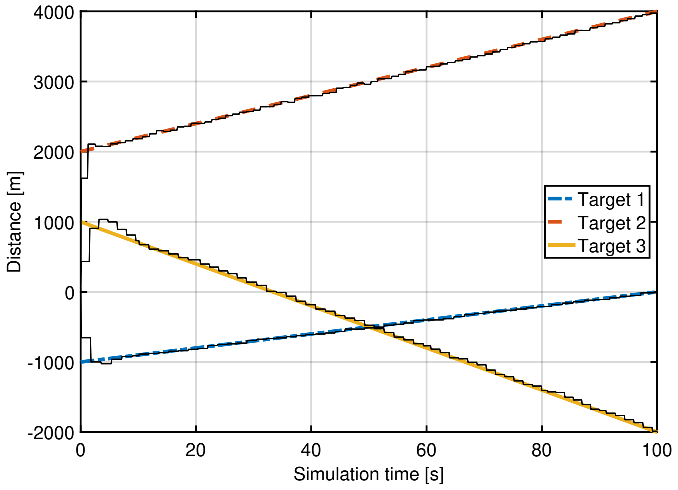


Figure 2.2: Target trajectory of the three simulated targets mentioned in Table 2.1. The thin lines show the predicted positions by the KFs.

more attention than the other targets. At the same time, the total sum of budgets stays within the constraint.

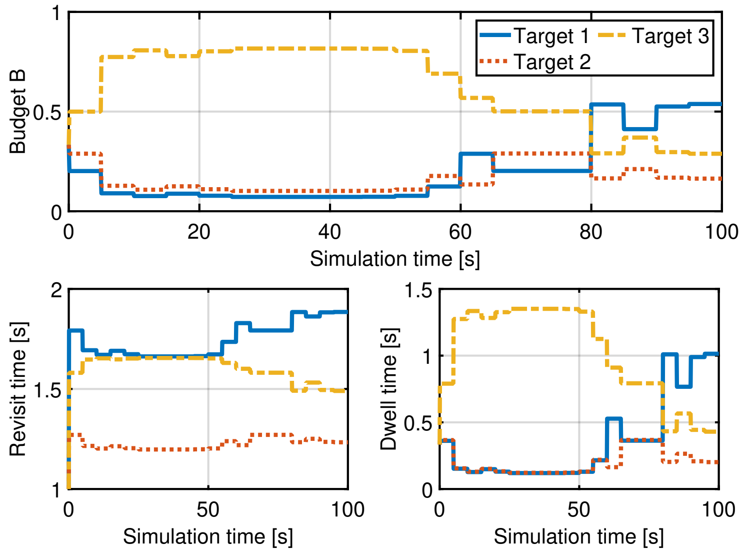


Figure 2.3: Simulation results of time-variant budget allocation using cost function $c_2$ for three tracked targets with a budget update interval of $\beta_t = 5$ s.

Of course, the budget update interval $\beta_t$ has an impact on the resulting budgets, which can be seen when comparing Figures 2.3 and 2.4. If the budgets are updated fast enough, the KFs can be assumed to a stay in a steady-state. If the target states change very quickly, the chosen budget update interval needs to be reduced. It is therefore important to choose this interval properly. Figure 2.5 shows the resulting cost differences when different fixed budget update periods are compared to a budget update period of $\beta_t = 1$ s. Contrary to the previous simulations, the LR budget algorithm is fed with the exact target position in order to remove any uncertainty and to be able to compare the cost properly. It can be seen that the smallest budget update interval always leads to the smallest cost differences. The longer the update interval, the higher the cost compared to the optimal solution.

### 2.6.4. Comparison of Time-Variant Solution with Other Approaches

To illustrate that this approach leads to improved results, it is compared with different budget allocation techniques according to the cost given by cost function $c_2$. The simulation parameters are identical with those in Tables 2.3 and 2.4, but the LR budget algorithm is again fed with the exact target positions. The different strategies are

- LR approach with a budget update interval of $\beta_t = 1$ s
- Random budget distribution
- Equal budget distribution ($1/N$)

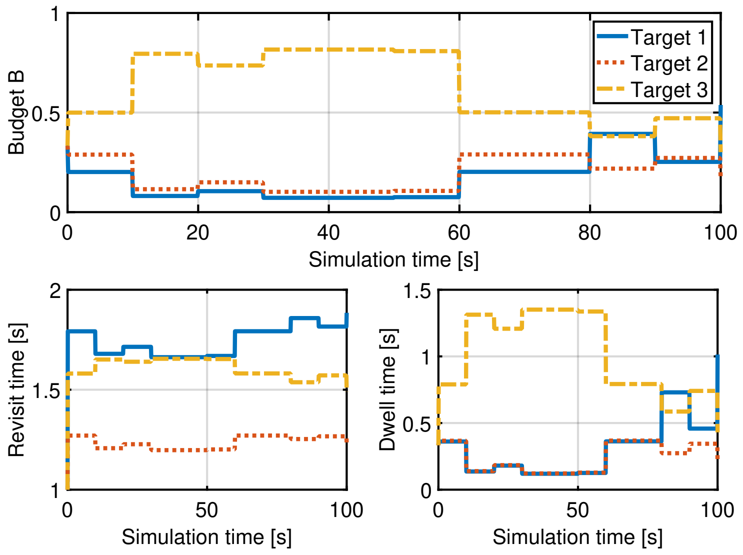


Figure 2.4: Simulation results of time-variant budget allocation using cost function $c_2$ for three tracked targets with a budget update interval of $\beta_t = 10$ s.

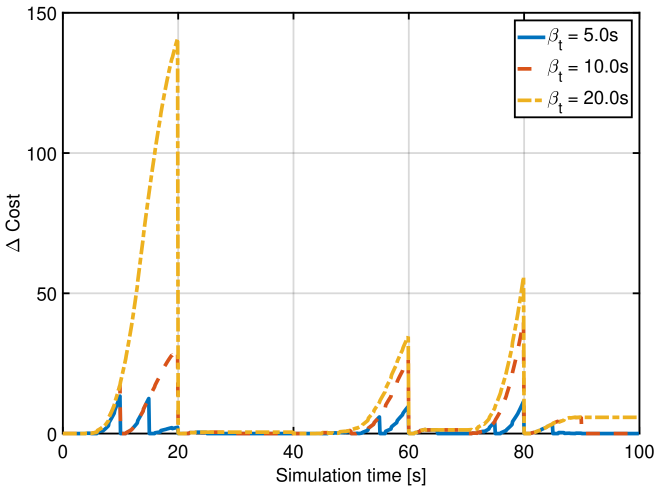


Figure 2.5: Simulation results of time-variant budget allocation using cost function $c_2$ for three tracked targets with different budget update intervals $\beta_t$. The difference in cost is with respect to the cost result for $\beta_t = 1$ s.

A comparison of the results from different budget assignment approaches is shown Figure 2.6. It can be seen that the LR approach always delivers the lowest cost.

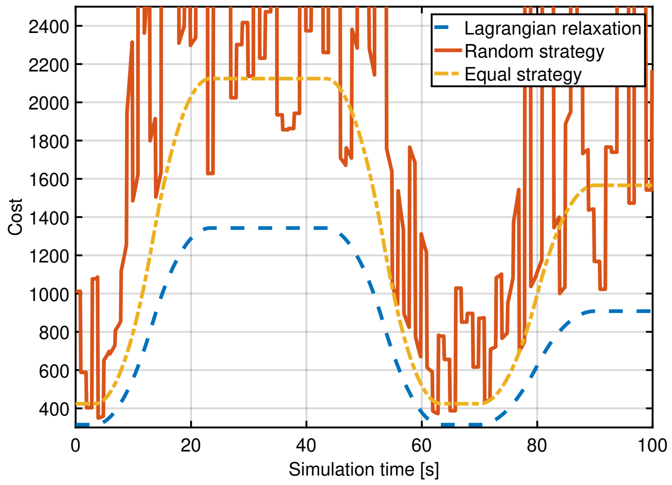


Figure 2.6: Comparison of the cost of three different budget assignment strategies. It can be seen that the LR strategy always leads to the lowest total cost.

## 2.7. Conclusions

This chapter explores the advantages of an optimal approach for solving the RRM problem of tracking multiple targets. Many solutions have been proposed previously that claim to solve RRM problems for single task optimization. However, the problem becomes more complicated to solve when multiple tasks have to be considered, and the MFR system is operating at its resource limit. Allocating more resources for one task will then inevitably reduce them for the other tasks, leading to a deteriorated performance. In this chapter, it has been shown that an optimal steady-state solution for such a multi-task resource balancing problem for an LTI tracking problem can be obtained by applying LR.

The optimization problem has been decoupled into sub-optimization problems using LR and has then been solved using the subgradient method. The proposed solution has been applied to a simple one-dimensional tracking scenario with three objects to be tracked. The revisit intervals and dwell times are the radar parameters that have to be tuned. A constraint on the total budget guarantees that the total available budget will not be exceeded. Two different cost functions based on the steady-state KF error covariance have been taken into account. The results show that the resources are successfully allocated according to the budget constraint, and optimality w.r.t. the cost function is reached with any desired precision. For the presented LTI scenario, the algorithm calculates the optimal non-myopic solution to the allocation problem, assuming an infinite optimization horizon. For this particular LTI case, the solution can be seen as the optimal POMDP solution.

The chosen cost function is based on the one-step-ahead prediction error variances, which leads to a larger relative budget for targets with higher maneuverability or larger measurement error variances. The time budget is therefore based on uncertainty, which is generally desirable. This would imply that long-range objects will receive more radar budget in an operational radar scenario than objects close by. Such behavior is usually not very useful from an operational point of view. It shows that for future practical implementation, the explicit formulation of operationally relevant cost functions is required.

It has been illustrated that slightly more useful solutions are already generated when applying a heuristic operationally inspired adaptation to the cost function. Since this adapted cost function introduces a time-varying setting, the LR algorithm based on steady-state analysis has been applied to predefined time intervals. It has been shown that the budget update intervals lead to approximately optimal solutions for a dynamic scenario, as long as they are chosen small enough. In that case, the tracking filters can be assumed to stay in a steady-state during those intervals. As the situation evolves and the algorithm does not consider these changes, the solution of the algorithm can now rather be considered myopic.

Finally, a comparison of our LR approach with other heuristic budget distribution strategies has been presented. It can be seen that our technique indeed always delivers the lowest cost. It is the first time that this kind of analysis has been performed.

# 3 A Generic Framework for Radar Resource Management in Multi-Target Tracking

*In the previous chapter it was shown, that LR and subgradient method are efficient tools to allocate sensor resources in a simple one-dimensional LTI setting. This chapter builds upon those results and extends it to a non-myopic solution approach by assuming an underlying POMDP for each target. Since Chapter 2 considered a cost function for the steady-state, that approach already treated a special case of the POMDP. In this chapter, the PR method is applied to solve these POMDPs. The performance of this algorithmic solution is evaluated through two-dimensional tracking scenarios that take more radar parameters into account than in the last chapter.*

Parts of this chapter have been published in:

M. I. Schöpe, H. Driessen, and A. Yarovoy, "Multi-Task Sensor Resource Balancing Using Lagrangian Relaxation and Policy Rollout", in *Proceedings of the 23rd International Conference on Information Fusion (FUSION)*, Online, 2020.

M. I. Schöpe, H. Driessen, and A. Yarovoy, "A Constrained POMDP Formulation and Algorithmic Solution for Radar Resource Management in Multi-Target Tracking", *ISIF Journal of Advances in Information Fusion (JAIF)*, vol. 16, no. 1, pp. 31-47, Jun. 2021.

## 3.1. Stochastic Optimization for Radar Resource Management

Stochastic optimization is widely applied in different fields, such as business, economics and finance, sciences and engineering, as well as health and transportation. In contrast to deterministic optimization, stochastic optimization has much fewer standardized definitions and notations. In [48] Powell gives a general introduction to the field and tries to present the most common stochastic optimization problems and their solutions using a unified framework and notation.

Generally, stochastic optimization is an optimization that includes random variables. The available information about these random variables is used to make decisions that also include possible future events. Therefore, a stochastic optimization approach is directly taking the uncertainty of the problem into account. Stochastic optimization techniques have extensively been covered in literature, e.g., in [49, 50]. An excellent general overview of how stochastic optimization can be applied in a CR or RRM context using the POMDP framework has been presented by Charlish et al. in [51].

The remainder of this chapter is structured as follows. Section 3.2 gives an overview of available methods that apply MDPs and POMDPs for stochastic RRM problems. Furthermore, Section 3.3 gives a short overview of the proposed approach. Section 3.4 introduces the POMDP framework, while Section 3.5 gives an overview of common POMDP solution methods. Section 3.6 introduces the assumed radar scenario. In Section 3.7, the results of the OSB and the AODB algorithm are compared in a simplified LTI scenario. Section 3.8 shows the simulation results for a dynamic radar-related scenario. It is solved by applying the AODB algorithm and optimizing both dwell time and revisit interval. Section 3.9 gives an overview of the performance of the AODB algorithm compared to other approaches and finally, Section 3.10 contains the conclusions.

## 3.2. Markov Decision Processes in Radar Resource Management

Markov Decision Processes (MDPs) and POMDPs are attractive frameworks for modeling and solving stochastic RRM problems. They use a number of states to formulate a dynamic control problem in which the optimal actions can be found through optimizing a cost or reward function. A depiction of an MDP is shown in Figure 3.1.

Those frameworks have been applied to single tasks, for instance, by Charlish and Hoffmann in [53] or by Krishnamurthy in [54]. Both methods optimize the time between consecutive measurement operations. Charlish and Hoffmann are considering a radar tracking example, where the track quality needs to be optimized. At the same time, Krishnamurthy presents a more general sensor scenario where the measurement performance is optimized regarding false alarm rate and the quality of the estimate. The former approach applied PR, while the latter used a stochastic dynamic programming algorithm. The PR evaluates each possible action by calculating the expected cost over a certain future horizon using Monte Carlo sampling (see Figure 3.2).

More specifically, PR simulates the chosen action to be evaluated and a possible observation following it. Subsequently, the expected situation of the environment is calculated

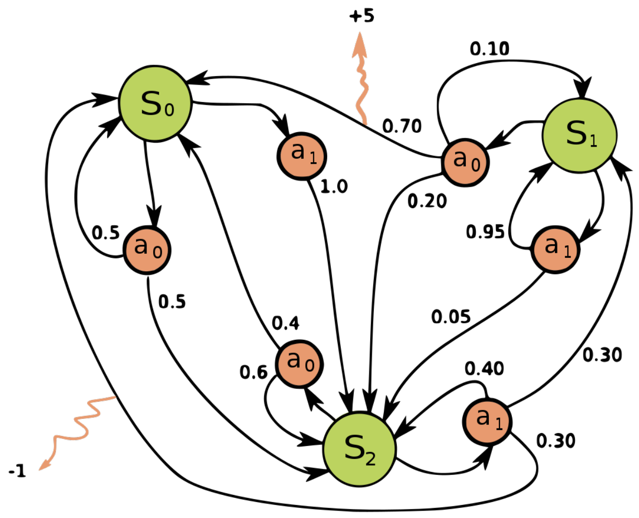


Figure 3.1: Depiction of an MDP [52].

for the next time step, and a new action and observation pair is simulated based on a predefined action schedule called Base Policy (BP). PR keeps simulating the future situation and executing the actions defined in the BP until a certain horizon is reached. One of those simulation strings is called a rollout. Based on the output of all the steps in one rollout, an expected cost can be calculated for the first action simulated in the rollout. Since PR is a Monte Carlo method, more rollouts per evaluated action will be more accurate. PR is potentially interesting due to its straightforward concept and the fact that it will deliver the optimal result if infinite Monte Carlo samples are evaluated.

Two other approaches show how radar actions can be determined by applying Reinforcement Learning (RL) [55] and deep RL [56] to solve an underlying MDP. In their papers, both Selvi et al. and Thornton et al. optimize the sensing strategies for a single target while a communication signal uses the same frequency bands. Both publications show that the optimal policy can improve the performance despite the presence of the interferer. Another approach by Pulkkinen et al. solves an MDP using RL in order to optimize the time budget of an MFR in a tracking scenario [57]. RL is an interesting approach to RRM, but it is often not feasible because of the enormous state space of many RRM problem formulations. In such a case, the algorithm's training would need an enormous amount of data and a lot of computation time. Additionally, the success of RL or other Machine Learning (ML) approaches highly depends on the data used by the algorithm to learn. If the amount of data is not sufficient, not appropriate, or a wrong model is assumed, the performance of ML approaches will deteriorate tremendously.

Constrained (PO)MDPs have been proposed to solve multi-task control problems, where the constraint(s), among others, can represent the limit on the available resources or budgets for all the tasks. Possible applications are radar networks or single radars with multiple tasks. The computational complexity of these problems is potentially large. It has been suggested to decouple the main optimization problem into smaller and easier to solve sub-problems

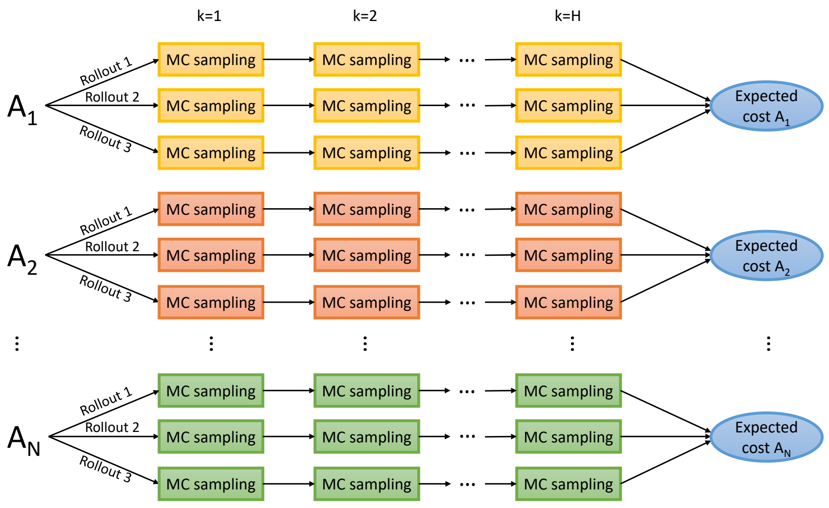


Figure 3.2: Block scheme of a PR. Each action A is evaluated through multiple rollouts. Each of those rollouts is a series of Monte Carlo sampling steps until the rollout horizon H is reached. At the end, the cost of all rollouts for each action is evaluated to calculate the expected cost for that action.

by the use of LR. One LR approach for sensor networks with an energy-constraint on the inter-sensor communication has been published by Williams et al. in [58]. Some notable LR approaches for multi-task radar scenarios are, e.g., [35] by Wintenby and Krishnamurthy and [36] by White and Williams. Wintenby and Krishnamurthy apply a Markov chain consisting of performance states for each tracking task and solve it with a combination of LR and approximate dynamic programming. White and Williams assume a discretized state space and a fully observable MDP, which they solve by using dynamic programming. In addition to that, Castãnón applies LR in combination with a constrained POMDP for multi-object classification in [37]. The chosen POMDP solution method in that approach is the so-called Witness algorithm. As an alternative to LR, one could also consider Q-RAM in combination with POMDPs (see Chapter 2).

Another interesting approach for applying POMDPs for RRM has been introduced by Krishnamurthy and Djonin in [59] where they divide the RRM algorithm into ”Sensor Micromanagement” and ”Sensor Macromanagement”. The former is formulated as a POMDP and determines when the resource allocation has to be updated. There is always one task that receives a high resource allocation, while the others receive a lower one. The macromanagement, on the other hand, decides which target will get the highest priority and therefore the highest resource allocation. A block scheme depiction of this approach is shown in Figure 3.3. This process is based on the realized cost of the micromanagement and some heuristic rules. This research, on the other hand, aims to combine micro- and macromanagement. The resource distribution is defined directly through the cost function and without any heuristic functions. In addition to that, the budgets of the tasks can gradually change over time, contrarily to the approach of Krishnamurthy and Djonin, where only two different actions exist.

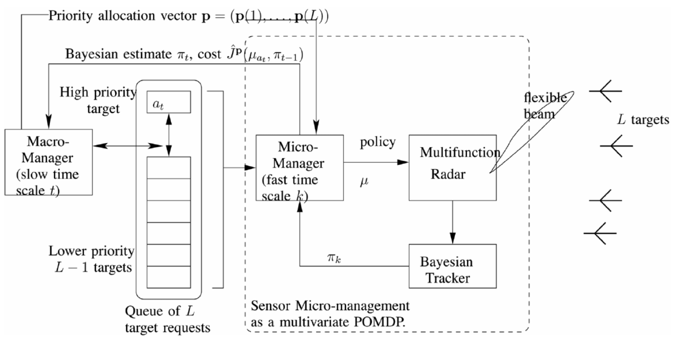

Priority allocation vector $\mathbf{p} = (\mathbf{p}(1), \ldots, \mathbf{p}(L))$
Bayesian estimate $\pi_t$, cost $\hat{J}^{\mathbf{p}}(\mu_{a_t}, \pi_{t-1})$
High priority target
$a_t$
Macro-Manager (slow time scale $t$)
Micro-Manager (fast time scale $k$)
policy
$\mu$
Multifunction Radar
flexible beam
$L$ targets
Lower priority $L-1$ targets
$\pi_k$
Bayesian Tracker
Queue of $L$ target requests
Sensor Micro-management as a multivariate POMDP.

Figure 3.3: A block scheme of the approach by Krishnamurthy and Djonin [59].

Although the approach by Krishnamurthy and Djonin is interesting, it still works with heuristic priorities and a very small amount of possible actions. In contrast, this thesis combines Micro- and Macromanagement into one method without explicitly defining priorities. Additionally, the algorithm can choose from a large number of different actions to optimize the performance.

Furthermore, MDP and POMDP frameworks are often applied in the context of artificial intelligence and robotics. In those contexts, the problem is often formulated to give the lowest cost for the action, increasing the information available to the so-called agent. For instance, specific sensor actions can be performed to reduce the agent's uncertainty about the current situation and simultaneously increase the information. In a robotic context, the agent might also adjust its position to observe the environment better, generating more information. Such approaches are not only using the POMDP's state to determine the cost but also the belief that the agent has about the state it is in. Some approaches that formulate the POMDP in such a way can be found in [60–63]. In an RRM context, such formulations are very interesting, as the sensor usually cannot directly impact the state of the POMDP but rather try to increase the available information on the state based on the sensor actions.

## 3.3. High-Level Description of the Proposed Approach

This chapter follows the general RRM problem definition as shown in Chapter 2.2. The details about the assumptions for the simulations of this chapter are presented in Section 3.6.

### 3.3.1. The Cost Function

The development of specific cost functions is essential and will be a development task in itself that will require close cooperation with potential users. However, as already mentioned in Chapter 2, the development of such user-specific cost functions is out of the scope of this thesis. For the simulations, some example cost functions will be presented that are by no means supposed to be perfect.

### 3.3.2. Proposed Approach

In this chapter, the RRM problem is considered a multi-task time budget-constrained control problem, where the individual tasks are different tracking tasks. The chosen problem formulation directly leads to the assumption of a constrained POMDP.

The optimal balancing of sensor budgets in an LTI setting by using the OSB algorithm has already been shown in Chapter 2. It applies LR to distribute the resources over the different tasks. Subsequently, generic dynamical problems have been considered by utilizing the POMDP framework, and this chapter introduces the Approximately Optimal Dynamic Budget Balancing (AODB) algorithm with a cost function based on the predicted error-covariance of the KF. It has been shown that the results of the AODB algorithm are approximately optimal with respect to the steady-state error-covariance of a KF. The RRM problem was solved non-myopically using PR, an online Monte Carlo technique, which stochastically predicts the future. The results of this chapter have also been published in [64].

## 3.4. Definition of a POMDP

A POMDP describes an MDP for which the state cannot be observed directly. Instead, an observation is taken, which generates a probability distribution over the possible states. This is called the belief state. Based on the belief state and the knowledge of the underlying MDP, a POMDP allows solving optimization problems non-myopically, meaning that it takes the expected future into account. In the following, the time is assumed to be discretized in intervals $k$ with length $T$, the time between two consecutive observations. This introduction to POMDPs is intentionally kept very general. It, therefore, also does not contain any explicit constraints.

Generally, a POMDP is defined by the following parameters (see for example [65] and [66]):

**State space $\boldsymbol{S}$:** Consists of all possible states that can be reached within the process, see (2.2). At time step $k$ the state is defined as $\boldsymbol{s}_k$. Based on the underlying states and the received observations, the belief-state defines a probability distribution over all possible states. It is defined as $\boldsymbol{b}_k$.

**Action space $\boldsymbol{A}$:** Consists of all possible actions within the process. Each action has a certain cost defined by the cost function. The action at time step $k$ is denoted $\boldsymbol{a}_k$.

**Observation space $\boldsymbol{Z}$:** Consists of all possible observations that can be received within the process. An observation at time step $k$ it is defined as $\boldsymbol{z}_k$.

**Transition probability $\boldsymbol{\Psi}(\boldsymbol{s}_k, \boldsymbol{s}_{k+1}, \boldsymbol{a}_k)$:** The probability function $p(\boldsymbol{s}_{k+1}|\boldsymbol{s}_k, \boldsymbol{a}_k)$ that defines the probability of transitioning from state $\boldsymbol{s}_k$ to state $\boldsymbol{s}_{k+1}$ given action $\boldsymbol{a}_k$. Note: In this chapter, the state transition probability does not depend on the chosen action, while the belief state transition does depend on it.

**Probability of observation $\boldsymbol{O}(\boldsymbol{z}_k, \boldsymbol{s}_{k+1}, \boldsymbol{a}_k)$:** The probability function $p(\boldsymbol{z}_k|\boldsymbol{s}_{k+1}, \boldsymbol{a}_k)$ that defines the probability to receive a certain observation $\boldsymbol{z}_k$ when executing action $\boldsymbol{a}_k$ with the resulting state being $\boldsymbol{s}_{k+1}$.

**Cost function $c(\boldsymbol{s}_k, \boldsymbol{a}_k)$:** The immediate cost of executing action $\boldsymbol{a}_k$ in state $\boldsymbol{s}_k$. Note: In this chapter the cost function does not directly depend on the state.

**Discount factor $\gamma$:** A discount factor that discounts future time steps w.r.t. the present.

Note: in this chapter the discount factor is always set to $\gamma = 1$.

POMDPs can be solved for finite or infinite horizons. Infinite horizons use the whole future to determine the optimal actions. As the very distant future probably has a negligible impact on the current situation, the future steps are often discounted using some factor. On the other hand, finite horizons only look a certain amount of steps ahead into the future. The advantage is that the computational load is finite as well. Therefore, approaches with finite horizons are usually easy to implement in practice. The main disadvantage of finite horizons is that the resulting policy is suboptimal, i.e., an approximation of the policy obtained when applying an infinite horizon.

For a finite horizon, the value of $\mathcal{H}$ represents the number of considered measurement time steps into the future. Commonly, finite horizons are implemented as receding horizons, which means that every time a new budget allocation is calculated, the horizon $\mathcal{H}$ will be shifted and reapplied from the current moment in time. This thesis uses receding horizons to limit the computational load.

In [53], Charlish and Hoffmann have written a very clear summary of the general solution of a POMDP, which is used as a base for the following equations. The goal is to find the actions that minimize the total cost (value $V_{\mathcal{H}}$ over horizon $\mathcal{H}$). Starting at time step $k_0$ this can be expressed as

$$V_{\mathcal{H}} = E\left[\sum_{k=k_0}^{k_0+\mathcal{H}} c(\boldsymbol{s}_k, \boldsymbol{a}_k)\right]. \tag{3.1}$$

Using $C_B(\boldsymbol{b}_k, \boldsymbol{a}_k) = \sum_{\boldsymbol{s}\in\boldsymbol{S}} \boldsymbol{b}_k(\boldsymbol{s})c(\boldsymbol{s}, \boldsymbol{a}_k)$ being the expected cost given belief state $\boldsymbol{b}_k$, $V_{\mathcal{H}}$ can be written as a so-called value function of the belief state $\boldsymbol{b}_{k_0}$ at time step $k_0$:

$$V_{\mathcal{H}}(\boldsymbol{b}_{k_0}) = E\left[\sum_{k=k_0}^{k_0+\mathcal{H}} C_B(\boldsymbol{b}_k, \boldsymbol{a}_k)|\boldsymbol{b}_{k_0}\right]. \tag{3.2}$$

For belief state $\boldsymbol{b}_0$ and taking action $\boldsymbol{a}_0$, the optimal value function is defined according to Bellman's equation [67] as

$$V^*_{\mathcal{H}}(\boldsymbol{b}_0) = \min_{\boldsymbol{a}_0\in\boldsymbol{A}}\left(C_B(\boldsymbol{b}_0, \boldsymbol{a}_0) + \gamma \cdot E\left[V^*_{\mathcal{H}-1}(\boldsymbol{b}_1)|\boldsymbol{b}_0, \boldsymbol{a}_0\right]\right). \tag{3.3}$$

For very long or infinite horizons, the discount factor can be set to $\gamma < 1$. Using this equation, the optimal policy can be expressed as

$$\pi^*_0(\boldsymbol{b}_0) = \underset{\boldsymbol{a}_0\in\boldsymbol{A}}{\arg\min}\left(C_B(\boldsymbol{b}_0, \boldsymbol{a}_0) + \gamma \cdot E\left[V^*_{\mathcal{H}-1}(\boldsymbol{b}_1)|\boldsymbol{b}_0, \boldsymbol{a}_0\right]\right). \tag{3.4}$$

For each $\boldsymbol{b}_k$ and $\boldsymbol{a}_k$ the optimal so-called Q-value is then defined as

$$Q_{\mathcal{H}-k}(\boldsymbol{b}_k, \boldsymbol{a}_k) = C_B(\boldsymbol{b}_k, \boldsymbol{a}_k) + \gamma \cdot E\left[V^*_{\mathcal{H}-k-1}(\boldsymbol{b}_{k+1})|\boldsymbol{b}_k, \boldsymbol{a}_k\right]. \tag{3.5}$$

Another way to find the optimal policy is to find the action $\boldsymbol{a}_k$ that minimizes the optimal Q-value:

$$\pi^*_k(\boldsymbol{b}_k) = \underset{\boldsymbol{a}_k\in\boldsymbol{A}}{\arg\min}(Q_{\mathcal{H}-k}(\boldsymbol{b}_k, \boldsymbol{a}_k)). \tag{3.6}$$

Therefore, it is necessary to calculate the Q-value for all possible actions, which is generally infeasible.

## 3.5. Solution Methods for POMDPs

For solving a POMDP, there are both online, as well as offline approaches. The choice of the type of these methods usually depends on the size of the state space. The so-called state space explosion limits the usefulness of exact offline techniques.

Most offline methods are based on the so-called Value Iteration (VI), which iteratively calculates the cost/reward values of all possible states. There are exact approaches to VI (e.g., One-Pass algorithm [68]), as well as approximate point-based algorithms (e.g., Point-Based Value Iteration (PBVI) or Perseus [69]). The former techniques often lead to highly complex optimization problems. In contrast, the latter require many grid points within the state space (and therefore much memory and computational effort) to converge towards the exact solution. The advantage of offline solutions is that the POMDP is solved only once, and the solution is always valid afterward. Unfortunately, those methods are already infeasible for a small dimensional state space.

Contrary to that, online algorithms only solve a small part of the POMDP that is relevant at the current moment. This makes them less accurate but much easier and faster to compute. Some of the online approaches involve approximate tree methods (see, for example, the overview in [65]) or Monte Carlo sampling (e.g., PR).

Since an exact and complete solution of the POMDP is usually infeasible in real scenarios, this chapter focuses on implementing PR as an approximate solution. The general structure of the proposed algorithm is illustrated in Figure 3.4. The outputs of the algorithm are the converged budgets for each task.

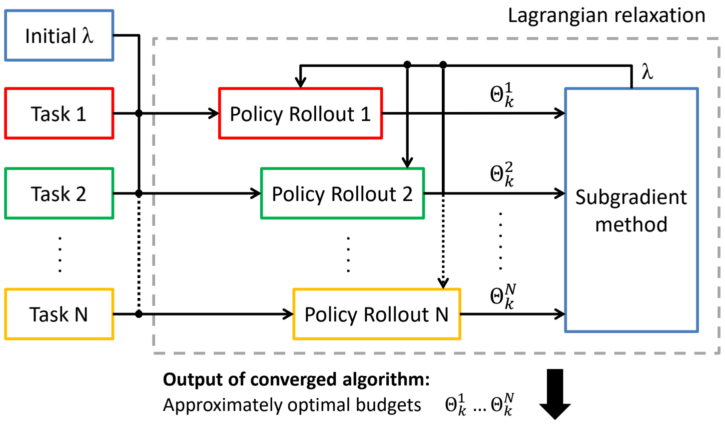


Figure 3.4: High level block scheme of the proposed algorithm.

### 3.5.1. Policy Rollout for POMDPs

The PR technique takes Monte Carlo samples of the expected future, which means that it stochastically explores the possible future actions and the according costs. Observations and belief states are generated from a given initial belief state and a given candidate action within a rollout. There is a rollout evaluation per action $\boldsymbol{a}$ in the action space $\boldsymbol{A}$. The candidate action is taken in the first step of the rollout, while a so-called BP $\pi_{base}$ is used for every following step, until the horizon $\mathcal{H}$ is reached. In each rollout, the total cost is summed up. This procedure is repeated $M$ times, and then the summed cost of all $M$ rollouts is averaged. This is the expected cost of the evaluated action. The action that produced the lowest expected cost is chosen for the next time step. It has been shown that PR leads to a policy that is at least as good as the BP with a very high probability if enough samples are provided [30]. The choice of the BP and the amount of samples to be taken are therefore crucial to the algorithm's performance. The number of samples is equivalent to the number of rollouts $M$ per action used to average the cost. In other words, one sample is the evaluation of one possible future. Finding a good BP for a radar scenario is no trivial task. For example, one could think about using information from previously experienced situations that were similar to the current one. If the executed actions from the last run have been saved, they can be reused again to improve the policy further. This could be considered in the context of RL, for instance. Unfortunately, it is not very likely to experience the exact same situation multiple times if a huge state space is assumed, so the usefulness in such a case is questionable (see also the remark about RL in Section 3.2). Another straightforward choice of the BP could be an equal resource allocation to all the tasks. PR has been covered extensively, e.g. by Bertsekas in [28–30].

The PR can be expressed mathematically as shown in (3.7) and (3.8). The Q-value is defined as

$$Q_{\mathcal{H}-k}^{\pi_{base}}(\boldsymbol{b}_k, \boldsymbol{a}_k) = C_B(\boldsymbol{b}_k, \boldsymbol{a}_k) + E\left[V_{\mathcal{H}-k-1}^{\pi_{base}}(\boldsymbol{b}_{k+1}) | \boldsymbol{b}_k, \boldsymbol{a}_k\right], \tag{3.7}$$

where $E[\cdot]$ is the expectation. The best policy can then be found by applying

$$\pi_k(\boldsymbol{b}_k) = \underset{\boldsymbol{a}_k \in \boldsymbol{A}}{\arg\min}(Q_{\mathcal{H}-k}^{\pi_{base}}(\boldsymbol{b}_k, \boldsymbol{a}_k)). \tag{3.8}$$

PR does not necessarily lead to the optimal policy. It rather aims at improving the chosen BP $\pi_{base}$.

## 3.6. Assumed Radar Scenario

For the rest of the chapter, a two-dimensional radar tracking example is assumed that will be solved using the AODB algorithm. Measurements are taken in range, angle, and possibly radial velocity. The algorithm jointly optimizes the revisit interval $T$ (the time between two consecutive measurements) and the dwell time $\tau$ (the time the sensor spends focused on a target). The algorithm calculates the budgets of all tasks and makes sure that they fit into the time frame but does not create an explicit schedule. Therefore, the assumed measurements are taken independently of each other and can be overlapping in time. In order to put all tasks into a single timeline, an explicit scheduler needs to be implemented on a lower level. At which moments this budget calculation is performed depends on the preferences of the user. In the following, the assumptions of the assumed radar scenario are explained in more detail.

## 3.6.1. Assumed Radar Systems

In the simulations, two different sets of system parameters are assumed, as shown in Table 3.1. The table shows the zero-mean Gaussian measurement noise variances for range ($\sigma^2_{r,0}$), azimuth angle ($\sigma^2_{\theta,0}$), and radial velocity ($\sigma^2_{d,0}$) w.r.t. the measurement of a reference target which is sensed with an SNR of 1. The parameters of the reference measurement are shown in Table 3.2 and are valid for all simulations that are presented in this chapter. Radar System A measures range and azimuth only, while System B is able to measure radial velocity as well. The values of the variances in Table 3.1 are chosen somewhat arbitrarily. It is not intended to compare the different radar systems but rather to show how the AODB algorithm can universally be applied to different systems.

Table 3.1: System parameters of the assumed radar systems with respect to the reference measurement.

| **System** | **Measurement** | $\boldsymbol{\sigma^2_{r,0}}$ [m$^2$] | $\boldsymbol{\sigma^2_{\theta,0}}$ [rad$^2$] | $\boldsymbol{\sigma^2_{d,0}}$ [m$^2$ s$^{-2}$] |
|---|---|---|---|---|
| A | r/θ | 625 | 4e-4 | - |
| B | r/θ/d | 2500 | 2e-4 | 25 |

Table 3.2: Parameters of reference measurement.

| **SNR** (SNR$_0$) | **RCS** ($\varsigma_0$) | **Dwell time** ($\tau_0$) | **Range** ($r_0$) |
|---|---|---|---|
| 1 | 10 m$^2$ | 1 s | 50 km |

## 3.6.2. Velocity Model

The velocity and maneuverability of the targets is assumed to be constant. Between two resource allocation updates the actions are assumed to stay unchanged. The action vector $\boldsymbol{a}_n \in \mathbb{R}^2$ consists of the dwell time and the revisit interval. The latter defines the time between the measurements of target $n$. In contrary to previous publications, in this chapter the revisit interval $T_n$ and the dwell time $\tau_n$ are optimized jointly. The revisit intervals with length $T_n$ are depending on the targets and are therefore denoted as $k_n$. Considering this, (2.3) can explicitly be written as

$$\boldsymbol{s}^n_{k_n+1} = \boldsymbol{F}_n \boldsymbol{s}^n_{k_n} + \boldsymbol{w}^n_{k_n}, \tag{3.9}$$

with $\boldsymbol{F}_n \in \mathbb{R}^{4\times 4}$ defined as

$$\boldsymbol{F}_n = \begin{bmatrix} 1 & 0 & T_n & 0 \\ 0 & 1 & 0 & T_n \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.10}$$

and the zero-mean Gaussian maneuverability noise $\boldsymbol{w}^n$ with covariance

$$\boldsymbol{Q}_n = \begin{bmatrix} T_n^4/4 & 0 & T_n^3/2 & 0 \\ 0 & T_n^4/4 & 0 & T_n^3/2 \\ T_n^3/2 & 0 & T_n^2 & 0 \\ 0 & T_n^3/2 & 0 & T_n^2 \end{bmatrix} \sigma_{w,n}^2, \tag{3.11}$$

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

Because of the non-linear relationship between measurements and states, an EKF is applied. The corresponding observation matrix $\boldsymbol{H}_{k_n}^n$ is defined as the Jacobian of the measurement transformation function $\boldsymbol{h}$:

$$\boldsymbol{H}_{k_n}^n = \left.\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{s}}\right|_{\boldsymbol{s}_{k_n}^n}. \tag{3.12}$$

It has dimensions $\boldsymbol{H}_{k_n}^n \in \mathbb{R}^{2\times4}$ for System A and $\boldsymbol{H}_{k_n}^n \in \mathbb{R}^{3\times4}$ for System B.

### 3.6.3. SNR Model

In the following examples, sensor measurements in range $(r)$, azimuth $(\theta)$ and radial velocity $(d)$ are assumed. Since the transformation between polar and Cartesian coordinates is non-linear, the measurement equation in (6.1) for target $n$ at time step $k_n$ can be defined as

$$\boldsymbol{z}_{k_n}^n = \boldsymbol{h}(\boldsymbol{s}_{k_n}^n) + \boldsymbol{v}_{k_n}^n, \tag{3.13}$$

where $\boldsymbol{h}(\boldsymbol{s}_{k_n}^n) \in \mathbb{R}^3$ is the measurement transformation function at $\boldsymbol{s}_{k_n}^n$ which for System B is defined as

$$\boldsymbol{h}(\boldsymbol{s}_{k_n}^n) = \left[\sqrt{(x_{k_n}^n)^2 + (y_{k_n}^n)^2}, \operatorname{atan2}\left(y_{k_n}^n, x_{k_n}^n\right), \frac{x_{k_n}^n \dot{x}_{k_n}^n + y_{k_n}^n \dot{y}_{k_n}^n}{\sqrt{(x_{k_n}^n)^2 + (y_{k_n}^n)^2}}\right]^T \tag{3.14}$$

and $\boldsymbol{v}_{k_n}^n \in \mathbb{R}^3$ is the zero-mean Gaussian measurement noise for target $n$. The range, azimuth and radial velocity components of $\boldsymbol{v}_{k_n}^n$ are independent:

$$\boldsymbol{v}_{k_n}^n = [v_{k_n}^{r,n} \quad v_{k_n}^{\theta,n} \quad v_{k_n}^{d,n}]^T, \tag{3.15}$$

with variances $\sigma_{r,n}^2$, $\sigma_{\theta,n}^2$ and $\sigma_{d,n}^2$. In this chapter, the SNR is calculated by using (3.16) which is based on equations by Koch in [70]:

$$\mathrm{SNR}_{k_n}(\varsigma_n, \tau_n, r_{k_n}^n) = \mathrm{SNR}_0 \cdot \left(\frac{\varsigma_n}{\varsigma_0}\right) \cdot \left(\frac{\tau_n}{\tau_0}\right) \cdot \left(\frac{r_{k_n}^n}{r_0}\right)^{-4} \cdot e^{-2\Delta\alpha}, \tag{3.16}$$

where $\Delta\alpha$ is the relative beam positioning error, $\varsigma_n$ is the constant Radar Cross Section (RCS) of the target $n$, $r_{k_n}^n$ is the distance of target $n$ at time step $k_n$ and $\varsigma_0$, $\tau_0$ and $r_0$ are the corresponding values for a reference target. In (3.16), the dwell time is used equivalently to the transmitted energy mentioned by Koch. Similar to the approach in [70], the relative beam positioning error is calculated using

$$\Delta\alpha = \frac{\left(\theta_{k_n} - \hat{\theta}_{k_n}\right)^2}{\Gamma^2}, \tag{3.17}$$

where $\theta_{k_n}$ is the real target angle and $\hat{\theta}_{k_n}$ is the predicted target angle in azimuth at time $k_n$ and $\Gamma$ is the one-sided beam-width in azimuth.

Using (3.16), the variance of the range, azimuth and radial velocity measurement noise for target $n$ can be defined as (see e.g. [71])

$$\sigma^2_{\bullet,n} = \frac{\sigma^2_{\bullet,0}}{\mathrm{SNR}_{k_n}(\varsigma_n, \tau_n, r^n_{k_n})}, \tag{3.18}$$

where $\bullet \in (r, \theta, d)$ and $\sigma^2_{\bullet,0}$ is the measurement noise variance for a reference target 0 as defined in Table 3.1.

Due to the independent measurements, the measurement covariance when using System B can be defined as

$$\boldsymbol{R}^n_{k_n} = \begin{bmatrix} \sigma^2_{r,n} & 0 & 0 \\ 0 & \sigma^2_{\theta,n} & 0 \\ 0 & 0 & \sigma^2_{d,n} \end{bmatrix}. \tag{3.19}$$

## 3.6.4. Cost Function

The assumed cost function is constructed from the predicted error-covariance matrix at time step $k_n + 1$. The current predicted error-covariance matrix $\boldsymbol{P}_{k_n|k_n-1} \in \mathbb{R}^{4\times4}$ at time step $k_n$ can be defined for target $n$ as

$$\boldsymbol{P}_{k_n|k_n-1}(T_n, \tau_n, \boldsymbol{s}^n_{k_n-1}) = \boldsymbol{F}_n \boldsymbol{P}_{k_n-1|k_n-1}(T_n, \tau_n, \boldsymbol{s}^n_{k_n-1}) \boldsymbol{F}^T_n + \boldsymbol{Q}_n, \tag{3.20}$$

where $\boldsymbol{F}_n$ is the transition matrix with interval length $T_n$ as defined in (5.23), $\boldsymbol{P}_{k_n-1|k_n-1} \in \mathbb{R}^{4\times4}$ is the last filtered error-covariance matrix and $\boldsymbol{Q}_n$ is the maneuverability covariance with interval length $T_n$ as defined in (5.24). Based on this, another estimation and prediction cycle is applied to the error-covariance. The result is the error-covariance $\boldsymbol{P}_{k_n+1|k_n} \in \mathbb{R}^{4\times4}$ for time $k_n + 1$:

$$\boldsymbol{P}_{k_n+1|k_n}(T_n, \tau_n, \boldsymbol{s}^n_{k_n}) = \boldsymbol{F}_n \boldsymbol{P}_{k_n|k_n}(T_n, \tau_n, \boldsymbol{s}^n_{k_n}) \boldsymbol{F}^T_n + \boldsymbol{Q}_n, \tag{3.21}$$

The cost function that is used in the following sections is based on this expression.

### 3.6.5. Optimization Problem:

It is assumed that there are $N$ tracked targets in the environment. The RRM problem can thus be expressed as

$$\begin{aligned}\underset{\boldsymbol{T},\boldsymbol{\tau}}{\text{minimize}} \quad & \sum_{n=1}^{N} E\left[c(T_n, \tau_n, \boldsymbol{s}_{k_n}^n, \boldsymbol{P}_{k_n+1|k_n})\right] \\ \text{subject to} \quad & \sum_{n=1}^{N} \frac{\tau_n}{T_n} \leq \Theta_{max},\end{aligned} \tag{3.22}$$

where $\Theta_{max} \in [0,1]$ is the total available budget. The term budget refers to a ratio of dwell time $\tau$ and revisit interval $T$. The expectation $E[\cdot]$ is used due to the fact that the PR evaluates the possible future.

Furthermore, every detection is always correctly assigned to the corresponding target.

## 3.7. Linear Time-Invariant Example

In this section, a simplified LTI scenario is assumed to investigate if the AODB algorithm converges to the same results as given by the OSB algorithm, which is the optimal solution in this case.

### 3.7.1. General Simulation Parameters

Radar System A as mentioned in Table 3.1 is considered. For this simple example, no beam positioning error is taken into account, e.g. due to a very wide beam by using an MFR with a phased array antenna applying DBF on receive. The probability of detection is assumed to be 1. The implemented BP is simply to apply the evaluated action in every step of the PR. Therefore $\pi_{base} = \boldsymbol{a}$. A constant LR step size is applied in all simulations. Within the PR, the expected future cost is simulated over the defined horizon for each possible action. The action that produces the lowest expected cost will be chosen for the measurements during the next time steps. No additional random movement (process noise) is considered within the PR. For the simulations in this section, the sum of the predicted error-covariance for the position in $x$ and $y$ direction is applied as cost function. To avoid choosing parameters that are impractical in a real application, an extra term is added that penalizes small values of $T$ (see also Chapter 2). Using (3.21) this can be expressed as

$$c(T_n, \tau_n, \boldsymbol{s}_{k_n}^n, \boldsymbol{P}_{k_n+1|k_n}) = \text{trace}\left(\boldsymbol{U}\boldsymbol{P}_{k_n+1|k_n}(T_n, \tau_n, \boldsymbol{s}_{k_n}^n)\boldsymbol{U}^T\right) + \frac{1000}{(T_n)^2}, \tag{3.23}$$

where

$$\boldsymbol{U} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{3.24}$$

is the selection matrix that selects the upper left 2-by-2 submatrix from the error-covariance matrix.

Table 3.3 shows general simulation parameters. The initial Lagrange multiplier value is set to 1. The budgets are recalculated every $t_B = 5$ s. Within the PR, the expected future is simulated and evaluated for each possible action. The radar is always positioned at the origin of the Cartesian coordinate system.

Table 3.3: General simulation parameters of LTI scenario.

| Parameter | Value |
|---|---|
| Precision of LR ($\delta_{LR}$): | 0.001 |
| Action discretization ($\Delta T$, $\Delta\tau$): | 0.0025 s |
| Number of rollouts ($\mathcal{N}_r$): | 10 |
| Rollout horizon ($\mathcal{H}$): | 10 steps |
| Base Policy ($\pi_{base}$) | $\boldsymbol{a}$ |
| Maximum budget ($\Theta_{max}$): | 1 |
| Budget update ($t_B$): | 5 s |
| Beam positioning error ($\Delta\alpha$): | 0 |
| Probability of detection ($P_D$): | 1 |

### 3.7.2. Comparison OSB and AODB

In order to prove the validity of the proposed AODB algorithm, a comparison is conducted with the OSB algorithm as proposed in Chapter 2. The OSB algorithm calculates the optimal steady-state error-covariance given a revisit interval $T$ and a dwell time $\tau$ by using equations by Kalata in [45] and by Gray and Murray in [46]. It is used as explained in Chapter 2 with the general simulation parameters from Table 3.3.

For the comparison, System A and five target tracking tasks are considered with the parameters shown in Table 3.4. The revisit interval $T$ and the dwell time $\tau$ are discretized in steps of 0.0025 s. It is assumed that the budget values are recalculated every 5 seconds. In between, measurements of the targets are taken with the previously calculated revisit intervals $T_n$ and dwell times $\tau_n$. The tracks are assumed to be initialized at the beginning of the simulation.

Since the steady-state solution of the OSB algorithm is only valid for a single dimension, it is assumed that the targets are all positioned at the same position, and the system knows the exact azimuth angle. All targets are static, and only the RCS is considered to be different.

The simulation results are shown Figure 3.5a. It can be seen that the budget allocations $\Theta^n = \tau_n/T_n$ converge to results that are very close to the values that have been determined with the OSB algorithm.

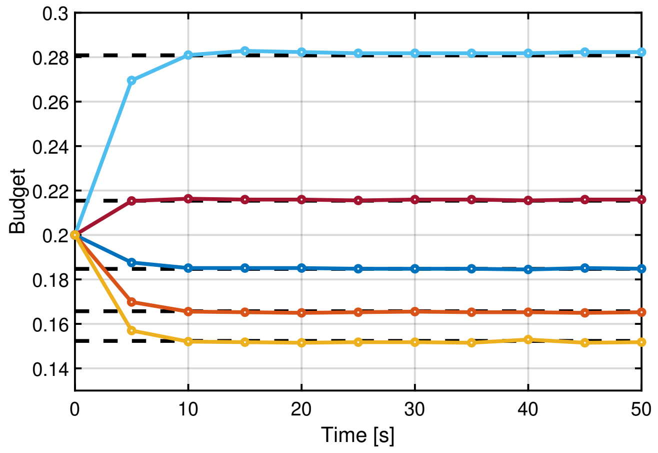
Theoretically, the AODB algorithm should work with any number of tasks. In order to demonstrate that, the above simulation has been repeated with 10 tasks. Equivalent to targets 1 to 5, the RCS values of targets 6 to 10 are increasing in steps of 10 m$^2$. Figure 3.5b shows the approximately optimal budget distribution.

## 3.8. Dynamic Radar Example

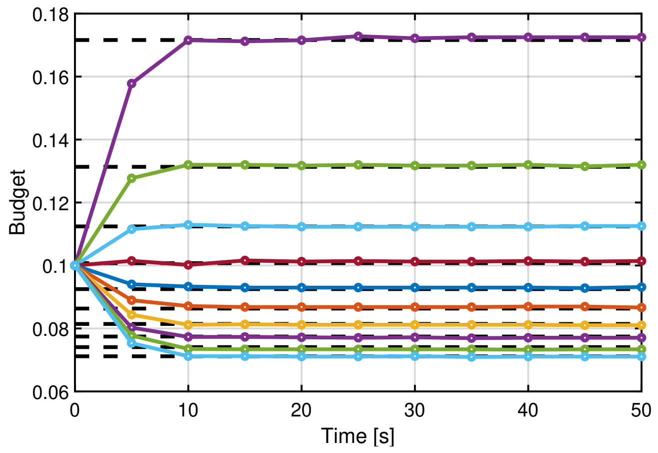
In this section, the performance of the AODB algorithm is investigated in a more realistic radar-related example with different system parameters.

Table 3.4: Initial target parameters for LTI scenario.

| **Target** $n$ | $x_0^n$ [km] | $y_0^n$ [km] | $\dot{x}_0^n$ [m s$^{-1}$] | $\dot{y}_0^n$ [m s$^{-1}$] | $\sigma_w^2$ [m$^2$ s$^{-4}$] | $\varsigma_n$ [m$^2$] |
|---|---|---|---|---|---|---|
| 1 | 50 | 0 | 0 | 0 | 25 | 10 |
| 2 | 50 | 0 | 0 | 0 | 25 | 20 |
| 3 | 50 | 0 | 0 | 0 | 25 | 30 |
| 4 | 50 | 0 | 0 | 0 | 25 | 40 |
| 5 | 50 | 0 | 0 | 0 | 25 | 50 |



(a) Lines from top to bottom: Targets 1 to 5.



(b) Lines from top to bottom: Targets 1 to 10.

Figure 3.5: Budget per task over time after initialization of AODB algorithm for 5 and for 10 targets. Solid lines: results from AODB. Dashed lines: Optimal steady-state results from OSB.

## 3.8.1. General Simulation Parameters

The cost function as introduced in (3.23) is applied. Table 3.5 shows general simulation parameters for these simulations. The initial Lagrange multiplier value is set to 1. The budgets are recalculated every $t_B = 5$ s and measurements are taken in between with the currently calculated resource allocations. The BP is executing the evaluated action in every step of the PR horizon ($\pi_{base} = \boldsymbol{a}$). Within the PR, the expected future is simulated and evaluated for each possible action. The radar is always positioned at the origin of the Cartesian coordinate system.

Table 3.5: General simulation parameters of dynamic scenario.

| **Parameter** | **Value** |
|---|---|
| Precision of LR ($\delta_{LR}$): | 0.001 |
| Action discretization ($\Delta T$, $\Delta\tau$): | 0.0025 s |
| Number of rollouts ($\mathcal{N}_r$): | 5 |
| Rollout horizon ($\mathcal{H}$): | 15 steps |
| Base Policy ($\pi_{base}$) | $\boldsymbol{a}$ |
| Maximum budget ($\Theta_{max}$): | 1 |
| Budget update ($t_B$): | 5 s |
| Beam positioning error ($\Delta\alpha$): | 0 |
| Probability of detection ($P_D$): | 1 |

## 3.8.2. Dynamic Radar Scenario for $P_D = 1$

A dynamic scenario with five moving targets is considered in this simulation. The initial target parameters are given in Table 3.6 and are valid at the moment when the corresponding track is started. Their trajectories are shown in Figure 3.6. The simulation is conducted with System A and B separately. As in the LTI simulations of Section 3.7, no beam positioning error is taken into account, e.g., due to a very wide beam by using an MFR with a phased array antenna applying DBF on receive. The probability of detection is assumed to be 1. A horizon of $\mathcal{H} = 15$ is assumed. Targets 1 to 4 are tracked from the beginning, while Target 5 joins as a new track after 25 s. After 60 s, the total budget is reduced to $\Theta_{max} = 0.9$. The reason for this could be that an operator manually assigned 10 percent of the budget to another task, for instance. At 90 s the maneuverability variance of Target 1 increases by a factor of 36 to a value of 900 $\mathrm{m}^2\,\mathrm{s}^{-4}$, which is known to the system in advance, for instance through some knowledge of the environment. The simulation results for System A can be found in Figures 3.7a and 3.7c, where the former shows the resource distribution over the tasks over time and the latter shows the amount of LR iterations that was needed for convergence. The corresponding simulation results for System B are shown in Figure 3.7b and 3.7d.

The algorithm manages to calculate the budget for both systems while adjusting to unknown and known changes. Before the known variance change at 90 seconds, the algorithm already gradually increases the budget for Target 1. The algorithm delivers very similar but still different solutions for the two systems. It can be seen that the amount of LR iterations needed until convergence stays low unless more significant changes in the situation are taking place. The maximum is 66 iterations for a single resource allocation calculation for the chosen parameters, assuming System B. Using System A leads to similar peak values.

Apart from the impact of the three mentioned sudden changes that are applied to the system, it is also evident that there seems to be a certain dependence of the budgets on the range. While the budget assigned to Targets 1 and 2 stays roughly constant between the different events, Target 3 gets an increasing amount of resources assigned, while the resources of Targets 4 and 5 are decreasing. This is because Target 1 and 2 are moving roughly perpendicular to the radar, while Target 3 is moving away from it, and Targets 4 and 5 are moving towards it. In Section 3.8.4 this effect is investigated with an extra simulation.

Table 3.6: Initial target parameters for dynamic scenario.

| **Target** $n$ | $x_0^n$ [km] | $y_0^n$ [km] | $\dot{x}_0^n$ [m s$^{-1}$] | $\dot{y}_0^n$ [m s$^{-1}$] | $\sigma_w^2$ [m$^2$ s$^{-4}$] | $\varsigma_n$ [m$^2$] |
|---|---|---|---|---|---|---|
| 1 | 12 | 10 | 9 | -15 | 25 | 20 |
| 2 | 12 | 15 | -30 | 15 | 25 | 20 |
| 3 | 7 | 11 | 45 | 30 | 64 | 10 |
| 4 | 19 | 2 | -35 | 0 | 64 | 10 |
| 5 | 10 | 11 | -20 | -25 | 64 | 10 |

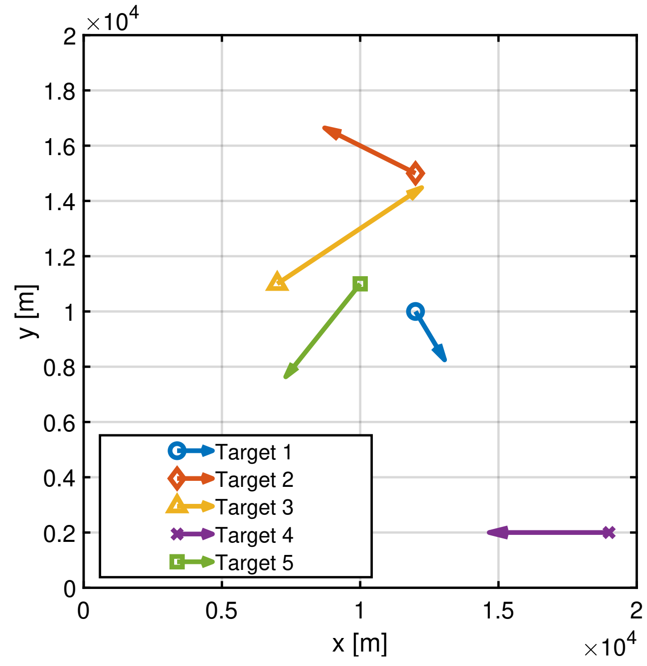
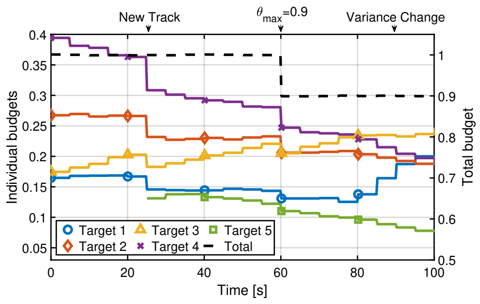
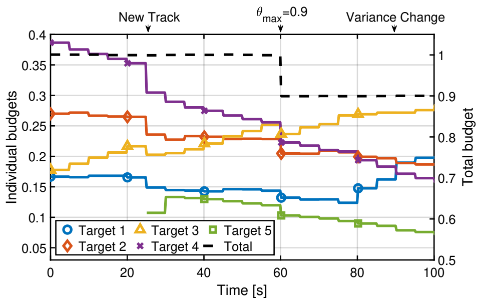


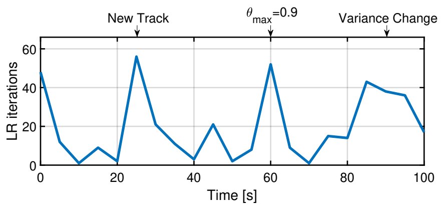
Figure 3.6: Trajectories of targets in dynamic scenario.



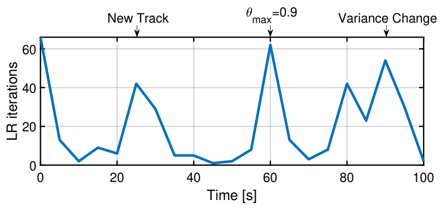
(a) Resource allocations using radar System A.



(b) Resource allocations using radar System B.



(c) Number of LR iterations using radar System A.



(d) Number of LR iterations using radar System B.

Figure 3.7: Dynamic scenario simulation results for Systems A and B. A few markers are added to help to distinguish the allocations for the different targets.

### 3.8.3. Dynamic Radar Scenario for $P_D < 1$

In a real situation, a low SNR can lead to missed detections. In addition to that, the used radar system might not be able to transmit with a wide beam and apply DBF on receive. Therefore, another simulation is presented in this subsection that considers a probability of detection based on the calculated SNR and the beam positioning error. The scenario is identical with the one shown in Section 3.8.2 and apart from the probability of detection and the beam positioning error, all values in Table 3.5 are applied. The SNR is calculated using (3.16) and taking into account a beamwidth of 2°. In addition to that, a measurement in the simulation as well as in the PR is only generated with the probability of detection [70]

$$P_{D,k_n} = P_{FA}^{\frac{1}{1+\mathrm{SNR}_{kn}}}, \tag{3.25}$$

where $P_{FA} = 10^{-4}$ is the constant probability of false alarm. It is assumed that the false alarms do not influence the tracks. The result of this simulation assuming System A can be found in Figures 3.8 and 3.9.

It can be seen that the resulting budget allocations are less smooth than in the simulations assuming $P_D = 1$. Still, the AODB algorithm leads to comparable results even though some of the detection probabilities are quite low.

### 3.8.4. Analysis of the Impact of the Chosen Cost Function

To show the impact of the range on the resource distribution by the AODB algorithm, another simulation has been conducted with three targets. Target 1 has the initial parameters

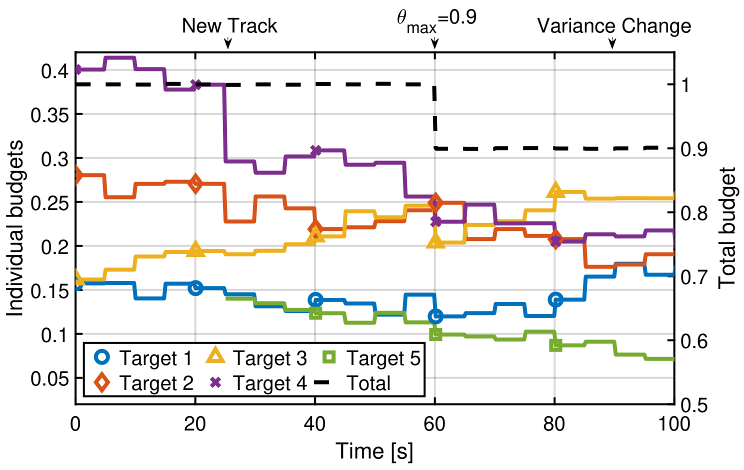


Figure 3.8: Dynamic scenario simulation using radar System A with $P_D < 1$. A few markers are added to distinguish the allocations for the different targets.

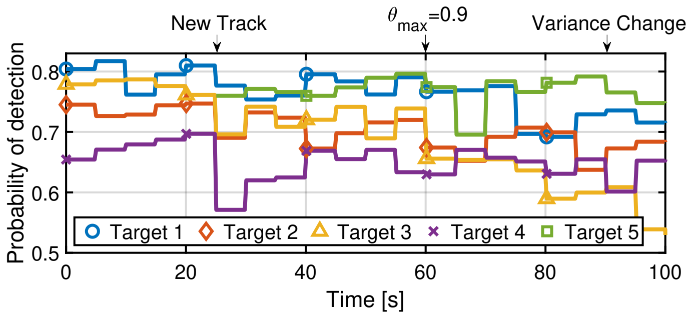


Figure 3.9: Average probability of detection per budget update interval for the dynamic scenario with $P_D < 1$. A few markers are added to distinguish the allocations for the different targets.

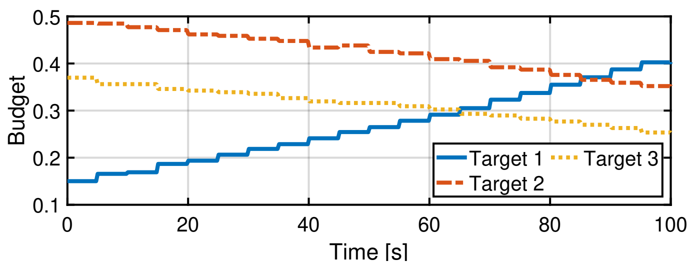


Figure 3.10: Budget allocation for two static and one moving target.

$x_0 = 6\,\mathrm{km}$, $y_0 = 6\,\mathrm{km}$, $\dot{x}_0 = 50\,\mathrm{m\,s^{-1}}$ and $\dot{y}_0 = 50\,\mathrm{m\,s^{-1}}$. Targets 2 and 3 are static at positions $x = 12.4\,\mathrm{km}$, $y = 9\,\mathrm{km}$ and $x = 8.4\,\mathrm{km}$, $y = 9.2\,\mathrm{km}$, respectively. The simulation result is presented in Figure 3.10 and shows that the budget assigned to Target 1 is increasing with growing target distance from the radar while the budget assigned to the other targets is decreasing. This behavior is expected but does not represent what is typically desired or expected for a radar application.

## 3.9. Analysis of Performance

In the following subsections, a closer look is taken at the general performance of the AODB algorithm w.r.t. other resource allocation methods.

The assumed scenario is the same as in Section 3.8, so the radar and target parameters are identical to Tables 3.1, 3.5 and 3.6. For the following simulations, one implementation of the AODB algorithm and three other strategies using radar System A are considered. The cost evaluation is done for two cases, firstly for $P_D = 1$ and secondly for $P_D < 1$ based on the SNR including a beam positioning error as presented in Section 3.8.3.

It is generally difficult to judge the performance of RRM algorithms in theory because it depends on the specific situation and the specific mission where they are applied. Depending on the user of the radar system, there might be different views on the different parameters. It is possible to show that an approach optimizes the resource distribution according to the chosen cost function. However, if the cost function is not well designed, the tracking, detection, or classification performance can still be unsatisfying. Therefore, the focus is on the expected cost in this section.

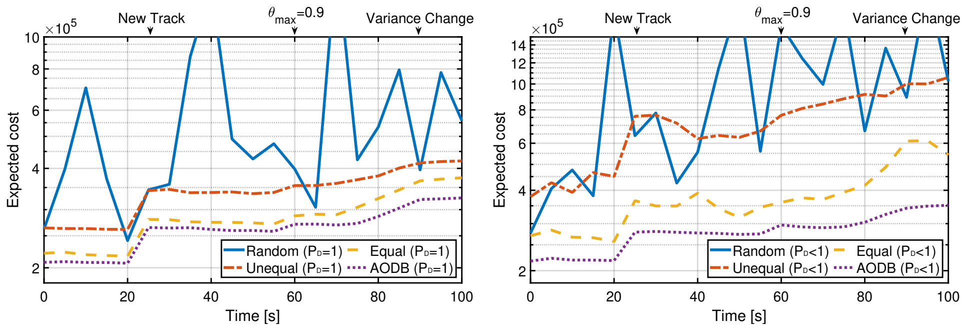
The techniques that are compared to each other are the following:

- **Random policy:** For a given revisit interval $T = 1.2\,\mathrm{s}$, randomly divide the available resources among all tasks.
- **Equal policy:** For a given revisit interval $T = 1.2\,\mathrm{s}$, the available budget is always distributed equally to all tasks.
- **Unequal policy:** For a given revisit interval $T = 1.2\,\mathrm{s}$, target 1 gets more resources assigned than the other targets. The remaining resources are distributed equally over targets 2 to 5.
- **AODB15:** AODB algorithm, assigning resources using PR ($\mathcal{H}$=15, $M$=5).

Figures 3.11a and 3.11b show how the expected cost develops over time for the different techniques that are mentioned above. For the heuristic methods, the future expected cost during a horizon of $\mathcal{H}$=15 has been evaluated stochastically assuming the chosen action, equivalently to the PR. One can see how the AODB clearly minimizes the cost compared to the other techniques for both $P_D = 1$ and $P_D < 1$.

## 3.10. Conclusions

In this chapter, a novel generic framework and proposed approximately optimal algorithmic solutions for solving RRM problems have been developed and shown the algorithm's applicability to a dynamic multi-target tracking scenario. The proposed framework models the



(a) $P_D = 1$. (b) $P_D < 1$.

Figure 3.11: Comparison of the expected cost for different resource distribution methods assuming radar System A. Note that the cost is plotted in a logarithmic scale.

different sensor tasks as constrained POMDPs and solves them by applying a combination of LR and PR. Contrary to the previous chapter, where primarily LTI scenarios were considered, this chapter focuses on dynamic situations with different parameters. The proposed solution is a step towards a genuinely generic framework.

In a simple radar tracking scenario, the dwell time and the revisit interval were optimized using a cost function based on the predicted position error-covariance computed using the EKF.

It was shown that the AODB algorithm budget allocations are close to the optimal steady-state solution in an LTI setting as presented in Chapter 2. Furthermore, the simulation results show that the AODB algorithm can be applied to different systems. It was pointed out how it adjusted itself to known and unknown situational changes in a dynamic scenario.

The presented cost function leads to a larger budget being given to tracking tasks with higher uncertainty. At first glance, this may seem to be entirely appropriate; however, this means that more budget will be assigned to targets at a more extended range. Thus, a simple error-covariance-based cost function will not always suit practical radar applications.

A detailed analysis of the performance of a practical implementation of the algorithm has also been conducted by comparing the optimized cost to other resource distribution methods. It was found that the AODB always led to the lowest cost values compared to the other considered techniques.

# 4 Investigating the Computational Load of the AODB Algorithm

*The results from the previous chapter already implied that the solution of a POMDP using PR can be computationally complex. Chapter 4, therefore, explores the computational load of the AODB algorithm in detail and makes recommendations for the optimal choice of the optimization parameters. Additionally, an alternative implementation of the AODB is introduced that replaces the PR by MPC. Through multiple simulation scenarios, it is shown that this approximation leads to lower computation times while maintaining similar results as the previously discussed PR implementation.*

Parts of this chapter have been published in:

M. I. Schöpe, H. Driessen, and A. Yarovoy, "A Constrained POMDP Formulation and Algorithmic Solution for Radar Resource Management in Multi-Target Tracking", *ISIF Journal of Advances in Information Fusion (JAIF)*, vol. 16, no. 1, pp. 31-47, Jun. 2021.

T. de Boer, M. I. Schöpe, and H. Driessen, "Radar Resource Management for Multi-Target Tracking Using Model Predictive Control", in *Proceedings of the 24th International Conference on Information Fusion (FUSION)*, Sun City, North West, South Africa, 2021, accepted for publication.

The ideas for this chapter and the general solution approaches originated from Max Ian Schöpe. The detailed implementation of the proposed MPC approach has been carried out by the MSc student Thies de Boer under the close daily supervision of Max Ian Schöpe. The student wrote the publication draft with a significant amount of suggestions and corrections by Max Ian Schöpe. Hans Driessen provided additional input during the whole process.

## 4.1. Computational Efficiency of POMDP Solution Methods

As mentioned in the previous chapters, the exact solution of a POMDP becomes increasingly intractable with growing state and action space. In fact, exact solutions are already intractable with a relatively small amount of discrete belief state [72]. Especially in radar applications, the state space of a target is often considered to be continuous, as it can move anywhere in the scene and have a wide range of velocities. This practically leads to an infinite number of possible states. Therefore, real-time radar approaches inevitably require approximate POMDP solution methods to be applied. An overview of these kinds of techniques can be found in [65, 66]. In this thesis and other recent publications, PR is applied in order to solve the POMDP [53, 73, 74]. The PR can theoretically deal with very large state spaces, but for accurate results, it requires more rollouts for averaging with an increasing amount of states. Additionally, the PR needs to explore all available actions, which leads to an even higher computational cost. Thus, applying PR is often a computationally expensive procedure. Although the PR is already an approximate POMDP solution, there exist approximations to the PR as well. In fact, it has been shown that Model Predictive Control (MPC) is a special case of PR with a much lower computational load [75].

In Section 4.2, computational load of the AODB algorithm as introduced in Chapter 3 is analyzed with respect to multiple input parameters. Furthermore, an implementation of the AODB that applies MPC instead of PR to find the best sensing action is introduced as a possible computationally more efficient alternative in Section 4.3. Finally, Section 4.4 contains the conclusions for this chapter.

## 4.2. Analysis of Computational Load

In this section, the computational load of the AODB algorithm is investigated. It should be noted that the current version of the algorithm has not been derived with high efficiency in mind. The following results should be seen as indications since the process can still be optimized.

The computational load of the algorithm has been investigated w.r.t. the following parameters:

- Amount of tracking tasks.
- Step size of LR.
- Desired precision of results.
- Initial value of the Lagrange multiplier.
- Rollout length.

In the following, simulation results are shown based on a single budget calculation. This means that way the LR converges to its final result based on the above parameters is investigated. To generate the figures, the results of 10 simulations have been averaged. The general simulation parameters are shown in Table 4.1. Those parameters are valid for all following simulations, except for the currently evaluated parameter. For that one, a sweep over different values is applied, which is specified in the corresponding subsection. A fixed action space is assumed that is the same for each calculation in the parameter sweep. The chosen system for these simulations produces measurements in range and angle (system A), and the target parameters are the same as in Chapter 3.8, see Table 3.6. The initial Lagrange multiplier value is set to 1. In addition to that, the cost function as introduced in (3.23) is used for all following simulations. In the following figures, normalized times and normalized LR iterations numbers are shown. This means that each data graph is normalized w.r.t. its maximum value. This is done to emphasize that the capability of the hardware and the choice of the general input parameters are not relevant for the discussion of the results.

Table 4.1: General simulation parameters for computational load analysis.

| **Parameter** | **Value** |
|---|---|
| Precision of LR ($\delta_{LR}$): | 0.01 |
| Action discretization ($\Delta T$, $\Delta\tau$): | 0.0035 s |
| Number of rollouts ($\mathcal{N}_r$): | 2 |
| Rollout horizon ($\mathcal{H}$): | 2 steps |
| Base Policy ($\pi_{base}$) | $\boldsymbol{a}$ |
| Maximum budget ($\Theta_{max}$): | 1 |
| Number of simulations: | 10 |
| Beam positioning error ($\Delta\alpha$): | 0 |
| Prob. of detection ($P_D$): | 1 |

### 4.2.1. Influence of Number of Tasks on AODB

The following simulation shows the influence of an increasing number of tasks on the computational load and execution time of the AODB algorithm. Using the above mentioned parameters, 24 different simulations have been conducted for 2 to 25 tracking tasks. The initial Lagrange multiplier value is 1, and the chosen constant LR step size is 8000. Therefore, it is assumed that there is no prior knowledge about the optimal Lagrange multiplier. The results of this simulation can be seen in Figure 4.1a. It can be seen that amount of iterations, the total time until the LR converges, and the time needed for each LR iteration are increasing approximately linearly for a rising number of tracked targets until the increase slows down for larger amounts of targets of 15 and more.

### 4.2.2. Influence of LR Step Size on AODB

In this subsection, a simulation shows the influence of an increasing LR step size on the computational load and execution time of the AODB algorithm. In total, 5 tracking tasks and 50 step sizes between 250 and 20000 are considered, while the initial Lagrange multiplier value is 1. The results of this simulation can be found in Figure 4.1b. It can be seen that the amount of LR iterations needed and the time until convergence are decreasing exponentially. The average time per LR iteration stays approximately constant.

### 4.2.3. Influence of LR Precision on AODB

The following simulation shows the influence of different LR result precisions on the computational load and execution time of the AODB algorithm. Again, 5 tracking tasks and 50 precision values between 0.001 and 0.2 are considered. The results of this simulation can be found in Figure 4.1c. The initial Lagrange multiplier value is 1, and the chosen constant step size is 8000. It can be seen that the amount of LR iterations and the total LR convergence time are decreasing roughly exponentially. The average time per LR iteration stays approximately constant.

### 4.2.4. Influence of Initial Lagrange Multiplier Value on AODB

This simulation shows the influence of different initial Lagrange multiplier values on the computational load and execution time of the AODB algorithm. The simulation considers 5 tracking tasks and 50 initial Lagrangian multiplier values between 1 and 100000. The chosen constant step size is 8000. The results of this simulation can be found in Figure 4.1d. It can be seen that the amount of LR iterations and the LR convergence time have a clear minimum at about 24000. This is the best starting value because it allows the fastest convergence. The average time per LR iteration stays approximately constant.

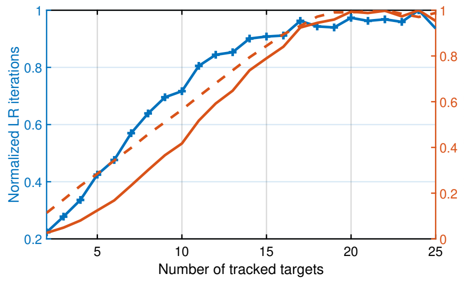
### 4.2.5. Influence of Rollout Horizon Lengths on AODB

The following simulation shows the influence of different PR horizon lengths on the computational load and execution time of the AODB algorithm. In total, 5 tracking tasks are considered, and the rollout length varies from 1 to 25. The initial Lagrange multiplier value is 1, and the chosen constant step size is 8000. The results of this simulation can be found in Figure 4.1e. It can be seen that the amount of LR iterations increases fast in the beginning, before slowly decreasing again for horizon lengths of 6 and longer. The total time needed increases approximately linearly, as well as the time needed per LR iteration.

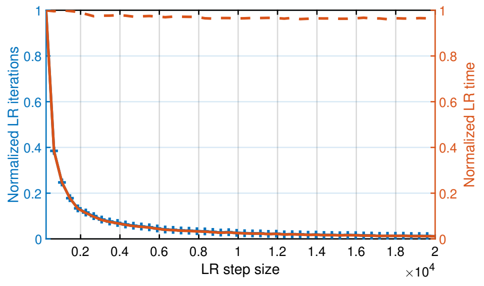
### 4.2.6. Conclusions on Computational Load

Based on the simulation result of the previous subsections, some conclusions can be made regarding the choice of the input parameters. They will be summarized in the following paragraphs.

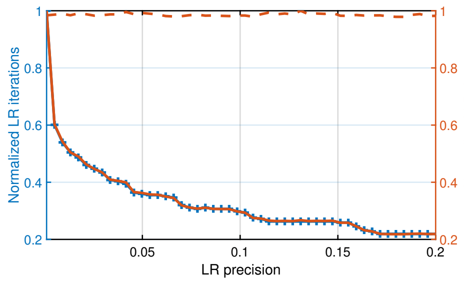
**Number of targets and initial Lagrange multiplier value:** Both the necessary number of LR iterations, as well as the total LR convergence time increases with an increasing number of tracking tasks. Unfortunately, it is generally not possible to influence the number of tasks at will. However, the effect of increasing convergence time can be reduced by choosing the appropriate initial Lagrange multiplier value. It was found that there is a distinct minimum in the number of LR iterations before convergence (see Figure 4.1d). The minimum convergence time, equivalent to a single LR iteration, is attained when that Lagrange multiplier value is chosen as the initial value. It is interesting to see that initial Lagrange multiplier values bigger than the optimal value lead to longer computations than values smaller than the optimum. If some prior knowledge about the Lagrange multiplier value is available (e.g. from the previous budget calculation), this can tremendously decrease the convergence time if the situation has not changed too much since. Boyd et al. have labeled this approach a ”Warm Start” [76].



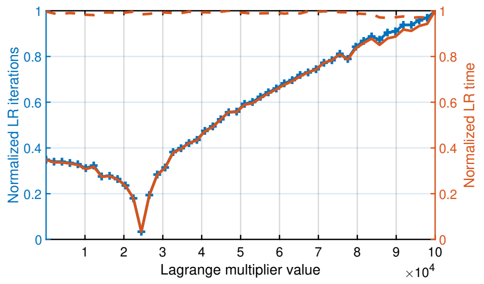
(a) Convergence for different numbers of tracking tasks.



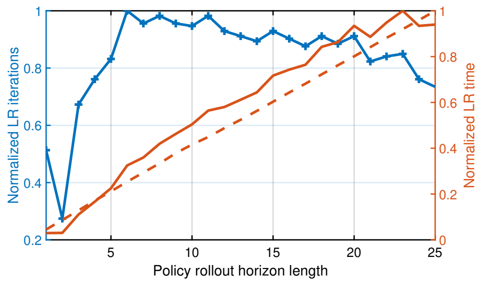
(b) Convergence for different LR step sizes.



(c) Convergence for different LR result precisions.



(d) Convergence for different initial Lagrange multiplier values.



(e) Convergence for different rollout horizon lengths.

Figure 4.1: Analysis of the computational load based on different parameters. Number of LR iterations (blue with crosses) and total time (red) needed for convergence, as well as average time per LR iteration (dashed red).

**LR step size and precision of LR result:** Increasing LR step size and decreasing precision both lead to a decreasing number of LR iterations and time until convergence, while the time needed for one LR iteration stays more or less constant. Generally, it is useful to choose a rather big LR step size. However, if it is chosen too big w.r.t. the precision and the action space discretization, the algorithm might not converge but oscillate around the minimum. If the desired results lie in a local minimum instead of the global one, the algorithm might miss that minimum entirely if the step size is chosen too big. Therefore, choosing a constant step size is probably not the best solution, and adaptive step sizes could increase the performance. There is more freedom to choose the precision, but one should keep in mind that a lower precision will lead to a less accurate result, leading to not precisely meeting the maximum budget constraint.

**PR horizon length:** Increasing the horizon length leads to an almost linear increase of the time per LR iteration. Very short horizons seem to lead to very low numbers of LR iterations until convergence. For horizon lengths longer than 2, the number of LR iterations increases very quickly, although it slightly decreases again for horizons longer than 6. The total LR convergence time increases with growing rollout length (see Figure 4.1e). It is, therefore, reasonable to choose the shortest horizon necessary. It needs to be kept in mind that this is a trade-off with an impact on the track performance, so a longer horizon can potentially improve the mission performance further.

**Size of the action space:** The PR is exploring all possible actions in the action space. Therefore, reducing the evaluated action will also significantly decrease the computation time. One possible way to approach this is to narrow down a part of the action space where the minimum is expected. Then only this part would have to be explored by the PR. A possible implementation could be an adaptive action space per task that only explores the area close to the solution of the last LR iteration.

## 4.3. An Alternative AODB Algorithm

Based on the results from the last chapters, it was decided to investigate alternative POMDP solution methods. Generally, there are four different classes of approximate POMDP solution methods:

- Policy Function Approximation
- Cost Function Approximation
- Value Function Approximation
- Direct Lookahead approximation

The previously applied PR method falls into the category of Direct Lookahead Approximation. For the given problem, Policy Function, Cost Function, and Value Function approximations are very difficult to achieve. Therefore, it was decided to implement MPC as another Direct Lookahead Approximation solution method. More details on the analysis of the different POMDP solution methods can be found in [77]. The expectation is that the computational effort will decrease while keeping a good result quality. Additionally, MPC can optimize the actions at multiple future moments over the evaluated horizon because of the reduced computational effort. This is an advantage over PR, which only optimizes the first action. The formulation of the solution and the presented results are based on the master's thesis result of Thies de Boer [77] and a submitted paper to the Fusion conference [78].

## 4.3.1. Model Predictive Control

The same POMDP formulation of the tracking tasks as in Chapter 3 is assumed. In the previous solution, the POMDP was solved using a PR algorithm that allocates the resources by sampling the possible future in a Monte Carlo fashion. In order to receive accurate results with big state spaces, many of those rollouts have to be performed and averaged. As mentioned before, this can lead to a high computational load.

It has been shown that MPC is a special case of the PR and replaces the randomly sampled future by a modeled approach [75]. It is therefore assumed to lead to a significant decrease in computational load.

Similar to the PR, MPC is a receding horizon approach. This means the prediction horizon is shifted forward after every iteration. The basic operating principle can be summarized as follows, where $\mathcal{H}$ indicates the prediction horizon, which is here always assumed to be equal to the control horizon.

Beginning at time step $k$, the procedure is as follows:

1. Assuming the predefined model, minimize cost over prediction horizon to get the optimal action sequence of length $\mathcal{H}$ over the prediction horizon:

$$\underset{\boldsymbol{a}_t}{\text{minimize}} \sum_{t=k}^{k+\mathcal{H}} c(\boldsymbol{a}_t, \boldsymbol{s}_t)$$

2. Execute only the action corresponding to the next time-step.

3. Go to time step $k+1$ and start again at step 1.

Essentially, those steps are equivalent to the PR with two exceptions. Firstly, as it is assumed that the tracked target strictly follows the underlying motion model, only one sampling of the future is necessary, as there are no other possible futures. Secondly, because of this simplification, the MPC approach can optimize the actions in every time step of the horizon. In contrast, PR applies a predefined BP for each horizon step, except the first one.

If a small number of rollouts per action is chosen or if the choice of the BP is bad, MPC could potentially lead to better results than PR. Additionally, MPC has the advantage that it can handle larger action spaces due to the reduced computational effort needed.

## 4.3.2. Optimization Problem and Simulation Scenario

The optimization problem is exactly the same as introduced in (3.22). For reasons of simplification, the revisit interval $T$ is set to a constant value in the following simulations. The chosen radar system is equivalent to System A in Table 3.1 and the same EKF as in Chapter 3 is applied in these simulations. Also, the motion and measurement models are the same.

Since the revisit interval $T$ is set to a constant value, the penalty term mentioned in (3.23) can be removed and the cost function can then be written as

$$c(\tau_k^n, \boldsymbol{s}_k^n \boldsymbol{P}_{k_n+1|k_n}) = \text{trace}\left(\boldsymbol{E}\boldsymbol{P}_{k|k-1}(\tau_k^n, \boldsymbol{s}_k^n \boldsymbol{P}_{k_n+1|k_n})\boldsymbol{E}^T\right), \tag{4.1}$$

where

$$E = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}. \tag{4.2}$$

As mentioned in the previous chapters (e.g., Section A), LR is also applied in this implementation of the AODB to include the constraint into the cost function and solve the problem iteratively. Instead of applying the subgradient method, the so-called Golden Section Search (GSS) is used as explained in Appendix B. The algorithm's performance is evaluated through the following simulation scenarios for both the PR and the MPC implementation.

### 4.3.3. Simulation Scenario A

Simulation scenario A consists of 5 targets. The corresponding trajectories are shown in Figure 4.2 and the target parameters can be found in Table 4.3. All noise is assumed to be Gaussian with zero-mean. In this example, some events are taking place during the simulation time which have an impact on the resulting budget allocations:

- 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 target 1 increases.

For the simulation the parameters listed in Table 4.2 were used.

Table 4.2: Simulation parameters scenario A.

| **Reference parameter** | **Value** |
|---|---|
| Maneuverability noise variance ($\sigma_w^2$) | 2.5 m$^2$ s$^{-4}$ |
| Radar Cross Section ($\varsigma$) | 10 m$^2$ |
| Prediction horizon length ($\mathcal{H}$) | 15 |
| Number of rollouts ($\mathcal{N}_r$) | 10 |
| Simulation update interval | 5 s |
| Budget precision ($\delta_{LR}$) | 0.002 |
| Revisit interval ($T$) | 1 s |

Figures 4.3a and 4.3b show the budget distributions for the example scenario from Figure 4.2 using the MPC and the PR algorithm, respectively. It can be seen that targets that are located further away from the radar are allocated a larger amount of the budget than those closer to the radar. This is due to the properties of the chosen cost function and has already been discussed in Section 3.8.4. Furthermore, the figure reflects the changes to the system at the different time steps. At $t = 20$ s, a new object needs to be tracked, and some budget

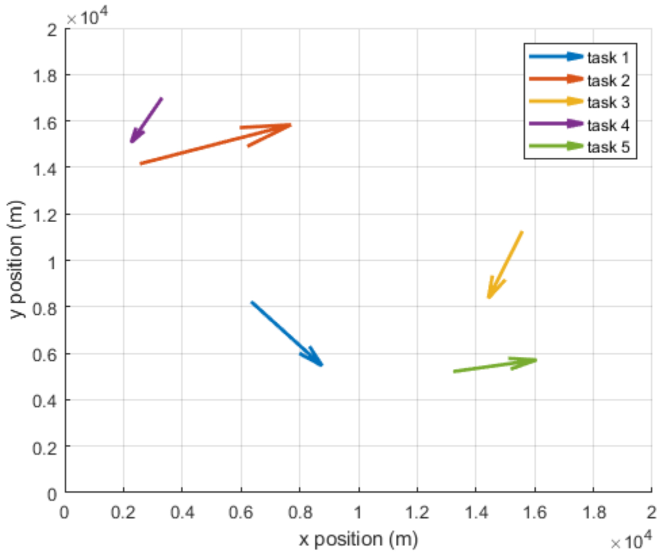
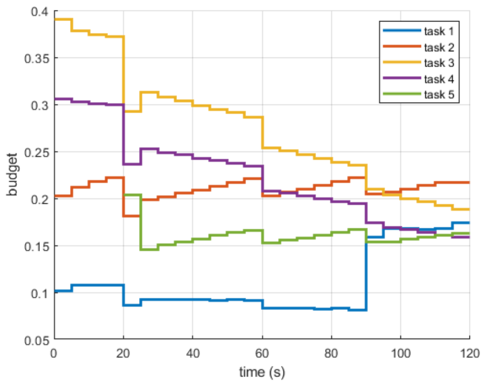


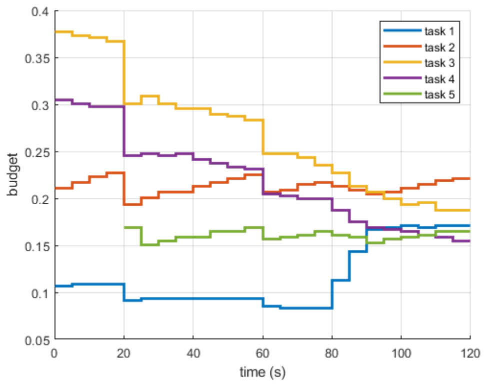
Figure 4.2: Trajectories of the 5 objects to be tracked for a scenario of 100 s.

Table 4.3: Initial target parameters for simulation Scenario A

| **Target** $n$ | $x_0^n$ [km] | $y_0^n$ [km] | $\dot{x}_0^n$ [m s$^{-1}$] | $\dot{y}_0^n$ [m s$^{-1}$] | $\sigma_w^2$ [m$^2$ s$^{-4}$] | $\varsigma_n$ [m$^2$] |
|---|---|---|---|---|---|---|
| 1 | 6.3 | 8.2 | 23 | -23 | 2.5 | 10 |
| 2 | 2.6 | 14.2 | 49 | 14 | 2.5 | 10 |
| 3 | 15.6 | 11.3 | -11 | -24 | 2.5 | 10 |
| 4 | 3.3 | 17 | -10 | -16 | 2.5 | 10 |
| 5 | 13.2 | 5.2 | 27 | 5 | 2.5 | 10 |



(a) Optimization using MPC.



(b) Otimization using PR.

Figure 4.3: Budget allocations for Scenario A.

is made available for this task. At $t = 60$ s, the total available budget decreases, which is expressed by a decreased budget for each task. At $t = 90$ s, the maneuverability noise variance of the first target increases, resulting in the need to take more accurate measurements of this target and therefore increasing the budget of task 1. When comparing the budget distributions from the PR 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. In the case of MPC, the effect of this change shows only after $t = 90$ s, while in the case of PR, the effect of this maneuverability change can already be observed at $t = 80$ s. This is because, in the PR optimization, a predefined BP is executed after the first horizon step, while the MPC is optimizing the actions for all horizon steps. As the horizon is 15 steps or 15 s long, this change in maneuverability already needs to be taken into account at $t = 80$ s. Therefore, the actions are already slightly influenced by this change.

### Execution Time Comparison

The main goal of applying MPC was to lower the computational load compared to the PR algorithm. This is evaluated by comparing the execution time of an average budget update operation using both algorithms. For this comparison, scenarios with linearly moving and non-maneuvering targets similar to Scenario A are used with varying numbers of moving objects. Figure 4.4 shows the results from averaging the execution times over three simulation runs. It can be seen that the execution times are significantly lower when using MPC compared to when PR is used.

### Realized Cost Comparison

For MPC to be a good replacement for the PR, the resulting budget allocations should be similar. To investigate this, the realized costs were compared using the same scenarios as the execution time comparison. The realized cost is defined as the sum of the evaluated cost function at every time step during the simulation. Figure 4.5 shows the average realized costs for three simulation runs. For comparison, the realized cost of using an equal distribution, i.e., allocating the same budget to all targets at every time step, is also included in the

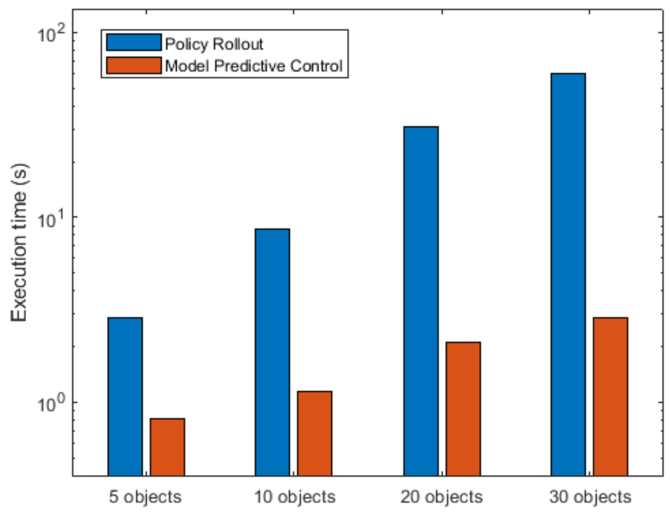


Figure 4.4: Comparison of execution times of PR and MPC algorithms.

figure. It can be seen that MPC and PR have similar performances when looking at the realized costs. For 20 objects, MPC even seems to perform better than PR due to the variance in the PR results. For more accurate results, more rollouts per action must be performed in the PR, and additionally, the result must be averaged over more simulation runs.

### 4.3.4. Simulation Scenario B

Simulation Scenario B consists of a static and a moving target. The first target moves straight at the beginning of the simulation before turning towards the radar sensor and then continuing a linear trajectory. This trajectory is shown in Figure 4.6. The parameters used for the simulation are summarized in Table 4.4. Figures 4.7a and 4.7b show the resulting budget distributions of this scenario for MPC and PR, respectively. In this case, the budget distribution is very similar. Furthermore, the resulting realized costs over the simulation time are very close to each other.

### 4.3.5. Simulation Scenario C

Simulation Scenario C consists again of two targets. The first target makes an unexpected maneuver at $t = 15\,\text{s}$ which can be seen in Figure 4.8. In the scenario there is an area $15\,\text{km} \geq x \leq 20\,\text{km}$, $15\,\text{km} \geq y \leq 20\,\text{km}$, in which the quality of the measurements are negatively affected due to, e.g. weather conditions. If an object enters this area, at least 80% of the budget is needed for the radar to keep the track. The second target makes the same maneuver but in the negative x and y quadrant, where there is no such grey area. For this simulation, the parameters listed in Table 4.5 were used. The resulting budget distributions for MPC and PR can be seen in Figures 4.9a and 4.9b. It can be seen that the PR implementation adapts earlier to the unexpected maneuver, as in some of the rollouts

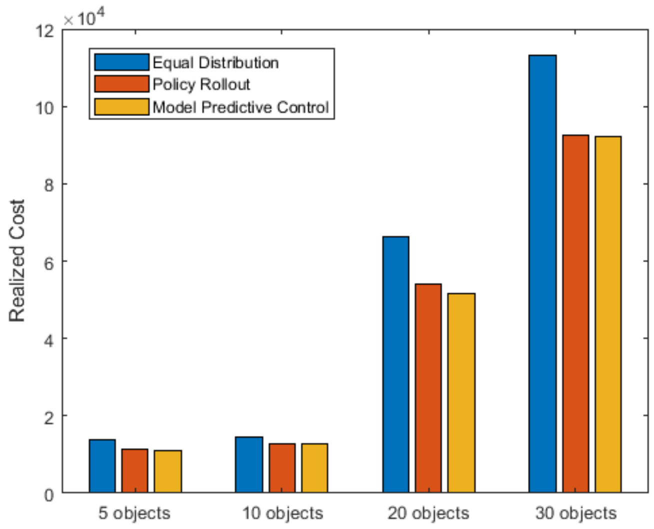


Figure 4.5: Comparison of realized costs of equal distribution, PR and MPC.

Table 4.4: Simulation parameters Scenario B

| **Reference parameter** | **Value** |
|---|---|
| Maneuverability noise variance object 1 ($\sigma_w^2$) | $2.5\,\mathrm{m}^2\,\mathrm{s}^{-4}$ |
| Maneuverability noise variance object 2 ($\sigma_w^2$) | $0.1\,\mathrm{m}^2\,\mathrm{s}^{-4}$ |
| Radar Cross Section ($\varsigma$) | $10\,\mathrm{m}^2$ |
| Prediction horizon length ($\mathcal{H}$) | 5 |
| Number of rollouts | 10 |
| Simulation update interval | 5 s |
| Budget precision ($\delta_{LR}$) | 0.002 |

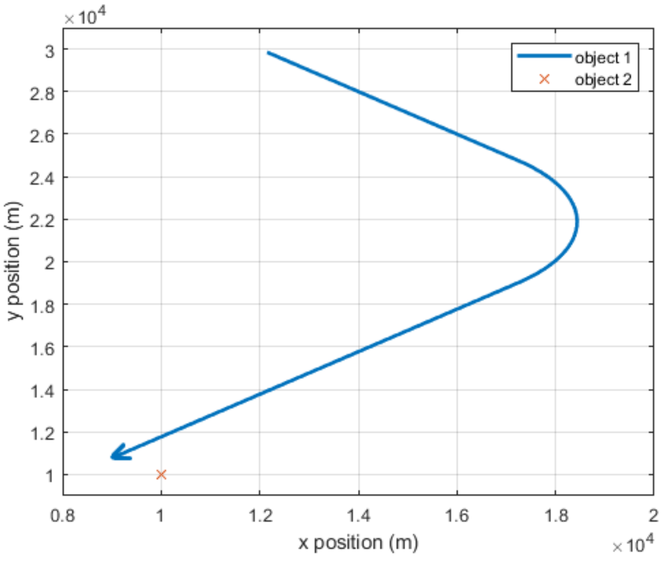
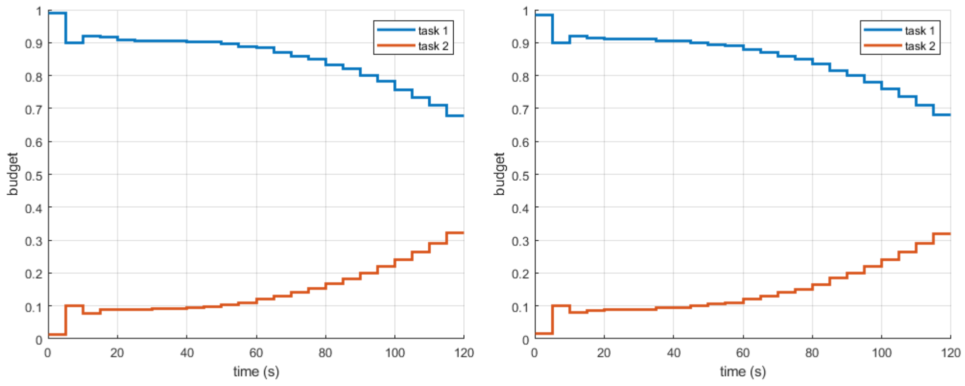


Figure 4.6: Trajectories of the targets in Scenario B.



(a) Optimization using MPC.

(b) Optimization using PR.

Figure 4.7: Budget allocations for Scenario B.

the object maneuvers into the grey area between $t = 15\,\mathrm{s}$ and $t = 20\,\mathrm{s}$, while the MPC approach at that point still assumes that the target continues moving straight into the same direction, avoiding the grey area. As a result, when applying the MPC implementation, the track would be lost during this scenario. This downside of MPC only assuming a perfect system model can be treated by using a more robust MPC scheme, similar to [79], where different disturbances in the state are considered. However, this leads to a computationally more complex control law.

Table 4.5: Simulation parameters for Scenario C

| **Reference parameter** | **Value** |
|---|---|
| Maneuverability noise variance object 1 ($\sigma_w^2$) | $2.5\,\mathrm{m}^2\,\mathrm{s}^{-4}$ |
| Maneuverability noise variance object 2 ($\sigma_w^2$) | $2.5\,\mathrm{m}^2\,\mathrm{s}^{-4}$ |
| Radar Cross Section ($\varsigma$) | $10\,\mathrm{m}^2$ |
| Prediction horizon length ($\mathcal{H}$) | 5 |
| Base Policy ($\pi_{base}$) | $\boldsymbol{a}$ |
| Number of rollouts | 10 |
| Simulation update interval | 5 s |
| Budget precision ($\delta_{LR}$) | 0.002 |

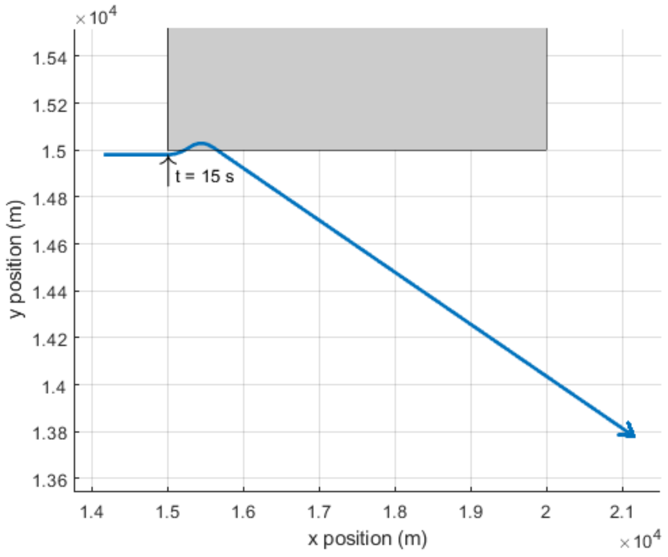
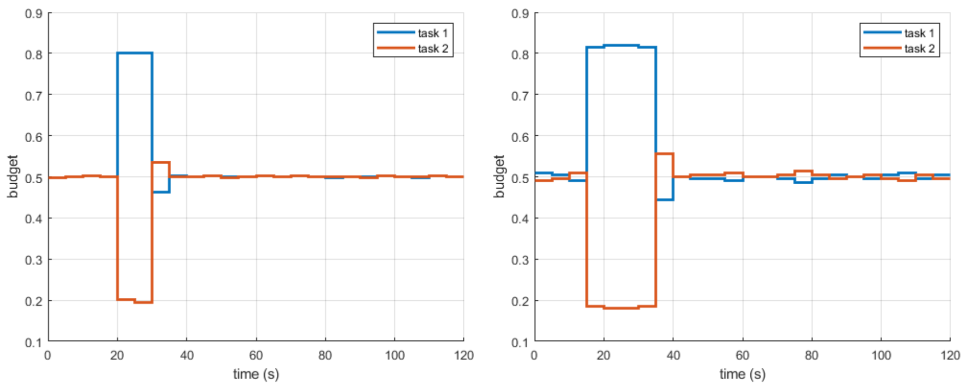


Figure 4.8: Trajectory of target 1 in Scenario C.



(a) Optimization using MPC.

(b) Optimization using PR.

Figure 4.9: Budget allocations for Scenario C.

## 4.4. Conclusions

In this chapter, the computational load of the AODB algorithm was investigated. Based on those results, suggestions about a good choice of input parameters have been presented. These practical results are very valuable as the literature on RRM usually stays on a very high level without presenting practical implementation details. It was shown that the computational load of the PR can become very high, and alternative implementation could be considered for increasing the computational efficiency.

Based on the previous experiences, it has been chosen to investigate the impact of replacing the PR with MPC. The fact that this is easily possible emphasizes the generality and flexibility of the framework. The applicability of the MPC method has been shown in a simple dynamic tracking scenario with linearly moving targets, as well as in a scenario with a maneuvering target. The budget allocations and cost results of both the PR implementation and the MPC implementation are very similar, highlighting that both solutions are valid in the chosen simulation scenarios.

For the same type of scenario, it has been shown that the new MPC approach clearly outperforms the previously considered PR method w.r.t. the computation time. Additionally, the reduction in computational effort enables the MPC approach to optimize actions for multiple time steps into the future, while PR only optimizes the subsequent action.

Nevertheless, it has also been shown that certain scenarios exist where the PR leads to better tracking results than the MPC. This emphasizes that the MPC is an approximation of the PR.

For future research, it might be interesting to investigate the possibility of combining aspects of both PR and MPC, such as applying PR for a continuous action space.

# 5 Radar Resource Management for Multi-Target Joint Tracking and Classification

*The previous chapters introduced a framework to RRM which can flexibly be used in many different use cases by adjusting the chosen cost function according to the needs and wishes of the user. It was shown that the framework is applicable to tracking scenarios and delivers very promising results. Additionally, a performance analysis discussed the problems and limitations w.r.t. the implementation of the proposed algorithm. This chapter focuses on the generality of the approach and shows how the presented framework can be applied to include different types of operations. It is also shown that tracking and classification can be treated jointly through a single task type with a proper model and a reasonable cost function definition. The resources for these tasks are optimized jointly by considering a single cost function.*

Parts of this chapter have been published in:

M. I. Schöpe, H. Driessen, and A. Yarovoy, "Optimal Threat-based Radar Resource Management for Multi-Target Joint Tracking and Classification", *ISIF Journal of Advances in Information Fusion (JAIF)*, 2021, under review.

## 5.1. Radar Resource Management for Classification

For a successful radar application, it is often necessary to make a distinction between different types of targets. Therefore, classification is a vital task for every modern radar system and needs to be considered in RRM. A general high-level overview of classification techniques in CR and RRM is shown by Brüggenwirth et al. in [10]. Furthermore, Kreucher and Hero presented a very generic framework that is potentially capable of doing joint detection, tracking and classification [26]. The explanation of the approach stays on a very high level and is only demonstrated through a detection and tracking scenario.

Most RRM approaches for classification are myopic and focus on a simple waveform or sensor mode selection, often for a single object. In [80] Sowelam and Tewfik present such an approach where the Kullback-Leibler information is maximized for the subsequent measurement. Based on this, the algorithm can decide if another measurement is necessary and which waveform needs to be chosen from a predefined library. Another example has been shown by Bell et al. and considers both tracking and classification [81]. The system is assumed to have separate tracking and classification modes which each have a predefined waveform library to choose from. The proposed algorithm makes a decision about the following sensor action to be executed. As objective functions, both task-driven and information-driven possibilities are discussed. While the task-driven approach requires different objective functions for the two sensing modes, the information-driven approach can compare the two different task types through the information gain.

A popular approach is to introduce a measure of risk or threat. The idea is to summarize the interesting task quantities into a single scalar number that is easy to compare. In [24], Martin introduces a risk-based approach where the risk depends on the probability of making a wrong classification and the possibility of track loss multiplied with predefined cost values. The approach finds a solution for both tracking and classification in a myopic fashion. The measurements are always taken in the same way, but the algorithm decides which target will be sensed. In [25] and [82], similar approaches are presented. From the perspective of this thesis, such a cost function definition is not preferred, as predefined cost values cannot easily represent the risk in all possible situations. This leads to a lack of flexibility in the approach. Bolderhij et al. present an approach for military radar applications that relies on a large amount of expert knowledge to decide the risk level [83]. Although many different situations are considered, this approach does not automatically balance the resources and can not flexibly adapt to different situations. A more interesting approach is shown by Katsilieris et al. in [47] where joint tracking and classification is performed by running a tracking filter per target class in parallel. The classification is done by comparing the likelihood of a measurement to belong to the different tracks. The next sensing action is then chosen by evaluating the threat's uncertainty based on the target state.

Many RRM approaches for classification assume an MDP or POMDP framework. This is relatively easy to implement for simple classification problems, as the number of considered states is usually quite low. An advantage of using such a framework is the possibility of taking the expected future into account. Chong et al. present a very simple general example of how to use classification in RRM with POMDPs [66]. Castãnón applies a non-myopic POMDP approach for a classification scenario of almost 10000 objects where the algorithm chooses from a set of sensor modes [37]. This approach does not take the position and velocity of the objects into account. Another interesting approach has been presented by La

Scala et al. in [84] and non-myopically solves the underlying POMDP in a detection and classification scenario. The algorithm selects the best waveforms from a predefined library. The authors promise that an extension to tracking can be achieved without much effort but do not demonstrate that explicitly. Two other classification approaches that assume an MDP or POMDP framework are shown in [25, 85]. Since the amount of states is relatively low in most classification problems, also ML techniques have been suggested for solving the underlying (PO)MDPs, e.g., by Langford and Zadrozny in [86] and Blatt and Hero in [87].

Most approaches that focus on RRM for classification are very concrete and apply heuristic rules to compare different task types. In most of the proposed approaches, the different tasks are assumed to be independent or very weakly dependent. This chapter will specifically focus on cases where the tasks are joint. In addition to that, most approaches are myopic and do not take into account MDP or POMDP frameworks. The introduction of risk or threat measures is widespread and seems promising as it simplifies the comparison of tasks. This chapter shows that the generic algorithm introduced in the previous chapters can be used to address the shortcomings of previously published literature. It is explained how the approach can easily be adjusted to include joint tracking and classification, using a single cost function for both task types. The underlying POMDP is solved non-myopically and the resulting policy is achieved by balancing all the considered actions in the action space.

The remainder of this chapter is structured as follows. Section 5.2 defines the general RRM problem and Section 5.3 introduces the proposed joint tracking and classification approach. Furthermore, Section 5.4 introduces the applied threat and cost function, while Section 5.5 gives details about the radar scenario used for the simulations as discussed in Sections 5.6 to 5.8. Finally, Section 5.9 contains the conclusions.

## 5.2. General Problem Definition and Radar Scenario

The general formulation of the problem is the same as shown in Chapter 2. The algorithm is essentially the same as presented in Chapter 3, but each target now has a class parameter assigned, which influences the movements of the targets. In the simulations, this is taken into account in the target tracking and optimization. In addition to that, a scalar number called “threat” is introduced that contains the uncertainty of both the track and the classification of a target.

If not stated otherwise, the assumptions from Chapter 3 are still valid here. This includes the assumed radar systems, the velocity and measurement models, and the SNR model. Only zero-mean Gaussian noise is considered in the following. Therefore, all mentioned PDFs are Gaussian.

## 5.3. Joint Tracking and Classification

This chapter assumes that each target is of a specific predefined class that is initially unknown to the radar system. To make the classification decision, a Bayesian classifier will be applied. Suppose a class feature could be measured directly, and the features were independent of each other. In that case, the classification problem can be solved, e.g., by applying a naive Bayes classifier using these class measurements directly.

If the class features cannot be observed directly, the behavior of the target often contains information about the underlying target type. In that case, joint tracking and classification can be applied. Similar approaches have been presented, e.g., in [88, 89]. Based on the measurements taken by the radar sensor, a track can be created with the help of a tracking filter (e.g., EKF as applied in the previous chapter). The track then describes the movements of the observed objects. The problem that needs to be solved contains both discrete (class) and continuous variables (e.g., position, velocity) of the targets are considered. The following equations are based on Bayesian theory (see, for instance, [90–92]).

Taking into account the class of the target, the state evolution equation in (2.3) changes to

$$\boldsymbol{s}_{t+\Delta t}^{n} = f_{\Delta t}\left(\boldsymbol{s}_{t}^{n}, c^{n}, \boldsymbol{w}_{t}^{n}\right), \tag{5.1}$$

where $c^n \in \boldsymbol{C}$ is a scalar and denotes the class of target $n$ which is not changing over time. The measurement function is defined similarly to (6.1):

$$\boldsymbol{z}_{t}^{n} = \mathfrak{h}\left(\boldsymbol{s}_{t}^{n}, \boldsymbol{v}_{t}^{n}, \boldsymbol{a}_{t}^{n}, f_{c}^{n}\right), \tag{5.2}$$

where $f^n$ is a directly measurable feature of target $n$, represented by a scalar value. The PDF for feature $f_c^n$ can explicitly be written as

$$p\left(z_{t,f}^{n} \,|\, \mathrm{SNR}, c^{n}\right) = \frac{1}{\sqrt{2\pi\sigma_f^2(\mathrm{SNR})}} \exp\left(-\frac{1}{2}\frac{(z_{t,f}^{n} - f_c^{n})^2}{\sigma_f^2(\mathrm{SNR})}\right), \tag{5.3}$$

where $z_{t,f}^n$ is the measurement of the feature of target $n$ at time $t$, and $\sigma_f^2(\mathrm{SNR})$ is the feature measurement variance which depends on the SNR. One could think for instance of the RCS or the micro-Doppler spectrum. This feature is assumed to be directly connected to the class of the object and not dependent on the state of the target. Therefore, the measurement $\boldsymbol{z}_t^n$ consists of a state (position and Doppler) and a class (feature) component. The PDFs of state, process or maneuverability noise and measurement noise can then depend on the underlying target class:

$$\begin{aligned} &p\left(\boldsymbol{s}_{t+\Delta t}^{n} | c^{n}\right), \\ &p\left(\boldsymbol{w}_{t}^{n} | c^{n}\right), \\ &p\left(\boldsymbol{v}_{t}^{n} | c^{n}\right) \\ &p\left(f_{c}^{n} | c^{n}\right). \end{aligned} \tag{5.4}$$

The goal of this joint tracking and classification approach is to recursively calculate the posterior joint PDF

$$p\left(\boldsymbol{s}_{t}^{n}, c^{n} | \boldsymbol{Z}_{t}^{n}\right) = p\left(\boldsymbol{s}_{t}^{n} | c^{n}, \boldsymbol{Z}_{t}^{n}\right) P(c^{n} | \boldsymbol{Z}_{t}^{n}), \tag{5.5}$$

where $\boldsymbol{Z}_t^n = [\boldsymbol{z}_t^n, \boldsymbol{z}_{t-\Delta t}^n, \boldsymbol{z}_{t-2\Delta t}^n, \dots, \boldsymbol{z}_0^n]$ are all measurements taken for target $n$ until time $t$ and $P(c^n|\boldsymbol{Z}_t^n)$ are the prior class probabilities which are known from the last iteration. Using the the Bayesian evolution and update equations, the conditional posterior density can be written as

$$p\left(\boldsymbol{s}_{t+\Delta t}^{n} | c^{n}, \boldsymbol{Z}_{t}^{n}\right) = \int_{S} p\left(\boldsymbol{s}_{t+\Delta t}^{n} | \boldsymbol{s}_{t}^{n}, c^{n}\right) p\left(\boldsymbol{s}_{t}^{n} | c^{n}, \boldsymbol{Z}_{t}^{n}\right) d\boldsymbol{s}_{t}^{n}, \tag{5.6}$$

where

$$p\left(\boldsymbol{s}_t^n|c^n,\boldsymbol{Z}_t^n\right)=\frac{p\left(\boldsymbol{z}_t^n|\boldsymbol{s}_t^n,c^n\right)p\left(\boldsymbol{s}_t^n|c^n,\boldsymbol{Z}_{t-\Delta t}^n\right)}{p\left(\boldsymbol{z}_t^n|c^n,\boldsymbol{Z}_{t-\Delta t}^n\right)}. \tag{5.7}$$

The normalizing constant in the denominator is calculated with

$$p\left(\boldsymbol{z}_t^n|c^n,\boldsymbol{Z}_{t-\Delta t}^n\right)=\int_S p\left(\boldsymbol{z}_t^n|\boldsymbol{s}_t^n,c^n,\boldsymbol{Z}_{t-\Delta t}^n\right)p\left(\boldsymbol{s}_t^n|c^n,\boldsymbol{Z}_t^n\right)d\boldsymbol{s}_t^n. \tag{5.8}$$

As the measurement consists of a state dependent and a state independent part which is based only on the class, this expression can also be written as

$$p\left(\boldsymbol{z}_t^n|c^n,\boldsymbol{Z}_{t-\Delta t}^n\right)=p\left(\boldsymbol{z}_t^{n,s}|c^n,\boldsymbol{Z}_{t-\Delta t}^{n,s}\right)p\left(\boldsymbol{z}_t^{n,c}|c^n\right), \tag{5.9}$$

where $\boldsymbol{z}^{n,s,c}$ denotes the state and class-dependent measurement and $\boldsymbol{z}^{n,c}$ the class-dependent feature $f_c^n$ measurement for target $n$ at time $t$. The posterior class probability is calculated via

$$P\left(c^n|\boldsymbol{Z}_t^n\right)=\frac{p\left(\boldsymbol{z}_t^n|c^n,\boldsymbol{Z}_{t-\Delta t}^n\right)P\left(c^n|\boldsymbol{Z}_{t-\Delta t}^n\right)}{p\left(\boldsymbol{z}_t^n|\boldsymbol{Z}_{t-\Delta t}^n\right)}. \tag{5.10}$$

The likelihood of the current measurement given all the previous measurement is defined as

$$p\left(\boldsymbol{z}_t^n|\boldsymbol{Z}_{t-\Delta t}^n\right)=\sum_{c=1}^{C}p(\boldsymbol{z}_t^n|c^n,\boldsymbol{Z}_{t-\Delta t}^n)P\left(c^n|\boldsymbol{Z}_{t-\Delta t}^n\right), \tag{5.11}$$

where $C$ is the number of assumed target classes. The recursive process that is described through (5.1) to (5.11) requires $C$ different tracking filters, each conditioned to a specific class. Based on the likelihood of the current measurement being associated with one of the tracks, the class probability is updated. The process is summarized in Figure 5.1.

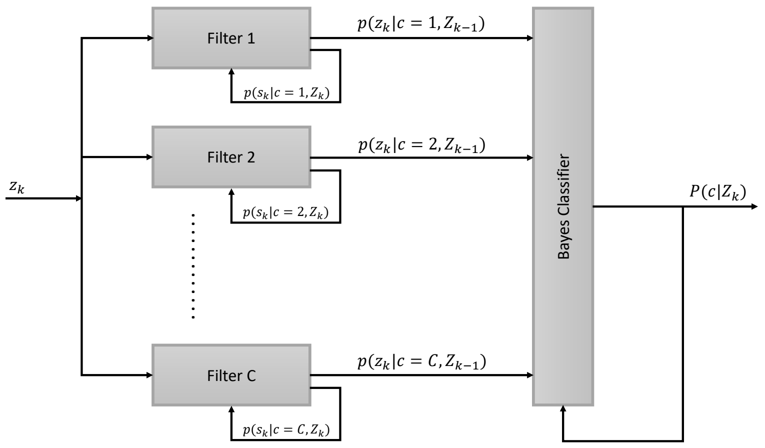


Figure 5.1: Joint tracking and classification process.

## 5.4. Formulation of Cost Function

The assumed cost function in this chapter is based on a definition of threat. This definition depends heavily on the considered scenario and the wishes and expectations of the user. There is practically an unlimited amount of possibilities for constructing such a function. In this section, it is considered that the threat $\phi(c, \boldsymbol{s})$ depends on the class and the state of a target. The cost function will be defined by the variance in the threat knowledge of a target. This means that the cost will be very high for unclassified targets, as all class-dependent threat values are equally likely. Once the knowledge of the target class increases, also this variance in threat will decrease. An explicit example formulation of the threat and the cost function will be introduced later, together with the simulation scenarios. First, the focus is on transforming the PDF from the state domain to the threat domain. As the cost calculation is done for each target separately, the target-related superscript $n$ is dropped to simplify the notations in the following subsections.

### 5.4.1. Unscented Transform

The running target tracks supply a PDF of the target state. Since the transformation of the state PDF to the threat PDF is nonlinear, a sampling approach is chosen. A possible implementation of this is to sample the threat PDF with a certain number of random samples in the state PDF. For an accurate result, many samples are necessary, which can make this approach very slow. Therefore, in this chapter the samples in the state space of the target are chosen with the help of the unscented transform that is also applied in the Unscented KF [93]. For a D-dimensional PDF, $2D + 1$ sigma points are necessary. The procedure for calculating the current threat at a certain moment in time is as follows:

1. Calculate the Cholesky decomposition of the belief state covariance matrix of the target:
$$\boldsymbol{L}\boldsymbol{L}^T = \boldsymbol{P}, \tag{5.12}$$
where $\boldsymbol{P}$ is the belief state covariance matrix of the target.

2. Calculate the so-called sigma points:
$$\begin{aligned} \mathbb{x}^0 &= \hat{\boldsymbol{s}}, \\ \mathbb{x}^i &= \hat{\boldsymbol{s}} + \sqrt{D+\kappa}\,\mathrm{col}_i\boldsymbol{L} \quad i = 1, \dots, D, \\ \mathbb{x}^{i+D} &= \hat{\boldsymbol{s}} - \sqrt{D+\kappa}\,\mathrm{col}_i\boldsymbol{L} \quad i = 1, \dots, D. \end{aligned} \tag{5.13}$$
where $\hat{\boldsymbol{s}}$ is the belief state mean of the target, $\kappa = 3 - D$ and $\mathrm{col}_i\boldsymbol{L}$ denotes the $i$-th column of matrix $\boldsymbol{L}$.

3. Now, each of these samples has to be transformed to the threat domain by using the threat function $\phi(c, \boldsymbol{s})$.
$$\mathbb{y}_c^i = \phi(\hat{c}, \mathbb{x}^i)) \quad i = 0, \dots, 2D, \tag{5.14}$$
where $\hat{c}$ is the believed class of the target.

4. From the samples in the threat domain, the threat PDF is defined by the mean and covariance:

$$\begin{aligned} \hat{\phi}_c &= \sum_{i=0}^{2D} w^i \mathbb{y}_c^i, \\ \Sigma_{\phi,c} &= \sum_{i=0}^{2D} w^i (\mathbb{y}_c^i - \hat{\phi}_c)(\mathbb{y}_c^i - \hat{\phi}_c)^T, \end{aligned} \tag{5.15}$$

where $\hat{\phi}_c$ is the mean and $\Sigma_{\phi,c}$ the covariance of the threat PDF based on class $c$ and $w^i$ are weights for the samples given as

$$w^i = \begin{cases} \frac{\kappa}{D+\kappa}, & \text{if } i = 0 \\ \frac{1}{2(D+\kappa)}, & \text{otherwise} \end{cases}. \tag{5.16}$$

### 5.4.2. Combination of Threat PDFs

Since the threat PDF depends on the class $c$, is has to be calculated for each class separately. Based on the resulting PDFs for $C$ different classes, a total PDF can be constructed. The total mean of the threat $\hat{\phi}_{\text{tot}}$ for the target is defined as

$$\begin{aligned} \hat{\phi}_{\text{tot}} &= \int_\Phi \phi p(\phi|\boldsymbol{z}) d\phi \\ &= \int_\Phi \phi \sum_{c=1}^{C} \mathcal{N}(\phi; \hat{\phi}_c, \Sigma_{\phi,c}) P(c|\boldsymbol{Z}) d\phi \\ &= \sum_{c=1}^{C} P(c|\boldsymbol{Z}) \int_\Phi \phi \mathcal{N}(\phi; \hat{\phi}_c, \Sigma_{\phi,c}) d\phi \\ &= \sum_{c=1}^{C} P(c|\boldsymbol{Z}) \hat{\phi}_c, \end{aligned} \tag{5.17}$$

where $\boldsymbol{z}$ is a recent measurement of the target state, $\mathcal{N}(\phi; \hat{\phi}_c, \Sigma_{\phi,c})$ denotes a normal distribution with mean $\hat{\phi}_c$ and variance $\Sigma_{\phi,c}$ and $P(c, \boldsymbol{Z})$ is the posterior class probability based on all previous measurements $\boldsymbol{Z}$. The variance can be calculated using

$$\begin{aligned} \Sigma_{\phi,\text{tot}} &= \int_\Phi (\phi - \hat{\phi}_{\text{tot}})^2 p(\phi|\boldsymbol{z}) d\phi \\ &= \int_\Phi (\phi^2 - 2\phi\hat{\phi}_{\text{tot}} + \hat{\phi}_c)^2) p(\phi|\boldsymbol{z}) d\phi \\ &= \int_\Phi \phi^2 p(\phi|\boldsymbol{z}) d\phi - \hat{\phi}_{\text{tot}}^2. \end{aligned} \tag{5.18}$$

Using

$$\begin{aligned}\int_\Phi \phi^2 p(\phi|\boldsymbol{z})d\phi &= \int_\Phi \phi^2 \sum_{c=1}^{C} \mathcal{N}(\phi; \hat{\phi}_c, \Sigma_{\phi,c})P(c|\boldsymbol{Z})d\phi \\ &= \sum_{c=1}^{C} P(c|\boldsymbol{Z}) \int_\Phi \phi^2 \mathcal{N}(\phi; \hat{\phi}_c, \Sigma_{\phi,c})d\phi \\ &= \sum_{c=1}^{C} P(c|\boldsymbol{Z})(\Sigma_{\phi,c} + \hat{\phi}_c^2),\end{aligned} \tag{5.19}$$

it can also be written as

$$\Sigma_{\phi,\mathrm{tot}} = \sum_{c=1}^{C} P(c|\boldsymbol{Z})(\Sigma_{\phi,c} + \hat{\phi}_c^2) - \hat{\phi}_{\mathrm{tot}}^2. \tag{5.20}$$

### 5.4.3. Variance of Threat

The previous subsection described a way to transform the PDF in the state and class domain to the threat domain by taking into account multiple possible target classes. Given this threat PDF, different cost functions could be constructed. A simple and unambiguous choice is to simply evaluate the total threat variance $\Sigma_{\phi,\mathrm{tot}}$. The underlying assumption is that the radar system cannot influence the target state but only the uncertainty about the knowledge of the target state by adjusting its sensing actions. Following this cost function, the most resources will be assigned to the targets where the biggest decrease in uncertainty (decrease in threat variance) is expected.

The hypothesis is that this will lead to more resources being assigned to objects of an uncertain class. Once all the objects are classified, the uncertainty in the threat will drop significantly and only depend on the uncertainty in the track. This emphasizes our joint tracking and classification approach, as the uncertainty in both the tracking and classification class is directly considered through this cost function. For the remainder of this chapter the cost function will thus be defined as

$$\mathcal{C}(\boldsymbol{a}, \boldsymbol{s}_{k|k-1}, \boldsymbol{P}_{k|k-1}, c) = \Sigma_{\phi,\mathrm{tot}}, \tag{5.21}$$

where $\boldsymbol{s}_{k|k-1}$ is the predicted state and $\boldsymbol{P}_{k|k-1}$ is the predicted error-covariance for the considered target given by the tracking filter. Therefore, the predicted belief state is used as input for the cost calculation.

## 5.5. Assumed Radar Scenario

If not mentioned otherwise in the following, the general assumptions about the two-dimensional scenarios and the radar systems are the same as introduced in Chapter 3.6. Additionally, some further assumptions are being made in this chapter.

## 5.5.1. Target Classes

Each target is assumed to belong to a specific class. The class is defined before the simulation scenario starts and cannot be changed. Therefore, it stays the same during the entire scenario. The measurement variance regarding the class feature $f^n$ of object $n$ is calculated as explained in (3.18). The corresponding variance value for the reference measurement $f_0$ is shown together with the other simulation parameters in the specific subsection. For the simulations discussed below, different target classes are considered that have an influence on the maneuverability of the targets. Therefore, (2.3) needs to be adjusted and made class-dependent. The next state can be written as

$$\boldsymbol{s}^n_{k_n+1} = \boldsymbol{F}_n \boldsymbol{s}^n_{k_n} + \boldsymbol{w}^{n,c}_{k_n}, \tag{5.22}$$

with $\boldsymbol{F}_n \in \mathbb{R}^{4\times 4}$ defined as

$$\boldsymbol{F}_n = \begin{bmatrix} 1 & 0 & T_n & 0 \\ 0 & 1 & 0 & T_n \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.23}$$

and the maneuverability noise $\boldsymbol{w}^{n,c}$ with covariance

$$\boldsymbol{Q}_{n,c} = \begin{bmatrix} T_n^3/3 & 0 & T_n^2/2 & 0 \\ 0 & T_n^3/3 & 0 & T_n^2/2 \\ T_n^2/2 & 0 & T_n & 0 \\ 0 & T_n^2/2 & 0 & T_n \end{bmatrix} \sigma^2_{w,c}, \tag{5.24}$$

where $\sigma^2_{w,c}$ is the maneuverability noise variance for class $c$.These maneuverabilities are implemented in the trajectory simulations of the targets and are also taken into account in the resource optimization algorithm. Note that in this chapter, a different maneuverability noise covariance is assumed compared to the previous chapters. As already discussed earlier, one tracking filter per target class is applied, each tuned to one of the classes.

## 5.5.2. Optimization Problem

There are $N$ tracked targets in the environment. Equivalently to Section 3.6, the RRM problem can thus be expressed as

$$\begin{aligned} \underset{\boldsymbol{T},\boldsymbol{\tau}}{\text{minimize}} \quad & \sum_{n=1}^{N} E\left[C\left(\boldsymbol{s}^n_{k_n|k_n-1}(T_n, \tau_n)\,,\ \boldsymbol{P}^n_{k_n|k_n-1}(T_n, \tau_n)\,,\ c^n\right)\right] \\ \text{subject to} \quad & \sum_{n=1}^{N} \frac{\tau_n}{T_n} \leq \Theta_{max}. \end{aligned} \tag{5.25}$$

The cost that is optimized is therefore based on the current prediction of the tracking filter, which is based on the measurement actions $T$ and $\tau$. Similar to Chapter 3.6, an EKF is applied, and every detection is automatically assigned to the correct track. Both the revisit time $T$, as well as the dwell time $\tau$ are optimized. The state measurements are influenced by both $T$ and $\tau$, while the state-independent feature measurement is only influenced by the dwell time.

### 5.5.3. Threat Definition

Since a two-dimensional scenario is assumed, the dimension parameter in the unscented transform is $D = 2$. As mentioned before, the choice of the "correct" threat definition depends on the scenario and the user's wishes. As an example, in the following, the threat $\phi$ is defined as

$$\phi(c, \boldsymbol{s}) = \frac{\rho_c \cdot \left(0.1 + \exp\left(-\frac{r(\boldsymbol{s}) - r'}{\eta}\right)\right)}{1 + \exp\left(-\frac{r(\boldsymbol{s}) - r'}{\eta}\right)}, \tag{5.26}$$

where $\rho_c$ is a scalar factor unique for each class, $r(\boldsymbol{s}) = \sqrt{x^2 + y^2}$ is the range of the target from the sensor, $r' = 18km$ is a reference range and $\eta = 5000$ is a parameter to fine-tune the threat function slope. A possible example of such a threat is shown in Figure 5.2. This formulation assumes that targets at a long distance pose a very low threat, while the threat increases the closer the target advances towards the sensor location. At a certain distance, the maximum threat value is reached. In addition to that, some classes generally pose a higher threat than others. One could think of an automotive scenario where a vehicle is moving towards the sensor location. When it is far away, the threat would be low as it will probably turn away at some point. However, once it comes closer, the threat increases until it is not very likely to turn away anymore, which means that the maximum threat level is reached, and a collision is almost inevitable. Regarding the different classes, one could think of a truck having a higher base threat level than a cyclist, for instance.

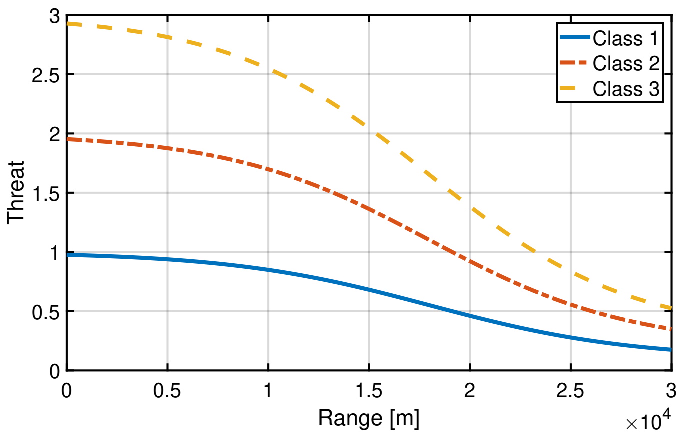


Figure 5.2: Example threat function for 3 different targets. The reference range $r'$ is 18 km and the tuning parameter $\eta$ is 5000. The class parameters $\rho$ are set to values 1,2 and 3.

## 5.6. Simulation Scenario A

In this section, the dynamic tracking example as presented in Chapter 3.8 is used to show the impact of the chosen cost function based on the threat. Essentially, the AODB algorithm from Chapter 3 is applied with the cost function as defined in (5.21). The radar sensor is placed at the origin of the coordinate system. Initially, there are four targets in the scene.

After 25 s, a fifth target is detected, and a new track is started. All targets move with constant velocities. Here, it is assumed that the class of the targets is not of interest, so no classification is applied during the simulation scenario. The simulation parameters are summarized in Table 5.1, while the target parameters are shown in Table 5.2. The trajectories of the targets during the simulation scenario are shown in Figure 5.3 and the resulting budget allocation of the simulations are shown in Figure 5.4.

Since classification is not considered in this example, the uncertainty in threat comes directly from the tracking accuracy. This is reflected in Figure 5.4 by the fact that the budgets overall show very similar behavior to the dashed lines. Those lines indicate the results from Chapter 3, where the error-covariance of the tracking filter was used directly to optimize the resource allocation. For example, Target 4 receives the largest budget allocation during the first 70% of the scenario, while Target 5 always receives the smallest, which is in line with the previous results. It should be noted that the algorithm decides the resource allocations on the expected threat variance reduction rather than the actual threat variance values. Nevertheless, the target with the highest threat variance will offer the biggest opportunity to reduce this threat variance in most cases.

Table 5.1: Simulation parameters for Simulation Scenario A.

| **Parameter** | **Value** |
|---|---|
| Precision of solution ($\delta$) | 0.01 |
| Action space discretization steps ($\Delta T$, $\Delta\tau$) | Adaptive |
| Action space limits revisit interval ($T_{min}$, $T_{max}$) | $T \in [0.1\text{ s} \ldots 5\text{ s}]$ |
| Action space limits dwell time ($\tau_{min}$, $\tau_{max}$) | $\tau \in [0.1\text{ s} \ldots \infty]$ |
| Number of rollouts ($M$) | 10 |
| Rollout horizon ($\mathcal{H}$) | 10 |
| Base Policy ($\pi_{base}$) | $\boldsymbol{a}$ |
| Maximum available budget ($\Theta_{max}$) | 1 |
| Budget update interval ($t_B$) | 5 s |
| Beam positioning error ($\Delta\alpha$) | 0 |
| Probability of detection ($P_D$) | 1 |
| Threat reference range ($r'$) | 18 km |
| Threat slope parameter ($\eta$) | 5000 |

## 5.7. Simulation Scenario B

In this section, a dynamic joint tracking and classification scenario is presented. The radar sensor is placed at the origin of the coordinate system. There are two possible classes and two observed targets. The first target is of Class 1, and the second one of Class 2. The radar sensor does not know these classes, which means that the initial class probabilities are equal for both classes for each target. The available budget is set to 1, implying that the radar system fully focuses on these two tracking tasks. The targets move with a class-typical maneuverability noise which can be seen from trajectories in Figure 5.5a. The simulation

Table 5.2: Initial target parameters for Simulation Scenario A.

| **Parameter** | **Target** | | | | |
|---|---|---|---|---|---|
| | **1** | **2** | **3** | **4** | **5** |
| $x_0^n$ [km] | 12 | 12 | 7 | 19 | 7.9 |
| $y_0^n$ [km] | 10 | 15 | 11 | 2 | 8.3 |
| $\dot{x}_0^n$ [m s$^{-1}$] | 9 | -30 | 45 | -35 | -20 |
| $\dot{y}_0^n$ [m s$^{-1}$] | -15 | 15 | 30 | 0 | -25 |
| $\varsigma^n$ [m$^2$] | 25 | 25 | 64 | 64 | 64 |

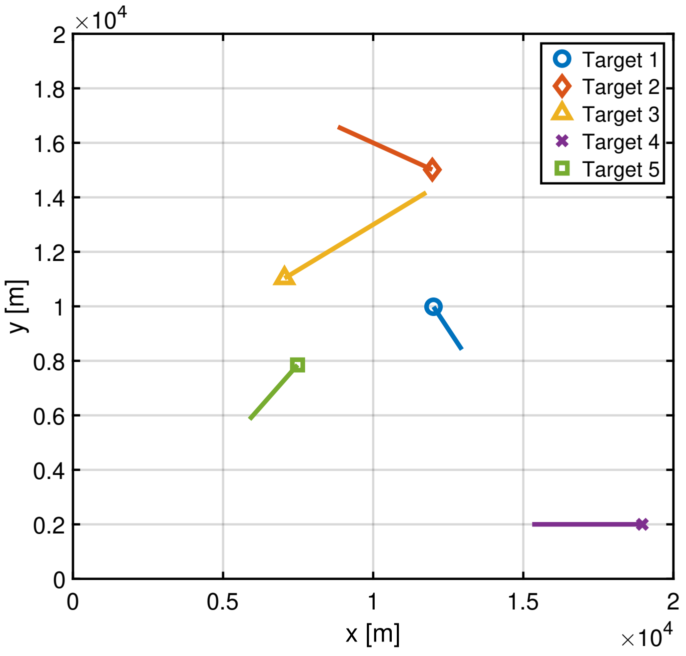


Figure 5.3: Trajectories of the targets for Simulation Scenario A. The symbols mark the starting positions.

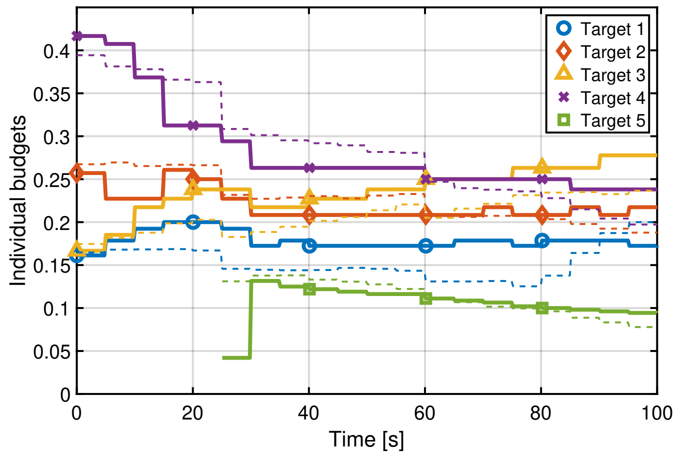


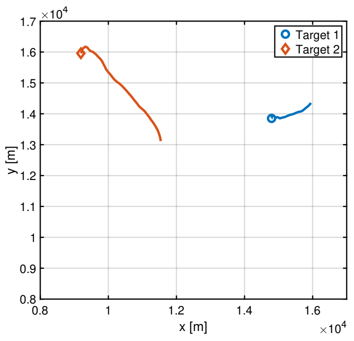
Figure 5.4: Resulting budget distribution for Simulation Scenario A. The dashed lines denote the results for tracking without classification as shown in Figure 3.7a.

parameters are identical with the ones for Simulations A as mentioned in Table 5.1. The target and class parameters are shown in Tables 5.3 and 5.4, respectively. The simulation results are presented in Figures 5.5b to 5.5f.

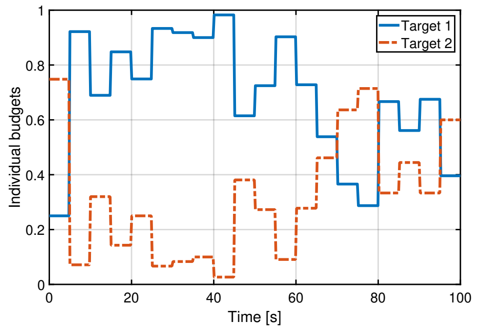
In the beginning, Target 2 gets a larger amount of dwell time assigned than Target 1, which leads to a quick classification. Target 2 is closer to the radar sensor than Target 1, which means that not knowing the class leads to a higher threat variance. Additionally, the feature measurements for Target 2 are more accurate than for Target 1 due to the smaller distance and, therefore, higher SNR. Subsequently, after 5 s, the dwell time and with it the relative budget for Target 2 drops, while Target 1 gets significantly more dwell time and budget assigned. During the bigger part of the scenario, the sensor focuses on classifying Target 1, which is more difficult due to its larger distance from the sensor. While Target 1 gets slowly classified, its budget starts to decrease after about 45 s. The budget for Target 2 increases at the same time. After both targets are successfully classified at about 70 s, the assigned budgets for both targets stay around 0.5, although Target 1 still gets a higher dwell time. The targets are both classified and theoretically deserve a similar amount of attention. However, as Target 1 was classified later, there is still slightly more uncertainty left about its class, leading to a higher dwell time allocation. This behavior emphasizes that the resource allocation is based on the expected threat variance reduction. Figure 5.6e shows how the early classification of Target 2 leads to a significant direct decrease in threat variance, while the classification of Target 1 takes longer, and the cost therefore also drops slower.

## 5.8. Simulation Scenario C

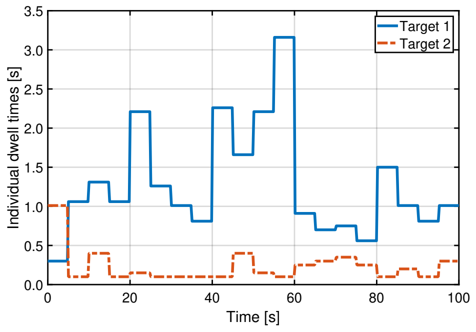
This simulation scenario is similar to Scenario B. The radar sensor is again placed at the origin of the coordinate system, and the available maximum budget is set to $\Theta_m ax = 0.5$. The reason for a lower maximum budget could be, e.g., that an operator of the radar system manually assigned some of the total budget to other tasks. Additionally, the length of the



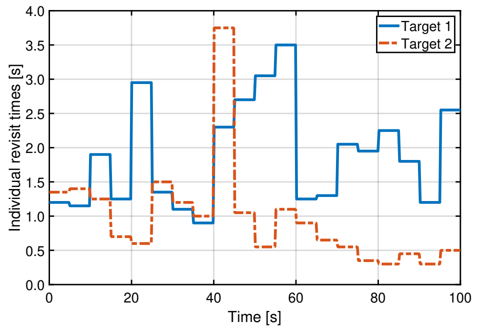
(a) Trajectories of the targets for Simulation Scenario B. The symbols mark the starting positions.



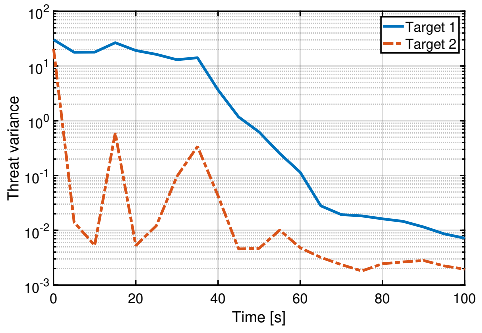
(b) Resulting budget distribution.



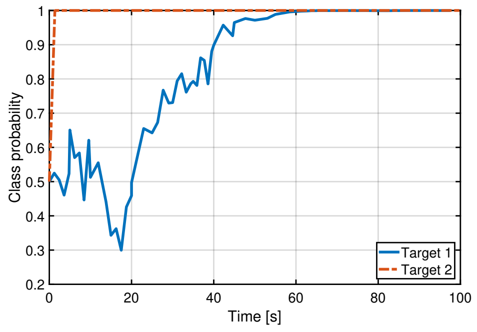
(c) Resulting dwell time distribution.



(d) Resulting revisit time distribution.



(e) Resulting optimized cost (threat variance).



(f) Resulting class probabilities. A value of 1 means that a target was correctly classified.

Figure 5.5: Trajectories and simulation results for Simulation Scenario B.

Table 5.3: Initial target parameters for Simulation Scenario B.

| **Parameter** | **Target** | |
|---|---|---|
| | **1** | **2** |
| $x_0^n$ [km] | 14.8 | 9.2 |
| $y_0^n$ [km] | 13.9 | 15.9 |
| $\dot{x}_0^n$ [m s$^{-1}$] | 2 | 2 |
| $\dot{y}_0^n$ [m s$^{-1}$] | -1 | 1 |
| $\varsigma_n$ [m$^2$] | 5 | 5 |
| $c^n$ | 1 | 2 |

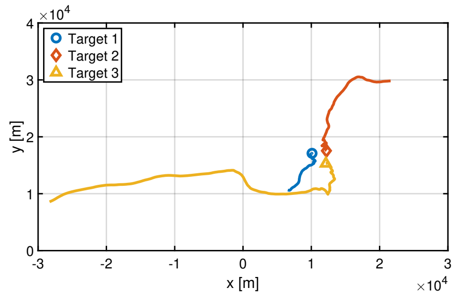
Table 5.4: Class parameters for Simulation Scenario B.

| **Parameter** | **Class** | |
|---|---|---|
| | **1** | **2** |
| Class feature $f_c$ | 1 | 2 |
| Threat parameter $\rho_c$ | 1 | 9 |
| Maneuverability $\sigma_{w,c}$ [m s$^{-2}$] | 2 | 5 |

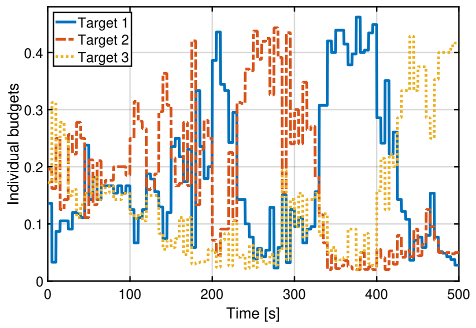
simulation is 500 s, which is longer than in Scenario B. All other general simulation parameters are the same as shown in Table 5.1. The initial target parameters are shown in Table 5.5 and the class parameters are summarized in Table 5.6. This time, there are three targets in the environment, and the targets can be of three possible classes. Figure 5.6a shows the trajectories of the simulated targets. The simulation results are shown in Figures 5.6b to 5.6f.

At the beginning of the scenario, it can be seen that Target 3 gets the largest relative budget assigned. Subsequently, it gets classified very quickly. It can be seen that the algorithm makes a wrong decision about the class of Targets 1 and 2. Figure 5.6e shows that making the first classification decisions leads to a large reduction of the calculated threat variance for all targets within the first 100 s. It can be seen that while the algorithm slowly classifies Target 2 between about 100 s and 300 s, the threat variance increases and then drops again. The reason is that the class probabilities are shifting during that phase, and there is no clear decision made yet. The same happens to Target 1, as its class probability values are also changing at that time. Between about 320 s and 420 s Target 1 is classified correctly, which also leads to an increased threat variance. After 400 s, all targets are correctly classified, and the threat variances decrease rapidly.

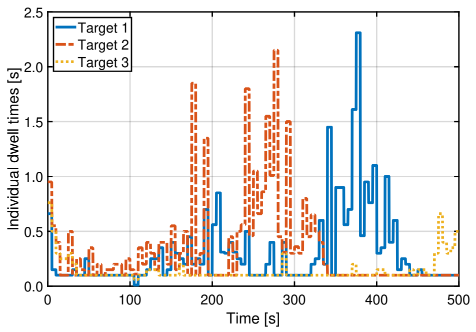
In Figure 5.6b, it can be seen that the budget allocations roughly follow the threat variances. The target with the highest threat variance usually receives the largest budget. Similarly, the dwell times are assigned approximately proportional to the threat variance.



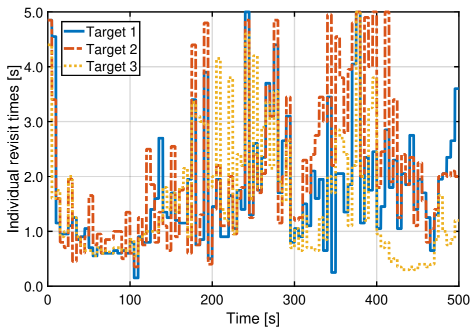
(a) Trajectories of the targets for Simulation Scenario C. The symbols mark the starting positions.



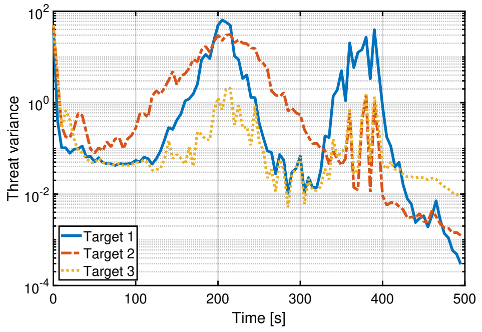
(b) Resulting budget distribution.



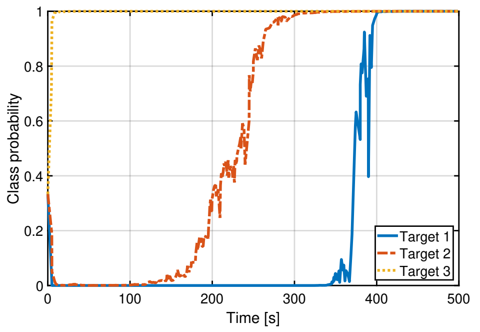
(c) Resulting dwell time distribution.



(d) Resulting revisit time distribution.



(e) Resulting optimized cost (threat variance).



(f) Resulting class probabilities. A value of 1 means that a target was correctly classified.

Figure 5.6: Trajectories and simulation results for Simulation Scenario C.

Table 5.5: Initial target parameters for Simulation Scenario C.

| **Parameter** | **Target** | | |
|---|---|---|---|
| | **1** | **2** | **3** |
| $x_0^n$ [km] | 10.1 | 12.3 | 12.1 |
| $y_0^n$ [km] | 17.1 | 17.5 | 15.3 |
| $\dot{x}_0^n$ [m s$^{-1}$] | 1 | -2 | 1 |
| $\dot{y}_0^n$ [m s$^{-1}$] | 2 | 2 | -2 |
| $\varsigma_n$ [m$^2$] | 5 | 5 | 5 |
| $c^n$ | 1 | 2 | 3 |

Table 5.6: Class parameters for Simulation Scenario C.

| **Parameter** | **Class** | | |
|---|---|---|---|
| | **1** | **2** | **3** |
| Class feature $f_c$ | 1 | 2 | 3 |
| Threat parameter $\rho_c$ | 1 | 9 | 19 |
| Maneuverability $\sigma_{w,c}$ [m s$^{-2}$] | 2 | 5 | 9 |

## 5.9. Conclusions

This chapter introduced a novel RRM approach for joint tracking and classification using the framework presented in Chapter 3. In contrast to most available approaches, two different task types are combined into one. It is shown that it is possible to solve the RRM problem for multiple task types by using only a single cost function based on a definition of mission threat. Such approaches have been suggested previously but have never been fully developed and demonstrated with the help of practical simulation scenarios.

Firstly, the joint tracking and classification framework has been introduced, which builds on the previous RRM framework as shown in Chapter 3.

Secondly, it has been explained how to move from the state to the threat domain and combine the cost of different target classes. The idea of threat is to transform the state of each task into an easily comparable scalar number.

Finally, an explicit definition of a possible mission threat has been introduced. The presented threat definition is based on the position and a class-dependent parameter. It has been shown how the threat looks like for multiple classes in a two-dimensional environment.

Through an analysis of the dynamic tracking Scenarios A to C, it has been shown that the algorithm works in different situations. It calculates the resource allocations based on the class probabilities and the tracking state accuracy. The algorithm tries to classify targets of unknown classes faster, especially when they are close to the radar sensor. The classification is done over time while tracking for targets that are further away and have a smaller threat variance. Once the targets are classified, the resource allocations depend primarily on the track uncertainty. This means that the target tracks get the resources assigned based on the expected decrease in uncertainty. The presented simulations confirm the correctness of the proposed algorithm.

# 6 Radar Resource Management for Multi-Sensor Multi-Target Tracking

*As shown in the previous chapters, the suggested RRM approach delivers promising results for single sensors. However, in many applications multiple sensors are cooperating in order to observe the environment. One could think of automotive scenarios with multiple cars or air surveillance with sensors at different locations. Therefore, this chapter extends the previous approach to include multiple connected sensors. The applicability of the generic framework and the proposed algorithmic solutions is shown through multiple dynamic simulation scenarios.*

Parts of this chapter have been published in:

B. van der Werk, M. I. Schöpe, and H. Driessen, "Approximately Optimal Radar Resource Management for Multi-Sensor Multi-Target Tracking", in *Proceedings of the 24th International Conference on Information Fusion (FUSION)*, Sun City, North West, South Africa, 2021, accepted for publication.

K. Jayachandra, M. I. Schöpe, and A. Yarovoy, "Joint Radar Communication Resource Allocation for Automotive Applications Using Policy Rollout", *IEEE Transactions on Vehicular Technology*, 2021, under review.

The ideas for this chapter and the general solution approaches originated from Max Ian Schöpe. The detailed implementation of the proposed algorithms has been carried out by the MSc students Bas van der Werk and Karan Jayachandra under the close daily supervision of Max Ian Schöpe. The students wrote the publication drafts with a significant amount of suggestions and corrections by Max Ian Schöpe. Hans Driessen provided additional input during the whole process.

## 6.1. Literature Review

The majority of RRM approaches focus on single sensor systems, as shown in the previous chapters. However, many modern applications of sensor systems require the cooperation of multiple sensors [94]. Many previous approaches to RRM for sensor networks have been focusing on sensor selection without resource balancing (see, e.g., [58, 95, 96]), which usually means that only the one sensor that results in the best measurement is chosen for the task in question. Bogdanović et al. show a game-theoretic approach to sensor selection in [97]. Another game-theoretic approach has been presented by Wang et al. and optimizes the time, and aperture for Inverse Synthetic Aperture Radar [98]. However, both approaches do not formulate the problem as a budget-balancing problem. In [99], Shi et al. use Deep RL to solve an underlying MDP for optimizing the selection of sensors. This is done in order to achieve a certain tracking accuracy while minimizing the power consumption. Also, that approach does not apply resource balancing. In addition to that, it only delivers a myopic solution. Another RRM approach to a network scenario has been presented by Han et al. [100]. It aims at decreasing the sensing time of the individual sensors while keeping the sensing performance at some desired level. This is done to free up sensor resources for extra communication tasks. Such an approach is not desirable for many applications since it does not lead to optimal measurement accuracy. In [101], Bell et al. developed an RRM solution that balances the resources for sensor networks. However, that approach does not exploit a non-myopic POMDP framework and only demonstrates results for a single-target tracking scenario.

A simple sensor selection ignores a part of the potential of an adaptive sensor network. Therefore, this chapter strives to optimize the actions for each sensor while taking into account the global mission, the local sensor constraints, and the expected future situation through the POMDP framework. The results presented in this chapter are an outcome of the master's theses of Bas van der Werk [102] and Karan Jayachandra [103] which have also been published as conference papers in [104, 105].

The remainder of this chapter is structured as follows. Section 6.2 describes the considered RRM problem for a general multi-target multi-sensor tracking scenario and defines the optimization problem. Subsequently, Section 6.3 proposes a distributed approach to solve the described RRM problem. Simulation results are provided that show the performance of the proposed method. Furthermore, Section shows a possible approach for an automotive scenario where both time and frequency resources are allocated. The algorithm decides to share information with a central system when they reduce uncertainty. Simulation results show the applicability of the approach. Finally, the conclusions can be found in Section 6.5.

## 6.2. Problem Definition

This chapter is largely based on the assumptions that were introduced in Chapter 2.2, which need to be extended for multiple sensors. This is done in the following. As before, an underlying POMDP is assumed to describe the development of the targets states.

### 6.2.1. Measurement Model

The state of the targets cannot be observed directly but is observed through noisy measurements using $M$ sensors. A measurement of target $n$ at time $t$ can be defined as,

$$\boldsymbol{z}_t^{m,n} = \mathfrak{h}(\boldsymbol{s}_t^n, \boldsymbol{v}_t^{m,n}, \boldsymbol{a}_t^{m,n}), \tag{6.1}$$

where $\boldsymbol{v}_t^{m,n}$ is the zero-mean Gaussian measurement noise and $\boldsymbol{a}_t^{m,n}$ is the action that is executed at sensor $m$ for target $n$ at time $t$. This measurement function directly defines a measurement PDF given as

$$p(\boldsymbol{z}_t^{m,n} | \boldsymbol{s}_t^n, \boldsymbol{a}_t^{m,n}). \tag{6.2}$$

In the following, it is assumed that measurements are taken in time steps $k$. As it simplifies the equations, the notation of those time steps does not reflect that they can be of different lengths for different sensors and tasks.

### 6.2.2. Tracking Algorithm

As mentioned in previous chapters, any filter that calculates the posterior density can be applied. For linear systems, this can be a KF, while the EKF or a particle filter are applicable methods for non-linear systems.

In the following, it is assumed that the sensors can communicate with a central processor that exploits the information received from all sensors. Measurement fusion is applied by using an update scheme in which the central processor updates the track estimate recursively for each measurement of the individual sensors. The result is a fused global estimate for the state $\boldsymbol{s}_{k|k}^{\mathfrak{f}}$ and for the covariance $P_{k|k}^{\mathfrak{f}}$. The updating process for one certain target using all $M$ sensors is summarized in Algorithm 1, where $\boldsymbol{R}$ is the measurement variance matrix and $\boldsymbol{H}$ is the observation matrix.

**Algorithm 1:** Measurement fusion in central processor.

```
Input $\boldsymbol{P}_{k|k-1}^{\mathfrak{f}}$, $\boldsymbol{s}_{k|k-1}^{\mathfrak{f}}$, $\boldsymbol{R}$, $\boldsymbol{H}$, $\mathfrak{h}$
$\boldsymbol{P} = \boldsymbol{P}_{k|k-1}^{\mathfrak{f}}$
$\boldsymbol{s} = \boldsymbol{s}_{k|k-1}^{\mathfrak{f}}$
$m = 1$
while $m < M$ do
    $\boldsymbol{P} = \text{update}_{\text{covariance}}(\boldsymbol{R}^m, \boldsymbol{H}^m, \boldsymbol{P})$
    $\boldsymbol{s} = \text{update}_{\text{state}}(\boldsymbol{R}^m, \boldsymbol{H}^m, \boldsymbol{P}, \boldsymbol{z}^m, \boldsymbol{s})$
    $m = m + 1$
end
$\boldsymbol{P}_{k|k}^{\mathfrak{f}} = \boldsymbol{P}$
$\boldsymbol{s}_{k|k}^{\mathfrak{f}} = \boldsymbol{s}$
Return $\boldsymbol{P}_{k|k}^{\mathfrak{f}}$, $\boldsymbol{s}_{k|k}^{\mathfrak{f}}$
```

In the following, the tracking process is based on the sensing information of all sensors following the process in Algorithm 1. It is assumed that all sensors produce independent measurements.

### 6.2.3. Optimization Problem

Equivalently to the formulation in (2.12), the goal is to minimize a cost function $\boldsymbol{\mathcal{C}}$ while each sensor $m$ only has a limited resource budget $b_m^{max}$ available. The optimization problem can be formulated as

$$\begin{aligned} &\underset{\boldsymbol{A}}{\text{minimize}} \quad E\left[\mathbf{1}^T \boldsymbol{\mathcal{C}}\right] \\ &\text{subject to} \quad \underbrace{\begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1N} \\ b_{21} & b_{22} & & \\ \vdots & & \ddots & \\ b_{M1} & & & b_{MN} \end{bmatrix}}_{\boldsymbol{B}(\boldsymbol{A})} \cdot \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \leq \underbrace{\begin{bmatrix} b_1^{max} \\ b_2^{max} \\ \vdots \\ b_M^{max} \end{bmatrix}}_{\boldsymbol{B}_{max}} , \end{aligned} \tag{6.3}$$

where

$$\boldsymbol{\mathcal{C}} = \left[\mathcal{C}_1(\boldsymbol{A}, \boldsymbol{s}^1) \quad \mathcal{C}_2(\boldsymbol{A}, \boldsymbol{s}^2) \quad \cdots \quad \mathcal{C}_N(\boldsymbol{A}, \boldsymbol{s}^N)\right]^T \tag{6.4}$$

represents the cost related to all targets, $\mathbf{1} \in \mathbb{R}^N$ is a vector of all ones and

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{a}^{1,1} & \boldsymbol{a}^{1,2} & \cdots & \boldsymbol{a}^{1,N} \\ \boldsymbol{a}^{2,1} & \boldsymbol{a}^{2,2} & \cdots & \boldsymbol{a}^{2,N} \\ \vdots & \dots & \ddots & \vdots \\ \boldsymbol{a}^{M,1} & \boldsymbol{a}^{M,2} & \cdots & \boldsymbol{a}^{M,N} \end{bmatrix}, \tag{6.5}$$

is the action matrix for all targets and sensors which is being optimized. The individual budgets $\mathrm{b}_{m,n}$ inside the resource budget matrix $\boldsymbol{B}(\boldsymbol{A})$ represent a percentage of the maximum budget $b_m^{max}$ spent by sensor $m$ on target $n$.

## 6.3. AODB for Multi-Sensor Scenarios

Sections 6.3.1 and 6.3.2 introduce two different implementations of the RRM algorithm to optimize the sensing resources.

### 6.3.1. Approximately Optimal Approach

The optimal approach is to optimize all actions of all sensors jointly through a central processor which takes care of the tracking, as well as the resource optimization. The proposed solution is based on this idea. The sensor resources for target $n$ are allocated by optimizing a cost $\mathcal{C}_n$ which depends on the actions of multiple sensors. Based on (6.4) at time $k$ this can be written as

$$\mathcal{C}_n(\boldsymbol{A}_k, \boldsymbol{s}_k^n) = \mathcal{C}_n(\boldsymbol{a}_k^{1,n}, \boldsymbol{a}_k^{2,n}, \cdots, \boldsymbol{a}_k^{M,n}, \boldsymbol{s}_k^n). \tag{6.6}$$

The problem can be decomposed into $N$ parallel sub-optimization problems using LR. The optimal actions for all $M$ sensors related to target $n$ are computed using a global PR in the central processing node. This centralized implementation optimizes a global policy per task to explore the actions of all sensors at the same time. Hence, the action space will be of size $D^M$ where $D$ is the number of actions each sensor can choose from. Increasing the number of sensors will result in an exponential increase of the action space. In practice, when described as a POMDP, this problem would be very complicated to solve exactly for a large number of possible states and actions. Therefore, it is proposed to solve it approximately by applying PR.

### 6.3.2. Distributed Implementation

Because of the increased computational complexity with growing number of sensors, a practical implementation is proposed which allows each sensor to solve a part of the problem separately. Instead of sharing its sensing information with a central processor, each sensor node broadcasts it to all sensors in the network. Each sensor then functions as a processing node on its own to compute fused estimates and to find the best possible actions $\boldsymbol{a}^{m,n}$ for that specific sensor. This also means that the tracking will be done in each sensor node separately. The cost $C_n$ for target $n$ can be decomposed into a summation of costs $c_{m,n}$ from multiple sensors as

$$C_n(\boldsymbol{A}_k, \boldsymbol{s}_k^n) = c_{1,n}(\boldsymbol{a}_k^{1,n}, \boldsymbol{I}_k^{1,n}, \boldsymbol{s}_k^n) + c_{2,n}(\boldsymbol{a}_k^{2,n}, \boldsymbol{I}_k^{2,n}, \boldsymbol{s}_k^n) + \cdots + c_{M,n}(\boldsymbol{a}_k^{M,n}, \boldsymbol{I}_k^{M,n}, \boldsymbol{s}_k^n). \quad (6.7)$$

Here, $\boldsymbol{I}_k^{m,n}$ represents the information that sensor $m$ received from all other sensors about target $n$ at time $k$. The distributed optimization problem is therefore defined as

$$\begin{aligned} \underset{\boldsymbol{A}_k}{\text{minimize}} \quad & \textstyle\sum_{n=1}^{N} \sum_{m=1}^{M} E\left[c_{m,n}(\boldsymbol{a}_k^{m,n}, \boldsymbol{I}_k^{m,n}, \boldsymbol{s}_k^n)\right] \\ \text{subject to} \quad & \boldsymbol{B}(\boldsymbol{A}_k) \cdot \boldsymbol{1} \leq \boldsymbol{b}_{max}. \end{aligned} \quad (6.8)$$

The problem is decomposed into a sub-optimization problem per task per sensor using LR. By doing so, each individual PR optimization only needs to explore the action space w.r.t. a single sensor and not a combination of all sensors.

As the PR is making predictions of the expected future for a single sensor, it does not have access to the currently optimized actions of all other sensors during the resource optimization process. In this implementation it is therefore assumed that the sensors use the last known optimized actions of the other sensors as input and calculate their expected impact on the cost. Hence, the information term $\boldsymbol{I}_k^{m,n}$ is defined as the last known actions of the other sensors and can be written as

$$\boldsymbol{I}_k^{m,n} = [\boldsymbol{a}_{k-1}^{1,n} \cdots \boldsymbol{a}_{k-1}^{m-1,n} \boldsymbol{a}_{k-1}^{m+1,n} \cdots \boldsymbol{a}_{k-1}^{M,n}]. \quad (6.9)$$

The additional communication overhead is assumed to be negligible compared to the reduction in computation time required for the PR. In the proposed approach, the optimization stops once each sensor has found the optimal solution given the previous actions of the other sensors. In case the sensors can also communicate efficiently during the optimization procedure, this step could be repeated multiple times. After each iteration, the sensors could then exchange the optimized actions until the solution converges to a steady result. The distributed implementation is assumed to converge to similar results as the centralized approach described in the previous section, as long as the situation doesn't change too much between exchanging optimized actions. A high-level block diagram of the algorithm is presented in Figure 6.1 showing the distributed approach.

### 6.3.3. General Simulation Assumptions

If not stated otherwise, the assumptions for the following simulations are the same as in Chapter 3. The assumed radar system is of type A as shown in Table 3.1. For the tracking of the simulated targets, an EKF is applied.

A two-dimensional radar tracking scenario is assumed. Measurements are taken in range and angle. For every sensor, the algorithm calculates the optimal dwell time $\tau$. The revisit

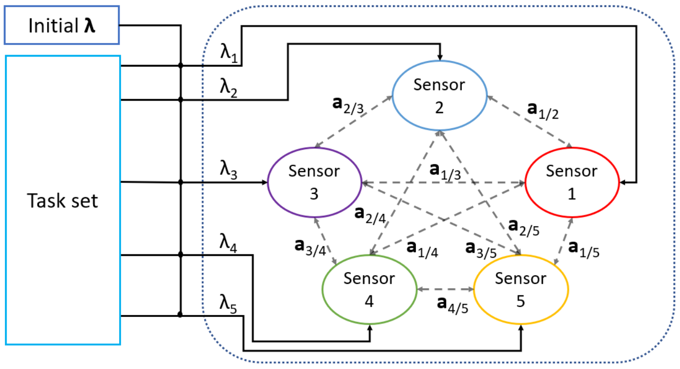


Figure 6.1: High level block diagram of the distributed implementation. Each sensor computes their budget allocations via the PR and then shares their last known actions with the other sensors.

time $T$ is assumed to be constant. In between budget allocation updates, the actions are assumed to remain unchanged. For all tasks per sensor, the budget allocation is computed such that the sum of the actions meets the resource constraint. In this implementation, the algorithm does not execute an explicit scheduling of all the tasks into the sensor timelines.

### 6.3.4. Observation Model

The observation model and the influence of the SNR on the measurements for each sensor are the same as explained in Chapter 3.6.3. In the following, the multi-sensor notation will be added.

An observation of target $n$ made by sensor $m$ is therefore given by

$$\boldsymbol{z}_k^{m,n} = \boldsymbol{h}^m(\boldsymbol{s}_{k'}^n) + \boldsymbol{v}_k^{m,n}, \tag{6.10}$$

where $\boldsymbol{h}^m(\boldsymbol{s}_k^n)$ is the measurement transformation function for sensor $m$ and $\boldsymbol{v}_k^{m,n}$ is the measurement noise for sensor $m$ sensing target $n$. The measurement noise for range and angle are assumed to be independent of each other which can be written as

$$\boldsymbol{v}_k^{m,n} = [v_r^{m,n}, v_\theta^{m,n}]^T, \tag{6.11}$$

with corresponding variances $\sigma_r^2$ and $\sigma_\theta^2$. The measurement transformation function which transforms the Cartesian measurements into polar measurements is defined as

$$\boldsymbol{h}^m(\boldsymbol{s}_k^n) = \begin{bmatrix} \sqrt{(x_k^n - x'_m)^2 + (y_k^n - y'_m)^2} \\ \text{atan2}(y_k^n - y'_m, x_k^n - x'_m) \end{bmatrix}, \tag{6.12}$$

where $x'_m$ and $y'_m$ are the location of sensor $m$ in Cartesian coordinates.

The observation matrix $\boldsymbol{H}_k^{m,n} \in \mathbb{R}^{2\times 4}$ for sensor $m$ observing target $n$ is defined as the Jacobian of the measurement transformation function $\boldsymbol{h}$ evaluated at the current predicted target state $\boldsymbol{s}_{k|k-1}^n$, which can be written as

$$\boldsymbol{H}_k^{m,n} = \frac{\delta \boldsymbol{h}^m}{\delta \boldsymbol{s}}\bigg|_{\boldsymbol{s}_{k|k-1}^n}. \tag{6.13}$$

### 6.3.5. Cost Function and Constraint

In both the centralized and distributed approach the cost $C_n$ is based on the predicted error covariance. The current predicted error covariance at time step $k$ is defined as

$$\boldsymbol{P}^n_{k+1|k} = \boldsymbol{F}\boldsymbol{P}^n_{k|k}\boldsymbol{F}^T + \boldsymbol{Q}_n, \tag{6.14}$$

where $\boldsymbol{F}$ is the state transition matrix, $\boldsymbol{Q}_n$ is the zero-mean Gaussian process noise covariance for target $n$ and $\boldsymbol{P}^n_{k|k}$ is the estimated error covariance for target $n$, which also depends on sensor $m$ in the distributed implementation.

The cost for target $n$ at time step $k$ inside the PR is defined to be the trace of the positional elements of the error covariance.

$$c_n = \begin{bmatrix} 1 & 1 & 0 & 0 \end{bmatrix} \cdot \mathrm{diag}(\boldsymbol{P}^n_{k+1|k}) \tag{6.15}$$

The individual budgets $b_{m,n}$ are defined as the ratio of the dwell time over the revisit time per sensor per target:

$$b_{m,n} = \frac{\tau_{m,n}}{T_{m,n}}. \tag{6.16}$$

In the following, the performance of the distributed implementation is evaluated in a radar tracking scenario. In addition to that, a comparison is made based on the resulting cost and computation time.

6

### 6.3.6. Simulation Parameters

The general simulation parameters used for the simulations are shown in Table 6.1. Two radar sensors are considered. The maximum budget for each sensor is set to 1. The budget allocation is recalculated every 20 seconds, and there is a total of 40 budget updates, which corresponds to a simulated scenario of 800 s in length. Between budget updates, measurements are taken using the currently allocated budgets and subsequently shared with either the central processor or the other sensors. The revisit interval $T$ is constant.

Table 6.1: General Simulation Parameters.

| **Parameter** | **Value** |
|---|---|
| Maximum Budget ($b_n^{max}$) | 1 |
| Budget update interval ($t_B$) | 20 s |
| Simulation steps ($\Delta t_s$) | 40 |
| Simulation length ($\mathfrak{S}$) | 800 s |
| Beam-positioning error ($\Delta\alpha$) | 0 |
| Probability of Detection ($P_D$) | 1 |
| Precision of LR ($\delta_{LR}$) | 0.05 |
| Action discretization ($\Delta\tau$) | 0.01 s |
| Number of rollouts ($\mathcal{N}_r$) | 4 |
| Rollout horizon Length ($\mathcal{H}$) | 15 |
| Base Policy ($\pi_{base}$) | $\boldsymbol{a}$ |
| Number of sensors ($M$) | 2 |

The chosen BP is defined as the evaluated action at each step in the PR ($\pi_{base} = \boldsymbol{a}_k$). The PR has a horizon length of 15 measurement steps. Each evaluation of the PR is repeated four times and averaged for the final result. The actions to optimize are the dwell times $\tau$ for each sensor. They are selected from a one-dimensional discrete action space. The discretization for the dwell time $\Delta\tau$ is defined to be 0.01 seconds.

## 6.3.7. Simulation A: Dynamic Tracking Scenario

A dynamic radar tracking scenario is considered involving six non-maneuvering targets with constant velocities. The placement of the sensors and the trajectories of the targets is given in Figure 6.2. For Targets 1 to 7, the velocities, the RCS, and the maneuverability variances that are assumed in the EKF are shown in Table 6.2.

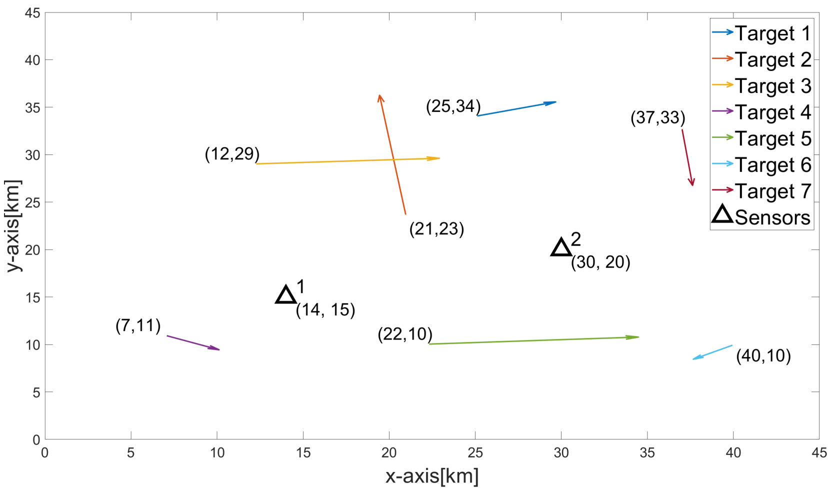


Figure 6.2: Visualization of the dynamic radar tracking scenario involving six targets and two sensors. The initial starting locations are shown at the beginning of each trajectory.

Table 6.2: Target parameters.

| **Target** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
|---|---|---|---|---|---|---|---|
| $V_x$ [m/s] | 30 | -10 | 70 | 20 | 80 | -15 | 4 |
| $V_y$ [m/s] | 10 | 85 | 4 | -10 | 5 | -10 | -40 |
| $\varsigma$ [m$^2$] | 50 | 20 | 90 | 80 | 20 | 100 | 40 |
| $\sigma_w^2$ [m$^2$ s$^{-2}$] | 13 | 24 | 15 | 22 | 17 | 11 | 9 |

The simulation runs for 40 simulation steps, where each step corresponds to the length of the budget update interval. Before each simulation step, the sensors' budget allocation is computed using the PR based on the distributed solution. This allocation is then applied to the sensors for 20 s until a new resource optimization is started.

Figure 6.3 shows how the cost changes during the first resource optimization procedure at the beginning of the simulation. Both the primal cost $\mathbf{1}^T\boldsymbol{C}$ and the dual cost $Z_D$ are shown.

Both the primal and dual cost converge in 25 LR iterations to the final values. There is still a so-called duality gap between primal and dual due to the choice of the solution precision.

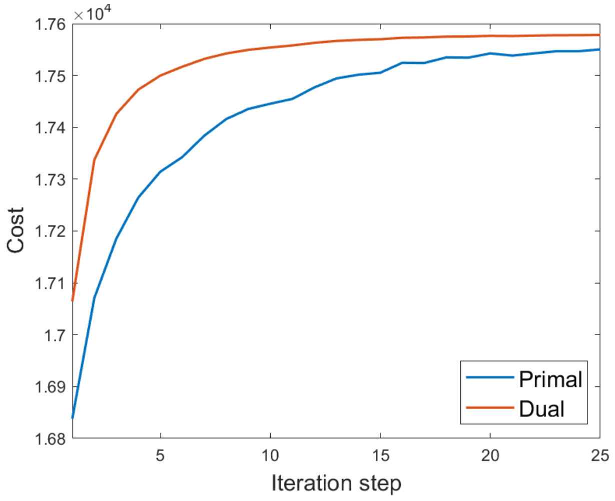


Figure 6.3: Evolution of the primal and dual cost from the distributed implementation during the first budget allocation process.

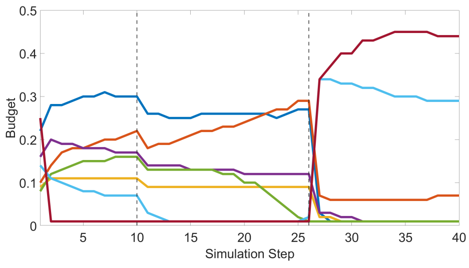
Figures 6.4a and 6.4b show the resulting budget allocations over multiple resource allocation steps for Sensor 1 and Sensor 2 respectively. Initially, Sensor 2 spends a significant amount of resource budget on Targets 6 and 7. Sensor 1, on the other hand, spends its budget more evenly over Targets 1 to 5 while not spending much attention on Targets 6 and 7. The figures indicate that the resources are allocated jointly to both sensors, based on the range-dependent measurement SNR.

At time step $\Delta t_s = 10$, the maximum budget of Sensor 1 is decreased to 0.8 which is also reflected in Figure 6.4a. At time step $\Delta t_s = 26$, Sensor 2 cannot track Targets 6 and 7 anymore due to, e.g., a restriction of the scanning angle. Consequently, Sensor 2 spends a significant amount of budget on the other targets, while Sensor 1 compensates for the missing information from Sensor 2 by spending most of its resources on Targets 6 and 7. Hence, by allowing communication between sensors, the sensor network is able to cope with sudden changes in the number of targets that the sensors can track.

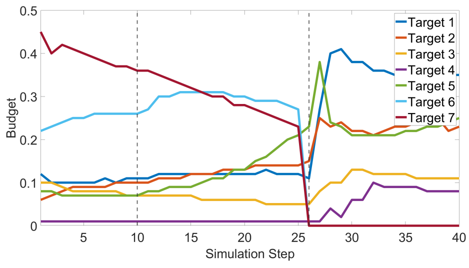
### **6.3.8.** Simulation B: Comparison of Implementations

In this simulation section, a comparison is made between the centralized, the distributed, and a third independent implementation. The independent implementation applies multi-sensor tracking as explained in Section 6.2.2, but applies the RRM algorithm from Chapter 3 for each sensor individually. There is a central tracking filter, but the individual resource optimizations for each sensor do not consider the other sensors' presence. For calculating the total cost of the independent solution, the cost from all sensors is summed up.

To compare different solution approaches, four dynamic scenarios are considered. Each scenario consists of six non-maneuvering targets with constant velocities, starting at different locations and two stationary sensors. For each scenario, 10 simulation steps $\Delta t_s$ are



(a) Budget allocation for Sensor 1.



(b) Budget allocation for Sensor 2.

Figure 6.4: Budget allocations over multiple simulation steps. At time step $\Delta t_s = 10$ the maximum budget for Sensor 1 is reduced to 0.8. At time step $\Delta t_s = 26$ Sensor 2 can no longer track Targets 6 and 7.

computed for all three solutions. In order to get a better comparison, each scenario is averaged over 10 simulation runs. Figure 6.5 shows the average primal and dual cost summed over all simulation steps of the simulation for the three implementations. It can be seen that both the centralized and distributed implementation outperform the independent implementation w.r.t. the primal and dual cost. Note that the costs for the centralized and distributed approach are formulated slightly differently. To make the costs of the different approaches comparable, the average cost per target track is evaluated. This is done by multiplying the total costs for the centralized and independent implementation by ${}^{1}/_{N}$ and for the distributed implementation by ${}^{1}/_{(MN)}$.

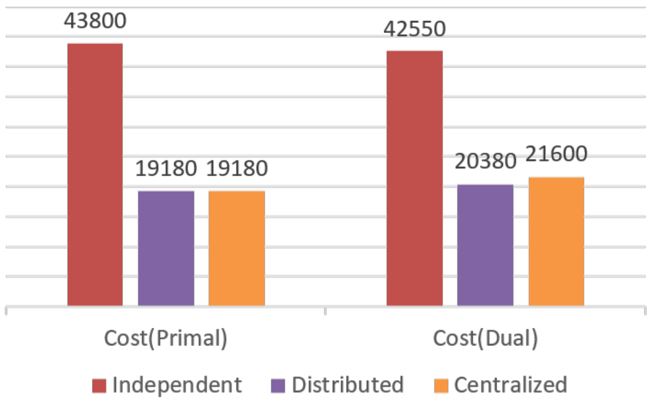


Figure 6.5: Cost comparison between the independent, centralized and distributed implementation averaged over multiple dynamic scenarios. The results are compared based on the average primal cost (left) and average dual cost (right).

Table 6.3 shows the average runtime of the four scenarios. As expected, due to the exponential increase of the action space for an increasing number of sensors, the average runtime of the centralized implementation is significantly larger w.r.t. the other two implementations. Interestingly, the average runtime of the distributed implementation is smaller than the independent implementation. This is probably due to the initial pick of the Lagrangian multiplier.

Table 6.3: Runtime comparison of independent, centralized and distributed implementation averaged over multiple runs.

| Approach | Runtime in seconds |
|---|---|
| Independent | 158 |
| Distributed | 113 |
| Centralized | 1187 |

It can be seen that the costs of the centralized en distributed implementation are approximately equal. This implies that both implementations compute more or less the same budget allocations. To verify this, the average difference between the budget allocations of the centralized and the distributed implementation is computed for each considered target over the whole simulation. This is done in another simulation scenario in which 24 randomly placed targets with constant velocities, and two static sensors are considered, similar to the setup in Figure 6.2. The results are shown in Figure 6.6.

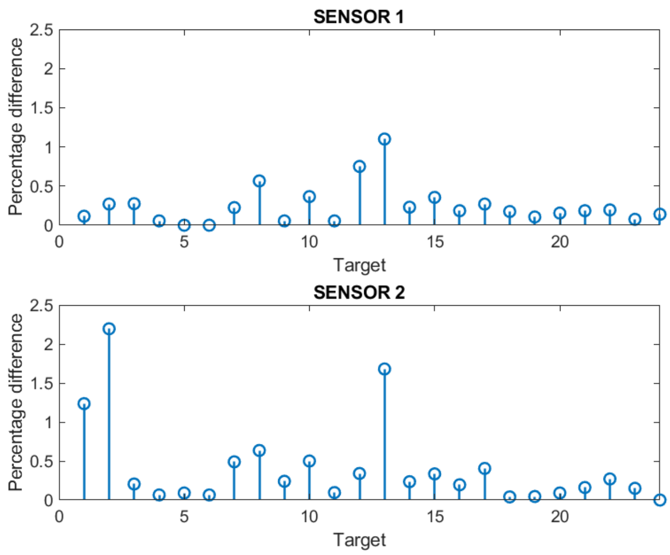


Figure 6.6: Percentage difference in dwell times for 24 randomly placed targets between the centralized and distributed implementation in a dynamic scenario.

Since the targets are displacing between the budget allocation updates, the last known dwell times of the other sensors used in the budget optimizations are the values of the previous resource allocation. Thus, it is expected to see a difference between the resource allocations of the centralized and distributed solution. However, the maximum difference between both implementations is below the defined LR solution precision of 5%. Therefore, the results of the centralized and the distributed implementation are approximately equal and converge to similar results over time, if the communication intervals of the sensors are chosen small enough w.r.t. the movements of the targets.

## 6.4. AODB for Multi-Sensor Automotive Applications

The approach covered in this chapter until this point was not designed for any specific use case. This chapter considers an automotive application to demonstrate how the proposed framework and algorithm can be used in practice. Especially with the emerging of self-driving cars, the sensors used in cars are becoming more and more important. As Figure 6.7 shows, not only radar but also lidar and cameras, as well as the Global Positioning System (GPS), play a role for self-driving cars. Therefore, the management of those sensors is becoming increasingly important. Some additional assumptions and definitions about this scenario are covered in the following.

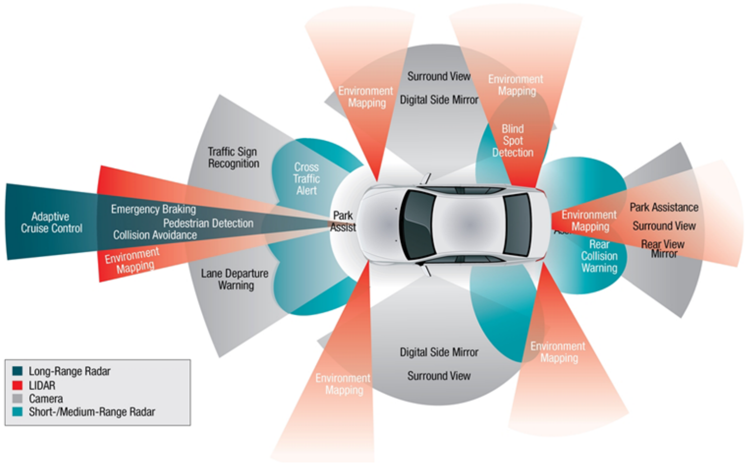


Figure 6.7: Depiction of the sensors in a self-driving car [106].

### 6.4.1. Automotive Radar Systems

Most MFR systems are pulse radar systems, which means that they transmit a pulsed radio signal. Using rotatable phased array antennas and DBF, they are capable of transmitting these pulses into different directions using a potentially narrow beam. The steering direction can be changed rapidly and therefore allows to focus on multiple individual targets almost simultaneously. Contrarily, standard automotive radar systems use Frequency-Modulated Continuous-Wave (FMCW) signals. While pulse radars need to turn off the transmitter to wait for the echo of the transmitted waveform to arrive, FMCW radars are constantly transmitting chirp signals. In addition to that, most automotive radars do not have very good beamforming capabilities and transmit a wide beam that illuminates a big part of the environment instead of focusing on specific single targets. A more detailed introduction to FMCW can be found, for instance, in [107].

The following basic assumptions are made about the radar system:

- The radar transmits signals in frames of a specific constant length. Each frame consists of a certain amount of slots, each consisting of an FMCW chirp. An example frame is shown in Figure 6.8.

- Based on the frame definition, the revisit time $T$ is constant and defined as the length of one frame. Additionally, the actions for the dwell time $\tau$ are directly defined by the chirp slots. This means that the dwell time actions are by definition already discretized, and the shortest possible dwell time is the length of one chirp.

- The radar is assumed to illuminate the environment using a wide beam. In the following, a car is considered to be a radar node that consists of multiple radar sensors that are able to illuminate all 360° around the car. It is assumed that the radar systems are not controlled separately by the algorithm. As it is not possible to focus on single objects, increasing the dwell time for improving the sensing accuracy for a single object will automatically also increase the sensing accuracies for the other objects. Therefore, each sensor is assumed to execute a single action for all targets.

- For a single car, the best action would always be to use the full frame for transmission. Therefore, no RRM would be necessary in that case. However, this changes when multiple cars are close to each other in the same environment. Therefore, it is assumed here that there is communication between the radar nodes, and the actions of each radar node depend on the others.

- For simplicity, it is assumed that the cars that are part of the optimization procedure are not detecting each other as a target. Instead, it is assumed that the cars observe other objects outside of their cooperative system and optimize their sensor resources w.r.t. these external targets.

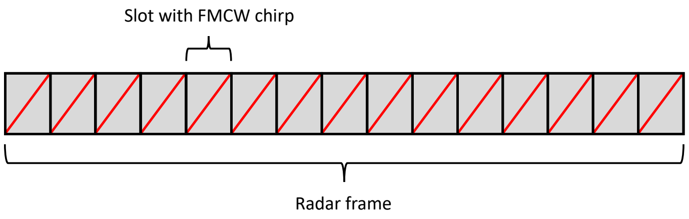


Figure 6.8: Depiction of a radar frame for an automotive scenario.

### 6.4.2. Radar-to-Radar Interference

Interference in automotive radar is a common topic in current research, as it can introduce extra noise in the signal or even produce so-called ghost targets. There are a few reasons, why interference is considered a big challenge in automotive applications. Firstly, the frequency spectrum of automotive radars is limited by regulations. Therefore, all car radars operate in similar frequency bands. Secondly, automotive radars are practically always transmitting and there is no silence as there is in between the signals of pulse radars. Therefore, interference is almost inevitable if no actions are taken against it. Thirdly, every modern car has multiple radar systems installed, which means that on crowded streets hundreds of radar systems are interfering with each other.

Much research has covered interference avoidance or mitigation techniques, some of which assume cooperation between the vehicles. This can already be seen as a basic RRM, as it adjusts the sensing parameters to the current situation. Some approaches that are dealing with this topic can be found, e.g., in [108–111].

For simplicity, in the following, it is assumed that interference is unacceptable and would make accurate sensing impossible. Therefore, only one radar node can sense at the same time, while the others are silent.

### **6.4.3.** Joint Sensing and Communication

For the cooperation of cars, some means of communication is necessary. This could be extra systems, such as 5G or Wi-Fi, for instance. Another possibility is to use the radar sensor for both sensing and communication tasks. Such an approach can be implemented through separate waveforms for the different tasks or by embedding the communication signal into the sensing waveform. It has been suggested that such an approach can reduce the costs and space needed for the devices. In addition to that, it would help to free up the frequency spectrum, as sensing and communication would use the same frequency band [112]. Many specific communication-related topics have been covered by recent research, e.g., how to synchronize different automotive radars for communication [113].

In the following, it is assumed that all information is communicated perfectly between the radar nodes.

### **6.4.4.** Time and Frequency Optimization Approach

The approach and most of the assumptions are the same as in the previous parts of this chapter. There are three main differences. Firstly, the radar sensors are moving through the environment. As the radar measurements only depend on the relative distance of sensors and targets, the same equations as in Section 6.3.4 can be used. Secondly, it is assumed that only one radar system is allowed to transmit at the same time. All the others need to be silent during that period and can only listen. And thirdly, it is assumed that the radar can choose from a set of different predefined frequency bands that all deliver the same sensing performance. This will allow multiple radar systems to operate simultaneously while not interfering with each other. The algorithm does not assign a specific frequency band to the sensors but instead allocates a relative budget. Based on this, a frequency can be chosen, which is not part of the simulations in this chapter. Another assumption is that each sensor can only operate in a single frequency band at a time.

It is assumed that a central processor exists which fuses the measurements for tracking and performs the RRM optimization. For this section, the communication is assumed to be performed through an external system. Additionally, it is assumed that the information is transmitted instantly. Based on those assumptions, the problem can be written as

$$\begin{aligned} \underset{\tau}{\text{minimize}} \quad & \sum_{n=1}^{N} E\left[c(\boldsymbol{A}_k, \boldsymbol{s}_k^n)\right] \\ \text{subject to} \quad & \sum_{m=1}^{M} \boldsymbol{a}_k^m \leq \boldsymbol{b}_{max} \cdot f_B \\ & \boldsymbol{a}_k^m \leq \boldsymbol{b}_{max} \; \forall \; m \in \{1, 2, \cdots, M\}, \end{aligned} \tag{6.17}$$

where $M$ is the number of sensors, $\boldsymbol{b}_{max}$ is the maximum budgets for all sensor nodes, $f_B$ is the number of available frequency bands and $\boldsymbol{A}_k = [\boldsymbol{a}_k^1, \cdots, \boldsymbol{a}_k^M]$ are the sensor actions that each sensor executes. The first constraint takes care that the maximum budget is not exceeded, while the second constraint makes sure that a single sensor cannot occupy multiple frequency bands at once. As the sensors are observing all targets at the same time, the actions do not depend on the target.

### 6.4.5. Communication Selection Optimization Approach

The problem formulation slightly changes when it is assumed that communication is done through the radar sensor. In the presented case, this assumption means that communication is considered an extra task that uses a part of the available sensing resources. In the following, it is assumed that communicating the measurement information for a certain target $n$ takes a certain transmission and reception time, denoted as $t_I$. Furthermore, it is assumed that measurements from some sensors might lead to a more significant improvement of the overall target information in the central processor than the measurements from other sensors. Therefore, the resource allocation for communication can be reduced if only the measurements leading to the biggest cost improvement are transmitted.

In order to do that, a measurement selection matrix $\boldsymbol{U} \in \mathbb{R}^{M \times N}$ is introduced. It is a matrix consisting of binary values, one for each target and sensor and can be written as

$$\boldsymbol{U} = \begin{bmatrix} U_{1,1} & U_{1,2} & \cdots & U_{1,N} \\ U_{2,1} & U_{2,2} & \cdots & U_{2,N} \\ \vdots & \vdots & \ddots & \cdots \\ U_{M,1} & U_{M,2} & \cdots & U_{M,N} \end{bmatrix}. \tag{6.18}$$

The optimization problem from (6.17) then changes to

$$\begin{aligned} \underset{\tau}{\text{minimize}} \quad & \sum_{n=1}^{N} E\left[c(\boldsymbol{A}_k, \boldsymbol{s}_k^n)\right] \\ \text{subject to} \quad & \sum_{m=1}^{M} \boldsymbol{a}_k^m + \mathbf{1}_M^T \boldsymbol{U} \mathbf{1}_N \cdot t_I \leq \boldsymbol{b}_{max} \cdot f_B \\ & \boldsymbol{a}_k^m \leq \boldsymbol{b}_{max} \; \forall \; m \in \{1, 2, \cdots, M\}, \end{aligned} \tag{6.19}$$

where $\mathbf{1}_M \in \mathbb{R}^M$ and $\mathbf{1}_M \in \mathbb{R}^N$ are vectors of all ones of length $M$ and $N$, respectively. Using this formulation, the RRM algorithm calculates a resource allocation for the sensing tasks, while deciding which measurement data from which sensor is going to be transmitted to the central node based on the expected influence on the cost.

### 6.4.6. Simulation A: Time and Frequency Allocation

A simple tracking example is considered in the following to show the validity of the approach in an automotive application. Both sensing time and frequency are optimized, and an EKF is applied for tracking the objects. The cost function is the same as in (6.15). In total, two separate frequency bands with identical sensing performance are considered. The communication between the sensors and the central node is assumed to be perfect and instantaneous. The optimization problem is considered as explained in Section 6.4.4. The simulation results are based on the algorithm as shown in [103] and [105].

The considered scenario consists of 2 targets without sensors that are moving through the scene. In addition to that, 3 radar nodes (e.g., cars, in the following referred to as sensors) are considered, of which one is static, and the other two are moving. The scenario is shown in Figure 6.9 and takes a total of 30 s. It can be considered an urban three-way intersection. While Target 1 is taking a right turn starting at 26 s, Target 2 is accelerating after 42 seconds and reaches a final velocity of $16\,\mathrm{m\,s^{-1}}$ or almost $60\,\mathrm{km\,h^{-1}}$. Sensors 2 and 3 are moving with a constant velocity. For reasons of simplification and a better overview in the figures, it is assumed that the radar nodes are not targets themselves, which means that the other radars will not sense them. The general simulation parameters are shown in Table 6.4 and the initial target and sensor parameters are shown in Table 6.5.

Figure 6.10 shows the resulting frequency allocations. It can be seen that most of the resources are first assigned to Sensors 1 and 3, as they are placed in the middle of the scenario and can observe both targets reasonably well. After about 24 s, the budget of Sensor 3 is reduced while the budget of Sensor 2 is increased. The reason is that Sensor 3 is moving away from the targets, while Sensor 2 is moving to a position where it can observe both targets more accurately. All sensor resource allocations are below the maximum possible sensor budget per frequency band. Therefore, no sensor needs to operate in two frequency bands simultaneously. However, it should be noted that at least one of the sensors needs to operate in both frequency bands, although not necessarily simultaneously. This might not be possible in a practical application. Thus, a more sophisticated approach might need to be chosen for the explicit scheduling, or an additional constraint needs to be added to the optimization problem.

### 6.4.7. Simulation B: Time and Frequency Allocation with Communication Selection.

This section considers the same simulation scenario as discussed in Simulation A, see Figure 6.9. Therefore, Tables 6.4 and 6.5 are valid for this simulation as well. The only difference is that communication is considered to consume a part of the sensor resources. Therefore, the sensor needs to evaluate which information improves the knowledge about the environment and decide if it is worth transmitting it to the central processing node. The sensing time required for transmitting measurement information of a single target is assumed to be $t_I = 0.05\,\mathrm{s}$. The problem formulation follows the definition in Section 6.4.5. The simulation results are based on the algorithm as shown in [103] and [105].

The simulation results can be found in Figures 6.11 and 6.12. It can be seen that the

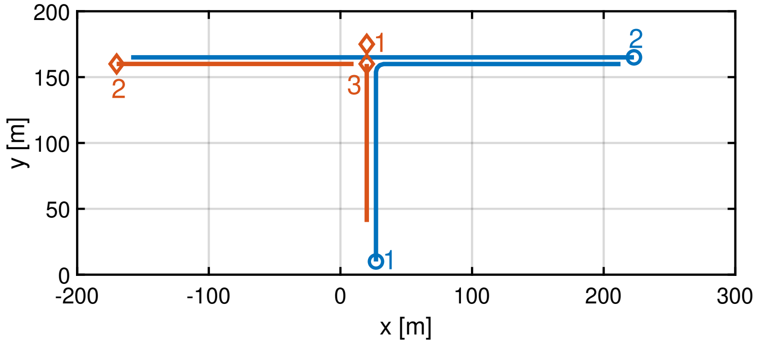


Figure 6.9: Trajectories of the targets (circles) and sensors (diamonds) for simulation scenarios A and B. The circles and diamonds mark the initial starting positions.

Table 6.4: Simulation parameters for simulation scenarios A and B.

| **Parameter** | **Value** |
|---|---|
| System range noise variance ($\sigma^2_{r,0}$) | 0.01 m$^2$ |
| System angle noise variance ($\sigma^2_{\theta,0}$) | $6.4\times10^{-7}$ rad$^2$ |
| Reference SNR (SNR$_0$) | 1 |
| Reference RCS ($\varsigma_0$) | 10 m$^2$ |
| Reference dwell time ($\tau_0$) | 1 s |
| Reference range ($r_0$) | 50 m |
| Revisit interval ($T$) | 1 s |
| Action space discretization steps ($\Delta\tau$) | 7.9 ms |
| Action space limits dwell time ($\tau_{min}, \tau_{max}$) | $\tau \in$ [7.9 ms ... 1 s] |
| Number of rollouts ($M$) | 4 |
| Rollout horizon ($\mathcal{H}$) | 4 |
| Base Policy ($\pi_{base}$) | $\boldsymbol{a}$ |
| Maximum available budget ($\Theta_{max}$) | 2 |
| Budget update interval ($t_B$) | 1 s |
| Beam positioning error ($\Delta\alpha$) | 0 |
| Probability of detection ($P_D$) | 1 |

Table 6.5: Initial target and sensor parameters for simulation scenarios A and B.

| **Parameter** | **Target 1** | **Target 2** | **Sensor 1** | **Sensor 2** | **Sensor 3** |
|---|---|---|---|---|---|
| $x_0^n$ [m] | 27 | 223 | 20 | -170 | 20 |
| $y_0^n$ [m] | 10 | 165 | 175 | 161 | 161 |
| $\dot{x}_0^n$ [m s$^{-1}$] | 0 | -12 | 0 | 6 | 0 |
| $\dot{y}_0^n$ [m s$^{-1}$] | 12 | 0 | 0 | 0 | -4 |
| $\varsigma^n$ [m$^2$] | 20 | 20 | - | - | - |
| $\sigma_w^n$ [m$^2$ s$^{-4}$] | 20 | 20 | - | - | - |

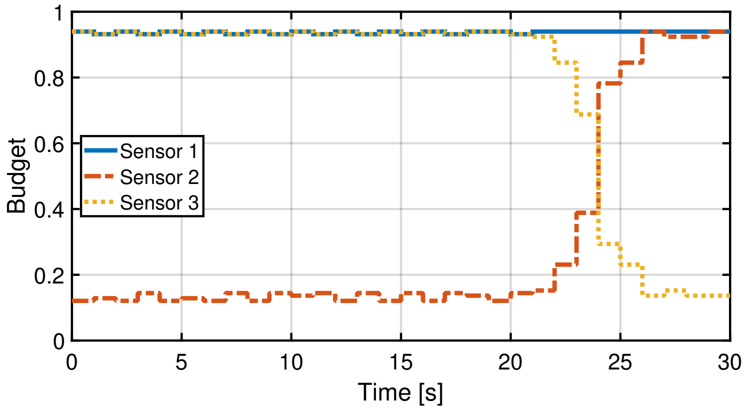


Figure 6.10: Resource allocations to the three sensors for simulation scenario A. As two frequency bands are considered, the maximum available budget is 2.

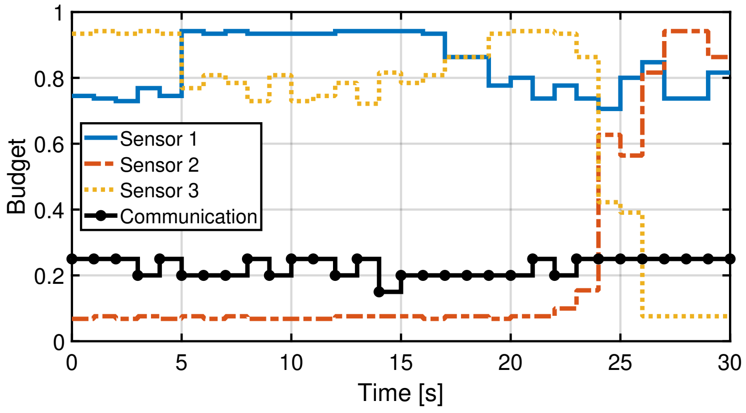


Figure 6.11: Resource allocations to the three sensors for simulation scenario B. As two frequency bands are considered, the maximum budget is 2. The black line denotes the time allocated to communication for all sensors (both for transmitting and receiving).

resource allocations to the three sensors are similar to the results from Simulation A. The differences in the budgets reflect the fact that sensing and communication tasks are competing for resources. The algorithm allocates the resources based on their impact on the cost. In Figure 6.12 it can be seen that Sensor 3 transmits its measurements of both targets at the beginning of the simulation until 13 s. Afterward, it is only transmitting the measurements of Target 1. The reason is that it is first placed in the middle of the intersection with a relatively short distance to both targets. As it moves away from the intersection, its measurements are getting more and more inaccurate, especially for Target 2. For Sensor 2, the opposite is happening as it is moving towards the intersection. It can be observed that the measurements of Sensor 2 are so bad in the beginning that none of them are being transmitted to the central processing node (for instance at 3 s and 5 s). As the stationary Sensor 1 is placed centrally at the intersection, it has a relatively short distance to both targets during the whole scenario and most of the time transmits both targets' measurement information.

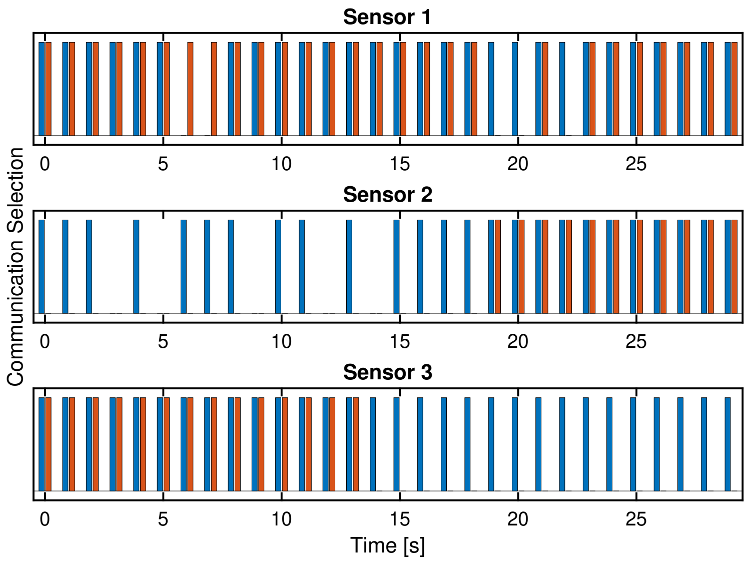


Figure 6.12: Communication selection of the three sensors for simulation scenario B. The blue and red bars denote that the information of Target 1 and Target 2, respectively, is transmitted to the central processing node from the specified sensor.

## 6.5. Conclusions

This chapter covered the extension of the previously suggested RRM framework to multi-sensor scenarios. Its applicability was shown in two different problem settings.

In a theoretical multi-sensor multi-target tracking scenario, it has been shown how the RRM problem can be solved using the framework mentioned in Chapter 3. The strength of this approach is the practical formulation as a distributed algorithm. It has been shown that both the approximately optimal and the distributed solution lead to a lower cost than doing individual resource optimization per sensor. In addition to that, the distributed solution reduces the computational complexity of the optimal solution significantly.

An automotive scenario has shown how the RRM problem can be solved by taking communication between cars into account. The algorithm, which is again based on the framework from Chapter 3, allocates the available sensor time, as well as available frequency bands to the different cars based on the uncertainty in the target tracks. The constraint takes care that no mutual radar interference between the cars takes place. In addition to that, the algorithm decides which car has relevant information that needs to be shared with the other cars. By doing that, electromagnetic spectrum usage can be reduced.

# 7 Conclusions

## **7.1.** Major Results and Novelties

The major results of this PhD research are discussed in the following subsections.

- *Optimal Multi-Task Resource Balancing (Chapter 2)*

  Many solutions have been proposed previously that claim to solve RRM problems focus on single task optimization. However, the problem becomes more complicated to solve when multiple tasks have to be considered, and the MFR system is operating at its resource limit. Allocating more resources for one task will then inevitably reduce them for the other tasks, leading to a deteriorated performance. **In this thesis, it has been shown for the first time that a steady-state solution for such a multi-task resource balancing problem for an LTI tracking problem can be obtained by applying LR using the proposed OSB algorithm.** The presented LTI problem can be seen as a special case of a general POMDP problem formulation. The resources are successfully allocated according to a chosen cost function, and optimality w.r.t. the cost function is reached with any desired precision. **Due to the optimality of this approach, the results are better than general heuristic RRM techniques.** In Chapter 2, this has been illustrated for a one-dimensional multi-target tracking scenario while using the optimal steady-state solution of the KF as the cost function. It is the first time that this kind of analysis has been performed. The proposed algorithm can also be used to solve dynamic problems in a myopic fashion.

- *A generic framework and algorithmic solution (Chapters 3 and 4)*

  Most available approaches have been specifically designed with a specific application scenario and sensor type in mind. **In this thesis, a novel generic framework and algorithmic solution for RRM in multi-target tracking have been introduced and demonstrated in practical scenarios.** Assuming an underlying POMDP, it has been shown that the proposed AODB algorithm using a combination of LR and PR can be applied for a non-myopic RRM solution in dynamic multi-task scenarios. **Additionally, it has been shown that this generic solution is applicable to a variety of sensors and leads to more optimal results than heuristic methods w.r.t. the chosen cost function.** In Chapters 3 and 4 it was demonstrated how the algorithm performs in a multi-target tracking scenario. **The novel framework has been further investigated through a detailed analysis of the performance and computational efficiency.** These practical results are very valuable since there is not much literature available for multi-task RRM solutions assuming POMDPs, and the ones that exist usually stay on a very high theoretical level without comparison to other techniques. **Since the PR was found to have a high computational load, MPC has been investigated as an alternative POMDP solution method and was found to be a time-saving but still accurate alternative.**

- *RRM for joint tracking and classification (Chapter 5)*

  Many approaches tackle the RRM problem for single types of tasks, such as searching, classification, or tracking. In practice, an MFR system would be able to perform multiple different task types in parallel. To make use of this versatility, earlier proposed solutions have been assuming, e.g., separate cost functions for the different task types or by introducing extra heuristics. **The proposed novel approach shows that it is possible to solve the RRM problem for multiple task types by using only a single cost function based on a definition of mission threat. Such an approach has been suggested previously but has never been fully developed and demonstrated with the help of practical simulation scenarios.** In Chapter 5 this novel approach is illustrated in a joint tracking and classification scenario for multiple targets. It is pointed out how the character of the sensing task changes once the corresponding target is classified. This leads to different budget allocations not only based on the tracking accuracy but also on the progress of the classification.

- *RRM for sensor networks (Chapter 6)*

  Due to the increase of cheaply available computational power and the growing interest in connecting multiple sensors, RRM will become increasingly crucial for sensor networks. The vast majority of RRM research focuses on single sensors, while there are many approaches for solving sensor selection problems. **In this thesis, novel approaches of RRM for sensor networks have been presented, and their usefulness has been demonstrated in general tracking scenarios, as well as in automotive scenarios.** The approach is illustrated in Chapter 6. The proposed approach is novel, as RRM approaches for sensor networks that assume a balancing of limited resources only existed on a conceptual level. This thesis demonstrates how such an algorithm can be implemented in practice.

## 7.2. Recommendations for Future Work

Due to the challenging problem setting, not all interesting aspects of this research topic could be explored in detail. The following recommendations are intended as possible starting points for further research:

- *Investigation of different cost functions.* Within this thesis, the focus was on solution methods for RRM, assuming that an informative cost function exists. In the presented examples, cost functions have been applied that are easy to understand and whose results are easy to compare. However, in real scenarios, appropriate cost functions heavily depend on the point of view of the sensor user or operator and the specific situation at hand. Depending on what goal the sensing mission is supposed to achieve, the cost function can therefore strongly vary. Therefore, it is recommended to have a closer look at possible cost functions and take different scenarios into account.

- *Investigation of base policies for the PR.* In this thesis, the BP of the PR is assumed to execute the same action again for all future time steps. When some previous knowledge is available, this straightforward BP could be replaced by another one leading to better performance, e.g., by using a sequence of different actions instead of simply repeating the same one. It would be very interesting to explore possible base policies and their impact on the mission. A possible method to investigate is Parallel Rollout, which evaluates the cost using multiple BPs.

- *Investigation of the impact of different horizon lengths.* Generally, one would assume that a longer horizon should lead to a lower cost in the long term. In POMDP theory, this is indeed true, but for practical implementations and scenarios, as shown in this thesis, this is much more difficult to illustrate. Therefore, it would be insightful to see how different horizon lengths for the POMDP solution influence the algorithm's performance when applied in a very uncertain dynamic environment.

- *Find optimal parameters for algorithm.* This thesis demonstrates that an algorithm applying LR and PR delivers useful results. However, the analysis also showed that it takes a very long time to converge. Therefore, for a practical implementation, it is recommended to investigate aspects such as an adaptive action space for the PR, an adaptive step size for the LR, and other improvements increasing the computational efficiency (e.g., parallel computing).
- *Including of search functionality.* The algorithm presented in this thesis was applied to a joint tracking and classification scenario. The underlying ideas should also be applicable to include searching functionality. For a genuinely generic RRM solution, this aspect would need to be added to the solution.

# A

# Lagrangian Relaxation for RRM

By using LR, one can decouple big constrained optimization problems into smaller ones that can be solved independently of each other. This section introduces how this technique can be used in RRM.

## A.1. Lagrangian relaxation principle

LR is an approach to simplify a complicated constrained optimization problem. In this process, constraints can be removed by adding them as penalty terms into the original problem, or PP, in combination with so-called Lagrange multipliers. As a consequence, a new optimization problem is created that has less constraints than the PP. The optimization procedure consists of maximizing the minimum of the cost function by adjusting the Lagrange multipliers. This is called the LDP which is usually a lower estimate of the PP if the initial Lagrange multipliers are chosen correctly (see for example [114]).

LR and Lagrange multipliers have been extensively covered in literature (for example in [28, 31, 34, 114, 115]). As an example of how LR is applied, the general optimization problem with $N$ input variables is considered that is shown in (A.1).

$$\begin{aligned} \underset{\boldsymbol{x}}{\text{minimize}} \quad & f(\boldsymbol{x}) \\ \text{subject to} \quad & \boldsymbol{g}(\boldsymbol{x}) \leq \boldsymbol{A} \\ & \boldsymbol{h}(\boldsymbol{x}) \geq \boldsymbol{B}, \end{aligned} \tag{A.1}$$

where

$$\begin{aligned}\boldsymbol{x} &= [x_1, ..., x_N]^T \in \mathbb{R}^N,\\ \boldsymbol{g}(\boldsymbol{x}) &= [g_1(\boldsymbol{x}), ..., g_m(\boldsymbol{x})]^T \in \mathbb{R}^m,\\ \boldsymbol{h}(\boldsymbol{x}) &= [h_1(\boldsymbol{x}), ..., h_p(\boldsymbol{x})]^T \in \mathbb{R}^p,\\ \boldsymbol{A} &= [A_1, ..., A_m]^T \in \mathbb{R}^m,\\ \boldsymbol{B} &= [B_1, ..., B_p]^T \in \mathbb{R}^p.\end{aligned}$$

As mentioned above, the idea is to include the constraints into the primal optimization problem. This is done by adding a penalty term for each removed constraint, multiplied by Lagrange multipliers, which are defined as $\boldsymbol{\lambda} = [\lambda_1, ..., \lambda_m]^T \in \mathbb{R}^m$ and $\boldsymbol{\mu} = [\mu_1, ..., \mu_N]^T \in \mathbb{R}^p$. The Lagrangian is defined as

$$L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = f(\boldsymbol{x}) + \sum_{i=1}^{m} \lambda_i (g_i(\boldsymbol{x}) - A_i) + \sum_{j=1}^{p} \mu_j (B_j - h_j(\boldsymbol{x})). \tag{A.2}$$

The relaxed problem is called LD (LD) function and is defined as

$$d(\boldsymbol{\lambda}, \boldsymbol{\mu}) = \underset{\boldsymbol{x}}{\text{minimize}}\, L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}). \tag{A.3}$$

The LDP is then characterized as finding the maximum of the LD function with respect to the Lagrange multipliers, as shown in (A.4).

$$Z_D = \underset{\boldsymbol{\lambda}, \boldsymbol{\mu}}{\text{maximize}}\, d(\boldsymbol{\lambda}, \boldsymbol{\mu}). \tag{A.4}$$

To summarize, the objective function is minimized over $\boldsymbol{x}$, while also being maximized over the Lagrange multipliers, in order to come as close to the PP as possible. To find the optimal Lagrange multipliers and therefore the tightest lower bound to the PP, iterative approaches can be used. There are many techniques available to calculate the Lagrange multipliers iteratively, like the commonly used subgradient method. It will be explained in the next subsection.

## A.2. Duality

Generally, applying LR does not lead to the optimal solution of the PP. In the general case, *weak duality* holds, which means that the optimal solution of the PP ($p^\star$) and the optimal solution of the LDP ($d^\star$) are not equal:

$$d^\star \leq p^\star. \tag{A.5}$$

In this case, the solution of the LDP provides a lower bound to the solution of the PP. The difference between the solutions $p^\star - d^\star$ is called the duality gap.

In some cases, *strong duality* holds. This means that the duality gap is zero, and the solution of the LDP provides the same solution as the PP:

$$d^\star = p^\star. \tag{A.6}$$

Strong duality usually holds when the primal function is convex. A more detailed discussion about sufficient conditions for strong duality can be found, e.g., in [31].

## A.3. Subgradient method

The subgradient method (see for example [114]) is an iterative process that starts with a chosen initial value for the Lagrange multipliers (e.g. 1). At each iteration $l$, first the minimum of the relaxed problem is calculated (LD function, see (A.2)). Next, the subgradients are chosen for each constraint as $\boldsymbol{e}_\lambda^l = [e_{\lambda,1}^l, ..., e_{\lambda,m}^l]^T \in \mathbb{R}^m$ and $\boldsymbol{e}_\mu^l = [e_{\mu,1}^l, ..., e_{\mu,p}^l]^T \in \mathbb{R}^p$. Following the notation that has been used above for the constraints, the subgradients are defined as

$$\begin{aligned} \boldsymbol{e}_\lambda^l &= (\boldsymbol{g}(\boldsymbol{x}^l) - \boldsymbol{A}) \\ \boldsymbol{e}_\mu^l &= (\boldsymbol{B} - \boldsymbol{h}(\boldsymbol{x}^l)). \end{aligned} \tag{A.7}$$

The Lagrange multipliers are then updated with a specific step size $\gamma^l$. For inequality constraints, the penalty terms are not allowed to become negative. Therefore the new Lagrange multipliers for the next iteration are calculated as shown in (A.8):

$$\begin{aligned} \boldsymbol{\lambda}^{l+1} &= max\{0, \boldsymbol{\lambda}^l + \gamma^l \boldsymbol{e}_\lambda^l\} \\ \boldsymbol{\mu}^{l+1} &= max\{0, \boldsymbol{\mu}^l + \gamma^l \boldsymbol{e}_\mu^l\}. \end{aligned} \tag{A.8}$$

The step size can be chosen freely. A possibility are constant or decreasing step sizes like $\gamma_0/l$ or $1/\gamma^l$, for example. The process is then started again with the new Lagrange multipliers. A new LD function is found and afterwards, new subgradients are calculated again. Theoretically, the exact result has been found when the gradients $\boldsymbol{e}_\lambda^l$ and $\boldsymbol{e}_\mu^l$ reach 0. Since this value will never be reached exactly, the process is repeated until the gradient reaches 0 with a desired precision.

To summarize, a short overview of the subgradient algorithm for the above mentioned optimization problem is given here:

1. $l = 0$: Set the Lagrangian multipliers to initial value ($\boldsymbol{\lambda}^0 = \boldsymbol{\lambda}_0, \boldsymbol{\mu}^0 = \boldsymbol{\mu}_0$).
2. Calculate solution for $d(\boldsymbol{\lambda}, \boldsymbol{\mu})$ and save $\boldsymbol{x}^l$.
3. Choose subgradients for Lagrangian multipliers $\boldsymbol{e}_\lambda^l$ and $\boldsymbol{e}_\mu^l$ (see (A.7)).
4. Check if $\boldsymbol{e}_\lambda^l \approx 0$ and $\boldsymbol{e}_\mu^l \approx 0$ with desired precision. If it is, stop the process.
5. Adjust Lagrangian multipliers as shown in (A.8).
6. Go to step 2 and set $l = l + 1$.

# B

# Golden Section Search

Golden Section Search (GSS) is a technique which is used to find an extremum of a function. In this thesis, it is used as an alternative to the subgradient method. This explanation is based on [77, 78]. In the implementation for this thesis, the goal is to find the Lagrange multiplier $\lambda$ such that

$$\left|f(\lambda)\right| \leq \varepsilon, \tag{B.1}$$

where

$$f(\lambda) = \sum_{n=1}^{N} \boldsymbol{a}_n(\lambda) - B_{max} \tag{B.2}$$

and $\varepsilon$ is indicating the tolerance of the constraint. The optimized action $\boldsymbol{a}_n$ directly follows from the chosen $\lambda$ in the LR optimization procedure, see explanation of LR in Appendix A. In the standard GSS method, as described in for example [116], is extended due to the fact that initially the lower and upper bounds of $\lambda$ are unknown. To find these bounds, $\lambda$ is increased with a certain step size after every timestep $l$. As $\lambda$ needs to make sure that (B.1) is met, the initial values for the upper and lower bounds are chosen as $\lambda_l$ and $\lambda_{l+1}$ when there is a sign change in between of $f(\lambda_l)$ and $f(\lambda_{l+1})$. These lower and upper bounds will be referred to as $x_L$ and $x_U$. Once these bounds are found, $f(\lambda)$ is evaluated between $x_U$ and $x_L$. It has been shown that an efficient way of choosing these intermediate points is to use the Golden Ratio conjugate ($r = \frac{\sqrt{5}-1}{2} \approx 0.618$). First, 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. If $f(x_1)$ is smaller or equal $f(x_2)$, then $x_U$ is shifted to $x_1$, otherwise $x_L$ or is shifted to $x_2$. Subsequently, the function is evaluated at the newly chosen point. This procedure is repeated until (B.1) is met. 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 $\lambda$ and compute f($\lambda$) until f($\lambda$) becomes
  negative;
Step 2 This $\lambda$ becomes $x_U$ while the previous $\lambda$ 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
```

# Bibliography

[1] L. Brown, *Technical and Military Imperatives: A Radar History of World War 2*. Institute of Physics Publishing, 1999.

[2] B. T. Neale, "CH - The First Operational Radar," *The GEC Journal of Research*, vol. 3 (2), pp. 73–83, 1985.

[3] P. A. Lynn, *Radar Systems*. Boston, MA: Springer US, 1987.

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

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

[6] A. Charlish, F. Hoffmann, and I. Schlangen, "The Development from Adaptive to Cognitive Radar Resource Management," *IEEE Aerospace and Electronic Systems Magazine*, vol. 35, 6 2020.

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

[8] 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.

[9] 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.

[10] 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.

[11] National Severe Storms Laboratory, "NWRT: End of an Era." https://www.nssl.noaa.gov/about/history/nwrt-decommission/. Accessed 2021-06-17.

[12] A. Steinberg and C. Bowman, "Rethinking the JDL Data Fusion Levels," in *Proceedings of the National Symposium on Sensor Data Fusion, JHUAPL*, 2004.

[13] F. Smits, A. Huizing, W. van Rossum, and P. Hiemstra, "A Cognitive Radar Network: Architecture and Application to Multiplatform Radar Management," in *2008 European Radar Conference*, pp. 312–315, Oct 2008.

[14] A. F. Martone and A. Charlish, "Cognitive radar for waveform diversity utilization," in *2021 IEEE Radar Conference (RadarConf21)*, pp. 1–6, May 2021.

[15] F. P. B. Osinga, *Science Strategy and War: The Strategic Theory of John Boyd.* Eburon Academic Publishers, 2005.

[16] F. Katsilieris, *Sensor management for surveillance and tracking: An operational perspective.* Phd thesis, TU Delft, Delft, The Netherlands, 2015.

[17] H. S. Mir and A. Guitouni, "Variable Dwell Time Task Scheduling for Multifunction Radar," *IEEE Transactions on Automation Science and Engineering*, vol. 11, pp. 463–472, April 2014.

[18] M. Shaghaghi and R. S. Adve, "Task Selection and Scheduling in Multifunction Multichannel Radars," in *2017 IEEE Radar Conference (RadarConf)*, pp. 0969–0974, May 2017.

[19] C. Kreucher, A. O. Hero, K. Kastella, and D. Chang, "Efficient Methods of Nonmyopic Sensor Management for Multitarget Tracking," in *2004 43rd IEEE Conference on Decision and Control (CDC) (IEEE Cat. No.04CH37601)*, vol. 1, pp. 722–727 Vol.1, Dec 2004.

[20] K. White, J. Williams, and P. Hoffensetz, "Radar Sensor Management for Detection and Tracking," in *2008 11th International Conference on Information Fusion*, pp. 1–8, June 2008.

[21] R. A. Romero and N. A. Goodman, "Cognitive Radar Network: Cooperative Adaptive Beamsteering for Integrated Search-and-Track Application," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 49, pp. 915–931, 4 2013.

[22] M. Byrne, K. White, and J. Williams, "Scheduling Multifunction Radar for Search and Tracking," in *2015 18th International Conference on Information Fusion (Fusion)*, pp. 945–952, July 2015.

[23] J. Yan, W. Pu, D. Jinhui, H. Liu, and Z. Bao, "Resource Allocation for Search and Track Application in Phased Array Radar based on Pareto Bi-objective Optimization," *IEEE Transactions on Vehicular Technology*, vol. 68, pp. 3487–3499, 2019.

[24] S. Martin, "Risk-Based Sensor Resource Management for Field of View Constrained Sensors," in *2015 18th International Conference on Information Fusion (Fusion)*, pp. 2041–2048, July 2015.

[25] M. E. Gomes-Borges, D. Maltese, P. Vanheeghe, and E. Duflos, "A Risk-Based Sensor Management Using Random Finite Sets and POMDP," in *2017 20th International Conference on Information Fusion (Fusion)*, pp. 1–9, July 2017.

[26] C. Kreucher and A. O. Hero, "Non-Myopic Approaches to Scheduling Agile Sensors for Multistage Detection, Tracking and Identification," in *Proceedings. (ICASSP '05). IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005.*, vol. 5, pp. v/885–v/888 Vol. 5, March 2005.

[27] A. O. Hero, D. A. Castanon, D. Cochran, and K. Kastella, *Foundations and Applications of Sensor Management*. New York, NY: Springer Publishing Company, Incorporated, 1st ed., 2008.

[28] D. P. Bertsekas, *Constrained Optimization and Lagrange Multiplier Methods*. Athena Scientific, 1st ed., 1996.

[29] D. P. Bertsekas, J. N. Tsitsiklis, and C. Wu, "Rollout Algorithms for Combinatorial Optimization," *Journal of Heuristics*, vol. 3, pp. 245–262, Dec. 1997.

[30] D. P. Bertsekas and D. A. Castanon, "Rollout algorithms for stochastic scheduling problems," *Journal of Heuristics*, vol. 5, pp. 89–108, Apr 1999.

[31] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 1st ed., 2004.

[32] D. P. Bertsekas, *Dynamic Programming and Optimal Control - Volume 2*, vol. 2. Athena Athena Scientific, 4 ed., 2012.

[33] D. P. Bertsekas, *Dynamic Programming and Optimal Control - Volume 1*, vol. 1. Athena Athena Scientific, 4 ed., 2017.

[34] F. L. Lewis, D. L. Vrabie, and V. L. Syrmos, *Optimal Control*. John Wiley & Sons, Inc., 3rd ed., 2012.

[35] J. Wintenby and V. Krishnamurthy, "Hierarchical Resource Management in Adaptive Airborne Surveillance Radars," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 42, pp. 401–420, April 2006.

[36] K. A. B. White and J. L. Williams, "Lagrangian relaxation approaches to closed loop scheduling of track updates," in *Signal and Data Processing of Small Targets*, pp. 8393–8393, 2012.

[37] 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.

[38] A. Irci, A. Saranli, and B. Baykal, "Study on Q-RAM and Feasible Directions Based Methods for Resource Management in Phased Array Radar Systems," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 46, pp. 1848–1864, Oct 2010.

[39] A. Charlish, K. Woodbridge, and H. Griffiths, "Phased Array Radar Resource Management Using Continuous Double Auction," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 51, pp. 2212–2224, July 2015.

[40] A. Charlish and F. Hoffmann, "Cognitive Radar Management," in *Novel Radar Techniques and Applications, Volume 2 - Waveform Diversity and Cognitive Radar, and Target Tracking and Data Fusion* (R. Klemm, H. Griffiths, and W. Koch, eds.), pp. 157–193, Scitech Publishing.

[41] F. Katsilieris, H. Driessen, and A. Yarovoy, “Threat-Based Sensor Management for Target Tracking,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 51, pp. 2772–2785, Oct 2015.

[42] C. Kreucher, K. Kastella, and A. O. Hero, “Sensor Management Using an Active Sensing Approach,” *Signal Processing*, vol. 85, pp. 607–624, Mar. 2005.

[43] M. I. Schöpe, H. Driessen, and A. Yarovoy, “Optimal Balancing of Multi-Function Radar Budget for Multi-Target Tracking Using Lagrangian Relaxation,” in *Proceedings of the 22nd International Conference on Information Fusion*, 2019.

[44] C. K. Chui and G. Chen, *Kalman Filtering with Real-Time Applications*. Springer-Verlag Berlin Heidelberg, 4th ed., 2009.

[45] P. R. Kalata, “The Tracking Index: A Generalized Parameter for alpha-beta and alpha-beta-gamma Target Trackers,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-20, pp. 174–182, March 1984.

[46] J. E. Gray and W. Murray, “A Derivation of an Analytic Expression for the Tracking Index for the Alpha-Beta-Gamma Filter,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 29, pp. 1064–1065, Jul 1993.

[47] F. Katsilieris, H. Driessen, and A. Yarovoy, “Threat-based sensor management for joint target tracking and classification,” in *2015 18th International Conference on Information Fusion (Fusion)*, pp. 435–442, July 2015.

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

[49] J. J. Schneider and S. Kirkpatrick, *Stochastic Optimization*. Berlin Heidelberg, Germany: Springer, 2006.

[50] A. Zhigljavsky and A. Zilinskas, *Stochastic Global Optimization*. Springer, 2007.

[51] A. Charlish, K. Bell, and C. Kreucher, “Implementing Perception-Action Cycles using Stochastic Optimization,” in *Proceedings of IEEE Radar Conference 2020*.

[52] Wikipedia, “Markov decision process.” https://en.wikipedia.org/wiki/Markov_ decision_process. Accessed 2021-06-17.

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

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

[55] E. Selvi, R. M. Buehrer, A. Martone, and K. Sherbondy, “Reinforcement Learning for Adaptable Bandwidth Tracking Radars,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 56, pp. 3904–3921, Oct 2020.

[56] C. E. Thornton, M. A. Kozy, R. M. Buehrer, A. F. Martone, and K. D. Sherbondy, “Deep Reinforcement Learning Control for Radar Detection and Tracking in Congested Spectral Environments,” *IEEE Transactions on Cognitive Communications and Networking*, vol. 6, pp. 1335–1349, Dec 2020.

[57] P. Pulkkinen, T. Aittomäki, A. Ström, and V. Koivunen, “Time Budget Management in Multifunction Radars Using Reinforcement Learning,” in *Proceedings of 2021 IEEE Radar Conference*, 2021.

[58] J. L. Williams, J. W. Fisher, and A. S. Willsky, “Approximate Dynamic Programming for Communication-Constrained Sensor Network Management,” *IEEE Transactions on Signal Processing*, vol. 55, pp. 4300–4311, Aug 2007.

[59] V. Krishnamurthy and D. V. Djonin, “Optimal Threshold Policies for Multivariate POMDPs in Radar Resource Management,” *IEEE Transactions on Signal Processing*, vol. 57, pp. 3954–3969, Oct 2009.

[60] C. Stachniss, G. Grisetti, and W. Burgard, “Information Gain-based Exploration Using Rao-Blackwellized Particle Filters,” in *Proceedings of Robotics: Science and Systems*, 2005.

[61] V. Krishnamurthy and D. V. Djonin, “Structured Threshold Policies for Dynamic Sensor Scheduling—A Partially Observed Markov Decision Process Approach,” *IEEE Transactions on Signal Processing*, vol. 55, pp. 4938–4957, Oct 2007.

[62] R. Martinez-Cantin, N. de Freitas, E. Brochu, J. Castellanos, and A. Doucet, “A bayesian exploration-exploitation approach for optimal online sensing and planning with a visually guided mobile robot,” *Autonomous Robots*, vol. 27, no. 2, pp. 93–103, 2009.

[63] M. T. J. Spaan, T. S. Veiga, and P. U. Lima, “Decision-theoretic planning under uncertainty with information rewards for active cooperative perception,” *Autonomous Agents and Multi-Agent Systems*, vol. 29, no. 6, pp. 1157–1185, 2015.

[64] M. I. Schöpe, H. Driessen, and A. Yarovoy, “Multi-Task Sensor Resource Balancing Using Lagrangian Relaxation and Policy Rollout,” in *Proceedings of the 23rd International Conference on Information Fusion*, 2020.

[65] S. Ross, J. Pineau, S. Paquet, and B. Chaib-draa, “Online Planning Algorithms for POMDPs,” *J. Artif. Int. Res.*, vol. 32, pp. 663–704, July 2008.

[66] 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.

[67] R. E. Bellman, *Dynamic Programming.* Princeton University Press, 1957.

[68] R. D. Smallwood and E. J. Sondik, “The Optimal Control of Partially Observable Markov Processes over a Finite Horizon,” *Operations Research*, vol. 21, no. 5, pp. 1071–1088, 1973.

[69] M. T. J. Spaan and N. Vlassis, "Perseus: Randomized Point-based Value Iteration for POMDPs," *Journal of Artificial Intelligence Research*, vol. 24, pp. 195–220, 2005.

[70] W. Koch, "On 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.

[71] H. Meikle, *Modern Radar Systems*. Artech House radar library, Artech House, 2008.

[72] E. M. P. Walraven, *Planning under Uncertainty in Constrained and Partially Observable Environments*. Phd thesis, TU Delft, Delft, The Netherlands, 2019.

[73] F. Hoffmann, A. Charlish, M. Ritchie, and H. Griffiths, "Policy Rollout Action Selection in Continuous Domains for Sensor Path Planning," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 57, pp. 2247–2264, 2021.

[74] M. I. Schöpe, H. Driessen, and A. Yarovoy, "A Constrained POMDP Formulation and Algorithmic Solution for Radar Resource Management in Multi-Target Tracking," *ISIF Journal of Advances in Information Fusion (JAIF)*, vol. 16, pp. 31–47, June 2021.

[75] 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.

[76] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers," *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[77] T. de Boer, "Radar Resource Management for Multi-Target Tracking Using Model Predictive Control," Master's thesis, TU Delft, Delft, The Netherlands, June 2021.

[78] T. de Boer, M. I. Schöpe, and H. Driessen, "Radar Resource Management for Multi-Target Tracking Using Model Predictive Control," in *Proceedings of the 24th International Conference on Information Fusion*, 2021, accepted for publication.

[79] P. Scokaert and D. Mayne, "Min-Max Feedback Model Predictive Control for Constrained Linear Systems," *IEEE Transactions on Automatic Control*, vol. 43, pp. 1136–1142, Aug 1998.

[80] S. Sowelam and A. Tewfik, "Optimal waveform selection for radar target classification," in *Proceedings of International Conference on Image Processing*, vol. 3, pp. 476–479 vol.3, Oct 1997.

[81] K. Bell, C. Kreucher, and M. Rangaswamy, "An Evaluation of Task and Information Driven Approaches for Radar Resource Allocation," in *Proceedings of 2021 IEEE Radar Conference*, 2021.

[82] D. Papageorgiou and M. Raykin, "A Risk-Based Approach to Sensor Resource Management," in *Advances in Cooperative Control and Optimization* (P. M. Pardalos, R. Murphey, D. Grundel, and M. J. Hirsch, eds.), (Berlin, Heidelberg), pp. 129–144, Springer Berlin Heidelberg, 2007.

[83] F. Bolderheij, F. G. J. Absil, and P. van Genderen, “A Risk-Based Object-Oriented Approach to Sensor Management,” in *2005 8th International Conference on Information Fusion*, vol. 1, pp. 8 pp.–, July 2005.

[84] B. La Scala, W. Moran, and R. Evans, “Optimal Adaptive Waveform Selection for Target Detection,” in *2003 Proceedings of the International Conference on Radar (IEEE Cat. No.03EX695)*, pp. 492–496, Sep. 2003.

[85] D. Hitchings and D. A. Castañón, “Receding Horizon Stochastic Control Algorithms for Sensor Management,” in *Proceedings of the 2010 American Control Conference*, pp. 6809–6815, June 2010.

[86] J. Langford and B. Zadrozny, “Reducing T-step Reinforcement Learning to Classification,” 2003.

[87] D. Blatt and A. O. Hero, “From Weighted Classification to Policy Search,” in *Proceedings of the 18th International Conference on Neural Information Processing Systems*, NIPS’05, (Cambridge, MA, USA), p. 139–146, MIT Press, 2005.

[88] S. Maskell, “Joint Tracking of Manoeuvring Targets and Classification of Their Manoeuvrability,” *EURASIP Journal on Advances in Signal Processing*, vol. 2004, no. 15, p. 613289, 2004.

[89] D. Angelova and L. Mihaylova, “Joint target tracking and classification with particle filtering and mixture Kalman filtering using kinematic radar information,” *Digital Signal Processing*, vol. 16, no. 2, pp. 180–204, 2006.

[90] J. M. Bernardo and A. F. M. Smith, *Bayesian Theory*. John Wiley & Sons, 1994.

[91] V. Savchuk and C. P. Tsokos, *Bayesian Theory and Methods with Applications*. Atlantis Studies in Probability and Statistics, Atlantis Press, 2011.

[92] L. D. Stone, R. L. Streit, T. L. Corwin, and K. L. Bell, *Bayesian Multiple Target Tracking*. Artech House, 2013.

[93] S. Julier and J. Uhlmann, “Unscented Filtering and Nonlinear Estimation,” *Proceedings of the IEEE*, vol. 92, pp. 401–422, March 2004.

[94] G. Ferrari, *Sensor Networks*. Signals and Communication Technology, Springer-Verlag, Berlin, Germany, 2009.

[95] S. Liu, S. P. Chepuri, M. Fardad, E. Maşazade, G. Leus, and P. K. Varshney, “Sensor Selection for Estimation with Correlated Measurement Noise,” *IEEE Transactions on Signal Processing*, vol. 64, pp. 3509–3522, July 2016.

[96] D. Zhang, M. Li, F. Zhang, and M. Fan, “New gradient methods for sensor selection problems,” *International Journal of Distributed Sensor Networks*, vol. 15, no. 3, p. 1550147719839642, 2019.

[97] N. Bogdanović, H. Driessen, and A. G. Yarovoy, “Target Selection for Tracking in Multifunction Radar Networks: Nash and Correlated Equilibria,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 54, pp. 2448–2462, Oct 2018.

[98] D. Wang, Q. Zhang, Y. Luo, X. Liu, J. Ni, and L. Su, “Joint Optimization of Time and Aperture Resource Allocation Strategy for multi-target ISAR Imaging in Radar Sensor Network,” *IEEE Sensors Journal*, vol. 21, pp. 19570–19581, 9 2021.

[99] Y. Shi, B. Jiu, J. Yan, and H. Liu, “Data-driven Radar Selection and Power Allocation Method for Target Tracking in Multiple Radar System,” *IEEE Sensors Journal*, vol. 21, pp. 19296–19306, Sep 2021.

[100] Y. Han, E. Ekici, H. Kremo, and O. Altintas, “Optimal Spectrum Utilization in Joint Automotive Radar and Communication Networks,” in *2016 14th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, pp. 1–8, May 2016.

[101] K. L. Bell, C. J. Baker, G. E. Smith, J. T. Johnson, and M. Rangaswamy, “Cognitive Radar Framework for Target Detection and Tracking,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 9, pp. 1427–1439, Dec 2015.

[102] 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.

[103] K. Jayachandra, “Approximately Optimal Radar Resource Allocation,” Master’s thesis, TU Delft, Delft, The Netherlands, July 2021.

[104] B. van der Werk, M. I. Schöpe, and H. Driessen, “Approximately Optimal Radar Resource Management for Multi-Sensor Multi-Target Tracking,” in *Proceeding of the 24th International Conference on Information Fusion*, 2021, accepted for publication.

[105] K. Jayachandra, M. I. Schöpe, and A. Yarovoy, “Joint Radar Communication Resource Allocation for Automotive Applications using Policy Rollout,” 2021, under review.

[106] Z. Wendt and J. S. Cook, “Saved by the Sensor: Vehicle Awareness in the Self-Driving Age.” https://www.machinedesign.com/mechanical-motion-systems/article/21836344/saved-by-the-sensor-vehicle-awareness-in-the-selfdriving-age. Accessed 2021-06-18.

[107] M. Jankiraman, *FMCW Radar Design*. Artech House, London, UK, 2018.

[108] J. Khoury, R. Ramanathan, D. McCloskey, R. Smith, and T. Campbell, “RadarMAC: Mitigating Radar Interference in Self-Driving Cars,” in *2016 13th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, pp. 1–9, June 2016.

[109] F. Uysal and S. Sanka, “Mitigation of automotive radar interference,” in *2018 IEEE Radar Conference (RadarConf18)*, pp. 0405–0410, April 2018.

[110] F. Uysal, “Phase-Coded FMCW Automotive Radar: System Design and Interference Mitigation,” *IEEE Transactions on Vehicular Technology*, vol. 69, pp. 270–281, Jan 2020.

[111] G. Hakobyan, K. Armanious, and B. Yang, “Interference-Aware Cognitive Radar: A Remedy to the Automotive Interference Problem,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 56, pp. 2326–2339, June 2020.

[112] K. V. Mishra, M. R. Bhavani Shankar, V. Koivunen, B. Ottersten, and S. A. Vorobyov, “Toward Millimeter-Wave Joint Radar Communications: A signal processing perspective,” *IEEE Signal Processing Magazine*, vol. 36, pp. 100–114, Sep. 2019.

[113] F. Lampel, F. Uysal, F. Tigrek, S. Orru, A. Alvarado, F. Willems, and A. Yarovoy, “System Level Synchronization of Phase-Coded FMCW Automotive Radars for RadCom,” in *2020 14th European Conference on Antennas and Propagation (EuCAP)*, pp. 1–5, March 2020.

[114] D. Bertsimas and J. N. Tsitsiklis, *Introduction to Linear Optimization*. Athena Scientific, 1st ed., 1997.

[115] S. S. Blackman and R. Popoli, *Design and Analysis of Modern Tracking Systems*. London :: Artech House,, 1999.

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

# Acknowledgements

This dissertation would not exist without a great deal of support and feedback throughout my PhD trajectory.

First of all, I would like to thank my daily supervisor and co-promotor Dr. Hans Driessen. I am deeply grateful for all the challenging questions and the encouragement that I received from him during our many discussions over the years. His expertise and his intuition helped me to formulate the research questions and methodology sharply. Moreover, I sincerely appreciate all the personal talks we had, during which we got to know each other quite well.

Secondly, I would like to sincerely thank my promotor Prof. DSc. Alexander Yarovoy for giving me the opportunity to conduct this PhD research in the MS3 group. If it were not for him, I would not have started the PhD trajectory in the first place. Whenever there was a problem, I could rely on his prompt feedback and support, for which I am extremely grateful.

Furthermore, I would like to thank the members of the i-CAVE P4 project for the interesting discussions. I especially thank Franz Lampel, Dr. Recep Firat Tigrek, Dr. Francesco Laghezza, Dr. Alessio Filippi, and Prof. Dr. Frans Willems, who I additionally thank for being part of my doctoral committee.

I sincerely thank the other members of my doctoral committee for evaluating my thesis: Prof. Dr. Geert Leus, Prof. Dr. Henk Blom, Dr. Matthijs Spaan, and Dr. Alexander Charlish. I especially thank Prof. Dr. Henk Blom, who attended my Go/No Go meeting after the first year and Dr. Matthijs Spaan, who took the time for several discussions over the past four years. I greatly appreciate all the helpful feedback that I received.

Additionally, I would like to thank Dr. Peyman Mohajerin Esfahani for his valuable feedback during multiple discussions.

Moreover, I would like to thank the MS3 support staff and secretaries for their great help during these stressful years. Without them, we would not be able to conduct our research so smoothly. I would especially like to thank Esther de Klerk, Minaksie Ramsoekh, Minke van der Put, Fred van der Zwan and Pascal Aubry.

Of course, I would also like to thank my current and former office mates for the many informal discussions that we had next to the coffee machine: Nikita Petrov, Utku Kumbul, Yun Lu, Jeroen Overdevest, Saravanan Nagesh, Valantis Kladogenis, Wietse Bouwmeester, Tworit Dash, Ronny Gündel, Nicolas Kruse, Berk Onat, Ignacio Roldan Montero, Sen Yuan, and Gerardo Parmentola. It was nice to know that I am not alone in this and that other people face similar challenges.

Furthermore, I thank all my other current and former colleagues: Bert-Jan Kooij, Yanki Aslan, Francesco Fioranelli, Oleg Krasnov, Peter Swart, Dinh Tran, Faruk Uysal, Nikola Bogdanović, Ozan Dogan, Jianping Wang, Jan Puskely, Rossiza Gourova, Nannan Chen, Sharef Neemat, Shengzhi Xu, Shilong Sun, Peter Svenningsson, Etienne Goossens and Çiğdem Okuyucu.

A special thank goes to the MSc students I had the opportunity to supervise: Bas van der Werk, Thies de Boer, and Karan Jayachandra. Our discussions and their critical questions helped me get a deeper understanding of my research topic.

Additionally, I would like to thank some of my fellow students during my master's studies, who made my first two years at TU Delft much more fun: Cantika Felita, Kajeng Wangkheimayum, Sjoerd Bosma, Zhengzheng Wang, Ralph van Schelven, Sasanka Sanka, and Diego Emer.

Moreover, I would like to thank one of my professors in Hamburg, Prof. Dr. Annabella Rauscher-Scheibe, and my supervisor at NXP in Hamburg, Renke Bienert. They both supported me in my decision to study in the Netherlands.

Finally, I would like to thank my parents and brothers for their support during these challenging years. And of course, I genuinely thank my girlfriend, Kalliopi Papangelopoulou, for her love, support, and patience. I would not have finished my PhD studies without her. Additionally, I thank our cat Froso for spending many hours next to me while working from home during the pandemic.

# About the Author

**Max Ian Schöpe** was born in Kassel, Germany, on January 5, 1990. After graduating from high school in Kassel, he moved to Hamburg, Germany, to start his Bachelor's studies of "Electrical Engineering" at the Technical University of Hamburg (TUHH) in 2009. After switching universities, he received his BEng in "Information and Electrical Engineering" at the University of Applied Sciences Hamburg (HAW Hamburg) in February 2015. Subsequently, he moved to The Netherlands and started the MSc program of "Telecommunication and Sensing Systems" at Delft University of Technology (TU Delft), which he finished in October 2017. After his successful graduation, Max Ian Schöpe started as a PhD candidate in the Microwave Sensing, Signals and Systems group of TU Delft. His main research was conducted in the area of Radar Resource Management and contained the formulation of a generic framework applying Partially Observable Markov Decision Processes.

# List of Publications

7. **M. I. Schöpe**, H. Driessen, and A. Yarovoy, “Optimal Threat-based Radar Resource Management for Multi-Target Joint Tracking and Classification”, *ISIF Journal of Advances in Information Fusion (JAIF)*, 2021, under review.

6. K. Jayachandra, **M. I. Schöpe**, and A. Yarovoy, “Joint Radar Communication Resource Allocation for Automotive Applications Using Policy Rollout”, *IEEE Transactions on Vehicular Technology*, 2021, under review.

5. B. van der Werk, **M. I. Schöpe**, and H. Driessen, “Approximately Optimal Radar Resource Management for Multi-Sensor Multi-Target Tracking”, in *Proceedings of the 24th International Conference on Information Fusion (FUSION)*, Sun City, North West, South Africa, 2021, accepted for publication.

4. T. de Boer, **M. I. Schöpe**, and H. Driessen, “Radar Resource Management for Multi-Target Tracking Using Model Predictive Control”, in *Proceedings of the 24th International Conference on Information Fusion (FUSION)*, Sun City, North West, South Africa, 2021, accepted for publication.

3. **M. I. Schöpe**, H. Driessen, and A. Yarovoy, “A Constrained POMDP Formulation and Algorithmic Solution for Radar Resource Management in Multi-Target Tracking”, *ISIF Journal of Advances in Information Fusion (JAIF)*, vol. 16, no. 1, pp. 31-47, Jun. 2021.

2. **M. I. Schöpe**, H. Driessen, and A. Yarovoy, “Multi-Task Sensor Resource Balancing Using Lagrangian Relaxation and Policy Rollout”, in *Proceedings of the 23rd International Conference on Information Fusion (FUSION)*, Online, 2020.

1. **M. I. Schöpe**, H. Driessen, and A. Yarovoy, “Optimal Balancing of Multi-Function Radar Budget for Multi-Target Tracking Using Lagrangian Relaxation”, in *Proceedings of the 22nd International Conference on Information Fusion (FUSION)*, Ottawa, Ontario, Canada, 2019.