Delft University of Technology

Approximately Optimal Radar Resource Management for Multi-Sensor Multi-Target Tracking

A First Step Towards a Distributed Solution for Radar Resource Management in a Sensor Network

Author:
H.B. van der Werk BSc

Supervisors:
M.I. Schöpe MSc
dr.ir. J.N. Driessen

May 27, 2021

Approximately Optimal Radar Resource Management for Multi-Sensor Multi-Target Tracking

A thesis submitted to the Delft University of Technology in partial fulfillment of the requirements for the degree of

Master of Science in Electrical Engineering

by

H.B. van der Werk Bsc

May 2021

This thesis was written in collaboration with:

Microwave, Sensing, Signals & Systems Group
Department of Microelectronics
Faculty of Electrical Engineering, Mathematics & Computer Science
Delft University of Technology

Supervisors: M.I. Schöpe MSc
dr.ir. J.N Driessen

Abstract

The Radar Resource Management (RRM) problem in a multi-sensor multi-target scenario is considered. The problem is defined as a constrained optimization problem in which the predicted error covariance is minimized subject to resource budget constraints. By applying Lagrangian Relaxation (LR) the problem is decoupled into multiple sub-optimization problems.
The problem is modeled according to a Partially Observable Markov Decision Process (POMDP). Using a stochastic optimization framework called policy rollout, the POMDP is solved non-myopically by looking ahead into the expected future.
Two novel implementations, namely a centralized and distributed implementation are presented as viable approaches for solving this problem for a multi-sensor case.
The centralized implementation, defined as the approximately optimal solution, utilizes a global policy per task. As such, the policy rollout for a single target needs to explore the actions of multiple sensors.
The distributed implementation is considered as a practical alternative to improve on the computational complexity of the policy rollout of the centralized implementation. Now per sensor and per task a policy rollout is computed. To maintain a similar performance as the centralized implementation, at the beginning of each policy rollout the last known actions of the other sensors are shared.
An additional third independent implementation is considered. The independent implementation uses no communication during the optimization process and is considered to be the implementation with the lowest performance with respect to the cost.
All implementations have been applied to multiple two-dimensional simulated radar tracking scenarios. A comparison is made between the centralized, distributed and independent implementation based on the average cost and runtime. Results indicate that both the centralized and distributed implementation outperform the independent implementation with respect to the cost by a factor two. Subsequently, the distributed solution converges to similar results as the centralized implementation while requiring significantly less computational resources.

Preface

This thesis finalizes my Master studies Electrical Engineering at the TU Delft. When I first started my studies in 2014 at the TU Delft, I had no clear vision in my mind of what my future studies would look like and which direction or track within Electrical Engineering I would select. Over the course of my bachelor, my interest towards signal processing in fields such as radar and audio increased which finally resulted in applying for the master track Signals & Systems. The selection of courses that I followed during my master culminated into doing a thesis project in the research group Microwave, Sensing, Signals & Systems (MS3) related to radar systems.

This thesis aims at finding a solution for resource management problems in multi-radar systems by applying optimization techniques such as Lagrangian relaxation and the subgradient method combined with stochastic optimization techniques and data fusion.
The idea behind this thesis originated from the work of Max Schöpe, who for the past three years has been focused on formulating solutions for sensor resource management in a single sensor. During my literature study, I noticed that he did not develop an extension to sensor resource management in a sensor network yet. Hence, I started to focus on formulating new implementations for solving this problem.

It was a great experience to take on this project and to be able to really go into the details of one specific topic. Learning new concepts and applying them was not always easy to do and I am thankful for the help I received from others. Hence, I would like to express my gratitude to all the people that supported me during my thesis project.

First of all, I would like to thank my daily supervisor, Max Schöpe for helping me throughout the duration of my thesis project. I am very grateful for all the usefull discussions that we had and for all the quick feedback I received even during these difficult times.
Secondly, I would like to thank dr. ir. Hans Driessen for the bi-weekly discussions that we had. He managed to guide me in the right direction when needed and gave me the opportunity to explore new ideas as well.
Thirdly, I would like to thank prof. dr. Alexander Yarovoy for the opportunity of performing my thesis project in the Microwave, Sensing, Signals & Systems group.
Finally, I would like to thank my family and friends for the support that they gave me throughout my studies.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The development of sensor technologies and software over the last couple of decades has led to a significant increase in the amount of controllable parameters in sensing devices. Sampling rates, bandwidths, center frequencies and more have become addressable on the fly[1]. Controlling each of these parameters to fulfill certain operational constraints and objectives is what is defined as Sensor Resource Management (SRM). More specifically SRM is related to defining a policy such that a certain objective is optimized to ensure that the sensor operates at its maximum operational capacity. In general, the SRM problem is often considered to be a scheduling problem[1].

For the particular topic of radar system design, resource management has become increasingly more important. Radar was traditionally mainly focused on a single application or task such as target tracking, surveillance or detection. The introduction of phased array radar, actively electronically scanned array technology, digital beamforming and waveform generation caused a paradigm shift from specialized systems focussing on a single task towards so-called Multi-function radar (MFR) systems. Systems capable of performing multiple tasks jointly[2].

The increased capabilities of MFR systems offer the possibility of instantaneous and automatic adjustment of transmission properties in time and frequency for optimal performance. Which is achieved based on information from previous measurements. This automatic adaptation is often referred to as Radar Resource Management (RRM) and is often considered in the framework of cognitive radar[3, 4, 5, 6].

One of the major challenges that MFR systems face is the limited amount of resources available. During operation MFR systems will often operate at their resource limit. Increasing the budget allocated to a single task will have effect on the available budget for the other tasks. Consequently, budget allocation requires careful consideration to ensure the best possible performance of the radar system.

In some cases, the radar system is extended to a sensor network consisting of multiple MFR systems capable of communicating with each other to exchange new information. By allowing communication between MFR systems, information from multiple connected MFR systems can be combined to improve on the performance of the network (e.g. the accuracy, resolution, coverage or robustness). Increasing the performance of the sensor network limits the amount of wasted resources and inevitably will improve the situational awareness of MFR systems.

1.1 Problem Definition

Consider a sensor network consisting of M sensor nodes where each node in the sensor network is capable of communicating with all other nodes in the sensor network. Per sensor, a limited resource budget is available which can be allocated to certain tasks or functions to improve the performance of the sensor itself but also the performance of the sensor network.
For a standalone sensor, there are already existing approaches that try to allocate the resources over multiple tasks in an approximately optimal manner such that the uncertainty of the environment is as low as possible. For a multi-sensor scenario however, there is little to no literature available up to this point for this specific topic. By combining the information of multiple connected radar sensors placed at different locations, the resolution, overall coverage and accuracy of the sensor network can be improved even further.
This thesis aims at extending the solution for resource allocation in a single sensor to a multi-sensor scenario. The problem here is focused on how each sensor should allocate its limited budget over a set of tasks while taking into account the other sensors in the sensor network.
Also, what information should be shared to improve the performance of the budget allocation algorithm? How can sensors with overlapping fields of view be optimally managed? Can the algorithm be implemented in a decentralized fashion while still managing to attain good performance? What is the best way to combine information (e.g. estimates or measurements) from several sensors to solve SRM problems?

1.2 Thesis Outline

Chapter 2 covers some fundamental information related to the working of a radar system. Chapter 3 provides an overview of the relevant solutions that already exist in literature related to the topic of SRM in sensor networks. The novel algorithms for solving the RRM problem in sensor networks are presented in Chapter 4. In Chapter 5 the novel algorithms are applied to several simulated scenarios to verify their validity and performance. Finally in Chapter 6 a conclusion and some final remarks are given.

Chapter 2

General Radar Background

The concept of radar resource management in MFR requires some fundamental background on the workings and configurations of radar systems in general. This section aims at providing general radar information and radar concepts required for resource management in MFR systems. The material presented in this chapter is loosely based on the work presented in [7] and [8].

Basic Single Antenna Radar

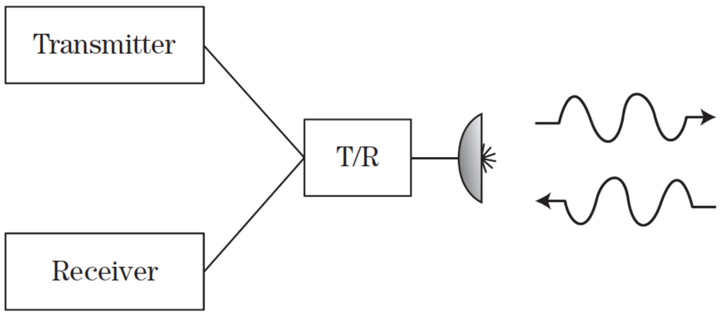
A basic radar system is comprised of a transmitter, receiver, some logic to switch between transmit and receive and a single antenna used for both the reception and transmission of signals (see Fig. 2.1a).
The transmitter is a system capable of transmitting electromagnetic (EM) waves at high power levels (i.e., in the range of kilo- or even megawatts) into a desired direction. If the propagating waves encounter an object (e.g. an airplane) then part of the signal is reflected and propagated back to the radar system. The time t between transmit and receive can be used to calculate the range of the object with respect to the radar according to,

$$R = \frac{c \cdot t}{2} \tag{2.1}$$

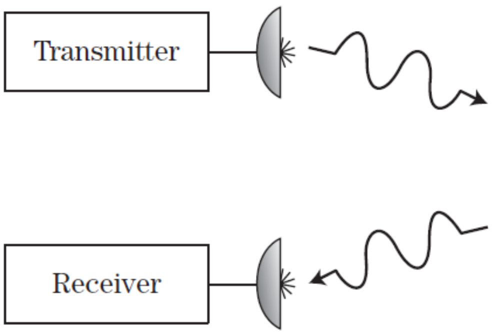
with c the speed of light, which is the speed at which the EM waves propagate through the medium (approximately 3.0×10^8 m/s). As t in this case represents the round trip time of the EM wave, the product between the speed of light and round trip time needs to be divided by two.
The velocity of a target can be measured by exploiting the phase of frequency shift between the transmitted and received pulse which is induced due to the movement of the target. This difference in observed and transmitted frequency is defined as the Doppler frequency. The Doppler frequency f_d can be computed using the wavelength and the relative velocity of the target with respect to the radar v_{target} according to,

$$f_d = \frac{v_{target} \cdot \lambda}{2} = f_{transmitted} - f_{received} \tag{2.2}$$

with f$_{transmitted}$ and $f_{received}$ the transmitted and received frequencies respectively. Where it is assumed that V$_{target}$ << c.
Since only a small fraction of the transmitted EM signal will reflect back to the radar system, the receiver needs to be able to measure low power EM waves (i.e milli- up to nanowatts).



(a)



(b)

Figure 2.1. block diagrams of a basic monostatic radar configurations. (a) is a monostatic configuration and (b) a bistatic configuration[7].

Radar Equation

The elementary equation used for all radar systems is called the radar range equation. The radar range equation describes mathematically the physical dependencies between parameters involved in the transmit and receive process of the radar system including the transmit power, wave propagation and power of the received signal.
Consider the radar system depicted in Fig. 2.2. During operation, the radar system transmits EM waves towards a certain target at range R. As the incident EM wave hits the target, time-varying currents are induced on the target, making it a source of emanating radio waves from which a part will propagate back to the radar system. The corresponding power related to the reflected wave is defined as a combination of the transmitted power and the Radar Cross Section (RCS). Where the RCS, denoted with σ, is a measure of the ratio of the backscattered power towards the radar over the total power intercepted by the target, i.e., it is a measure of how the radar system perceives the size of the target[9]. It depends on the size and shape of the target and the materials from which the target is made. The reflected power is given by,

$$P_{refl} = \frac{P_t G_t \sigma}{4\pi R^2} \tag{2.3}$$

with P_t, the transmitted power and G_t the gain of the transmit antenna. The reflected signal propagates back towards the radar system over a range R such that the power density at the receiver (Q_r) is given by,

$$Q_r = \frac{P_{refl}}{4\pi R^2} \tag{2.4}$$

The received power at the antenna from a target at range R is computed by taking the product of the power density and the effective area A_e. The effective area is a measure of how effective the antenna is at receiving power (i.e., for an ideal antenna the effective area is equal to its physical size),

$$P_r = Q_r A_e = \frac{P_t G_t A_e \sigma}{(4\pi)^2 R^4} \tag{2.5}$$

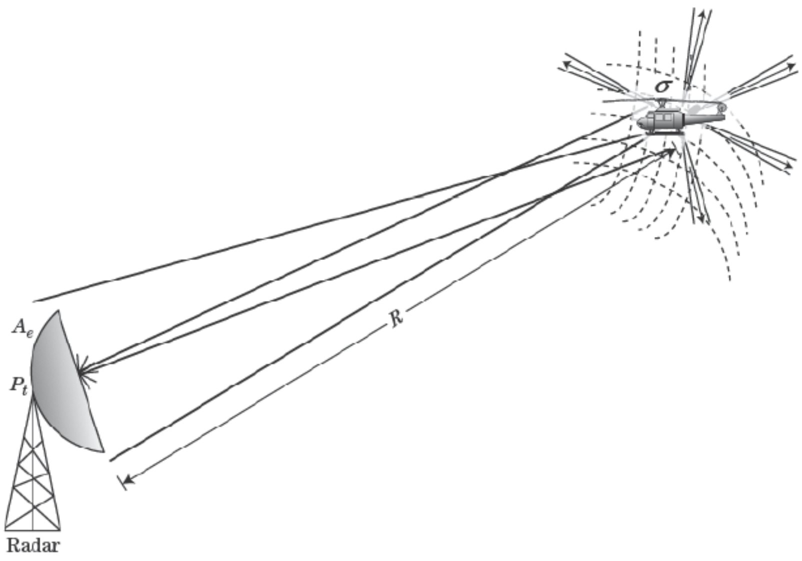


Figure 2.2. Depiction of the transmit and receive process of a radar system. The radar (left) emits EM waves towards a target at range R (right). Part of the reflected signal is propagated back to the radar to be measured [7].

The effective area can be expressed in terms of received antenna gain and the wavelength λ,

$$G_r = \frac{4\pi A_e}{\lambda^2} \tag{2.6}$$

Substituting 2.6 into 2.5 results in,

$$P_r = \frac{P_t G_t G_r \lambda^2 \sigma}{(4\pi)^3 R^4} \tag{2.7}$$

Equation 2.7 is an expression for the radar range equation.

Phased Array radar

The radar system under consideration in this thesis is a pulsed phase array tracking radar. The reasoning behind this comes from the fact that pulsed radar system are well suited for tracking targets, while the phased array opens up many degrees of freedom from a scanning point of view. This section will briefly explain both concepts separately.
A pulse radar in general is a radar which emits short and powerful pulses of EM waves, these EM waves propagate through a medium and reflect from potential targets. The portion of the reflected signal that is bounced back towards the radar can consequently be measured by the pulse radar in the silent period, which is the period in which the radar does not transmit pulses.
Relevant parameters of a pulsed radar system are the dwell time and revisit time. The dwell time, denoted by τ, is defined as the length of each pulse being transmitted and the revisit time, denoted by T, is defined as the time duration between each consecutive measurement. Note that the dwell time is in general smaller than the revisit time.
Phased array radar systems are an essential part of the topic at hand. The basic form of a phased array system is depicted in Fig. 2.3. The phased array radar is a collection of antenna elements assembled in some configuration in such a way that the radiation pattern of each individual element is combined with the radiation pattern of surrounding elements to form a radiation pattern called the main lobe. By allowing phase shifts for every individual element determined by a controller C, the main lobe can be steered in any desired direction.
For example, if it is assumed that the direction of the main lobe (ψ) is oriented within

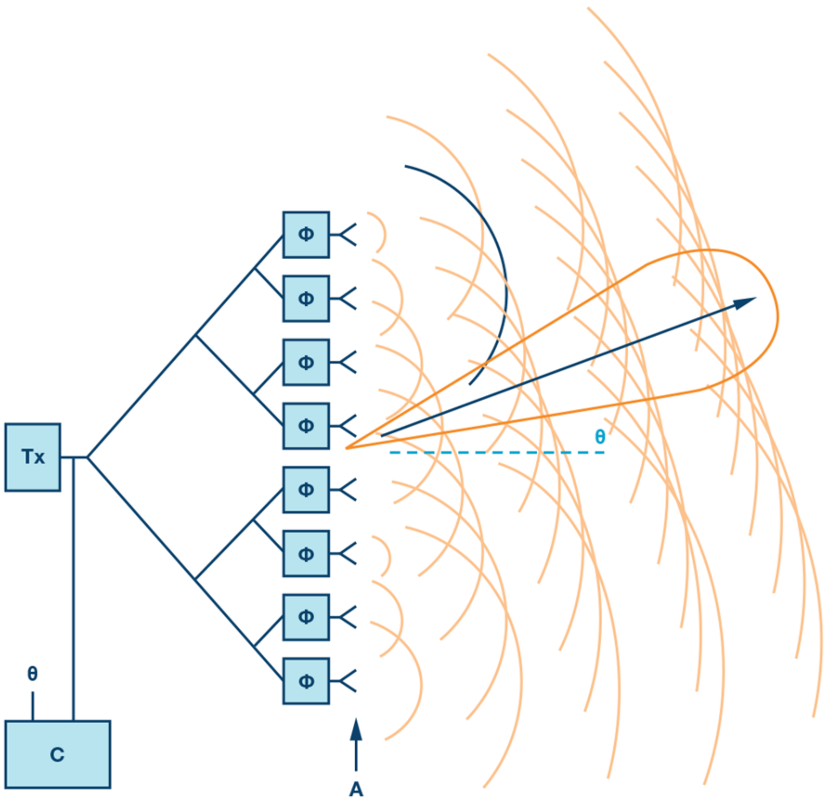


Figure 2.3. Diagram of an elementary phased array system. The controller (C) determines the phase shifts between the antenna elements to determine the direction of the main lobe[10].

± 90 deg then the phase difference β between the elements must be adjusted such that the following holds,

$$\psi = kdcos(\theta) + \beta \tag{2.8}$$

with k, the wavenumber given by $k = \frac{2\pi}{\lambda}$, d is the distance between elements, θ is the steering angle and β is a phase shift. The big advantage of using such a system is related to the high beam agility and the possibility of changing the beam direction in a fast manner.

Multi-Function Radar

The possibilities introduced with the phased array radar led to the development of the Multi-Function Radar (MFR). The MFR system exploits the operational benefits of the phased array radar to support numerous and potentially conflicting tasks. The MFR system is capable of simultaneously generating independent beams which can be utilized for different tasks. Fig. 2.4 shows an example of an MFR system mounted on a ship.
In this specific example, multiple transmit beams are generated to support missile guidance, target tracking, target classification, target confirmation but also for surveillance purposes. A certain amount of resource budget i.e., time or energy must be allocated by the system for each individual task of the system. With an increasing number of tasks, the amount of energy or time spent on each task must be considered carefully. Hence, effective resource management is required for reaching maximum performance. The MFR system distinguishes itself from a standard phased array radar system by automatically managing and configuring the available resources[11].

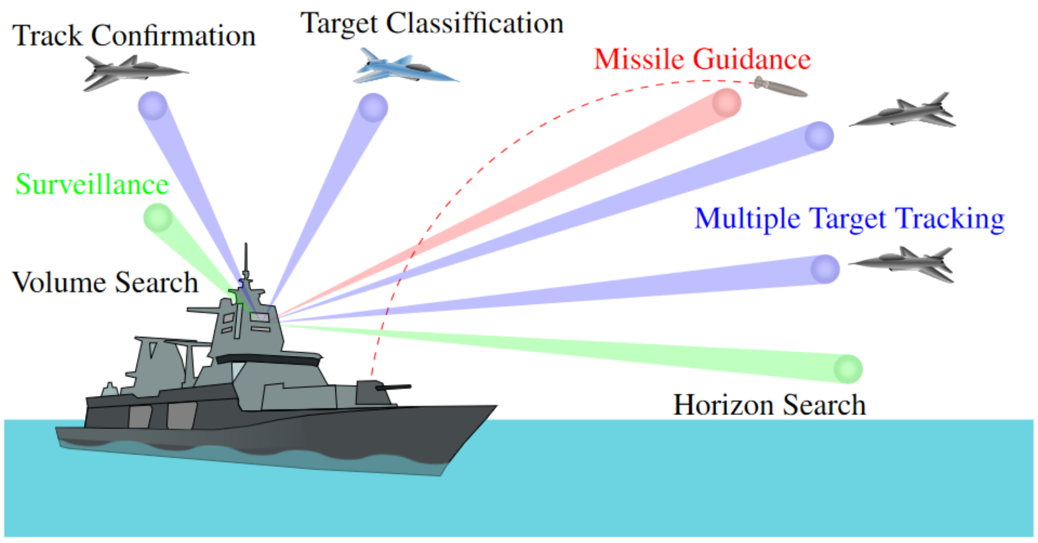


Figure 2.4. A ship mounted MFR system[12].

Chapter 3

Existing Solutions

The goal of this chapter is to provide a review of the main concepts related to SRM with a focus on RRM in a sensor network. First, section 3.1 will cover the RRM topic and the theory required to solve a RRM problem for a single sensor. Section 3.2 will aim to highlight the relevant background of sensor networks to lay the foundation for expanding the RRM problem to a multi-sensor scenario.

3.1 Sensor Resource Management

In general, the SRM problem for a single sensor refers to the problem of searching for an (optimal) set of actions that the sensor should take based on a given task that maximizes or minimizes a certain performance. For example, finding the best possible actions based on the (uncertain) knowledge of the state of a system (e.g. the state of a moving target). Instead of searching for the actions of a single task, the interest lies in jointly searching for the actions of multiple tasks.
The sensors considered in SRM problems (e.g. a MFR) often operate at their resource limit. Hence, the budget distribution over all tasks needs to be taken into account instead of analyzing the budget allocation for each task individually. If the budget is reduced for one task, the sensor can simultaneously increase the budget for other tasks. As such, the allocated budgets are mutually dependent.
Section 3.1.1 provides a description of the theoretical framework of optimization theory such as the method of Lagrange multipliers and the subgradient method but also the reasoning behind the cost function selection. Section 3.1.2 provides the stochastic optimization framework and a method called policy rollout. In Section 3.1.3 a full solution will be given for the RRM problem in a single sensor.

3.1.1 Optimization

Due to the nature of resource management problems, optimization is a field often utilized for solving the RRM problem. The field of optimization is rather broad and contains many theories. However, for this thesis specific, a subset of theories and literature are selected that are deemed relevant.

Cost function Selection

A fundamental aspect in RRM problems revolves around the selection of a proper objective function that needs to be evaluated. Although the topic of RRM has been widely studied from different perspectives, the selection of the objective function is still under investigation and there has not yet been an optimal solution. Selection of the cost function heavily depends on the type of approach used for solving the RRM problem. A comprehensive overview of different approaches has been given by Katsilieris[13]. He divides the approaches into the following four categories,

- Heuristic
- Task-Based
- Information-driven
- Risk-Based

The heuristic or rule based approach refers to a set of rules that determine the behaviour of the resource management problem. For example setting a threshold on the track uncertainty such that the maximum revisit time is not exceeded. Other examples can be found in e.g. [14] or [15]. The approach in general is rather basic to implement and it can simplify complex problems. However, the behavior of this approach can become unpredictable,the computational complexity can be significant due to e.g. many nested rules and in addition to that the approach is not very flexible. In addition to this, RRM will perform best if the resources are assigned based on mission objectives rather than rules, making the heuristic approach not the most usable[16].
Task based resource management focuses on optimizing quantities related to the sensing tasks and the operational goal of the system. Some potential cost functions mentioned by [13] in task based resource management are,

- Expected probability of existence of a target
- Expected Signal to Noise Ratio (SNR)
- Expected uncertainty in the position estimate of a target

For tracking a target, an often used approach is to express the uncertainty of the position of a target in terms of the error covariance. More specifically, the trace of the error covariance is an often used cost due its simple form and intuitive implementation. Downside of this approach is the fact that it does not explicitly take into account what the users needs are.
The information driven approach utilizes information theoretic measures of uncertainty. Some example cost functions are the Shannon Entropy or the Kullback-Leibler Divergence. Disadvantage of this approach is the fact that it is less intuitive to apply to different operational contexts.
The risk based approach is an approach that takes into account the operational goal of the system. However, the approach does not try to directly optimize the uncertainty in the problem. It could however be included into the risk or threat definition for example.
Apart from the heuristic approach all categories are suitable and can be applied in different ways. Since the measures used are mainly task based, the task based approach appears to be the most straightforward to utilize for resource management.
Besides [13], the topic related to cost function selection for sensor management has also been covered by e.g. [17].

Lagrangian Relaxation

Consider a constrained optimization problem,

$$\begin{aligned} \min_{x} \quad & f(x) \\ \text{subject to} \quad & \mathbf{g}_i(x) = \mathbf{c}_i, \quad \forall \mathrm{i} \in \mathrm{N} \\ & \mathbf{h}_j \geq \mathbf{d}_j, \quad \forall \mathrm{j} \in \mathrm{M}. \end{aligned} \tag{3.1}$$

With N equality constraints and M inequality constraints. Lagrangian Relaxation (LR) can be used to relax the constrained problem into an unconstrained problem. Thereby reducing the complexity of the problem.
The idea in the method of Lagrange multipliers is to take into account the constraints by

adding them to the objective function f(x) as a weighted sum of the constraint functions. Define a weight vector associated to inequality constraints as λ and μ as a weight vector related to equality constraints respectively. The Lagrangian L of problem 3.1 is given by,

$$L(x, \lambda, \mu) = f(x) + \sum_{i=0}^{N} \lambda_i g_i + \sum_{j=0}^{M} \mu_j h_j \tag{3.2}$$

Note that a generalized form of the method of Lagrange multipliers is assumed here. Meaning that the Karush-Kuhn-Tucker (KKT) conditions are satisfied. This allows both inequality and equality constraints to be taken into account. See [18] on convex optimization for further details on this theorem.
The Lagrangian dual function can now be defined as,

$$d(\lambda, \mu) = \inf L(x, \lambda, \mu) = \inf \left[f(x) + \sum_{i=0}^{N} \lambda_i g_i + \sum_{j=0}^{M} \mu_j h_j \right] \tag{3.3}$$

The optimal values between the primal (3.1) and the dual (3.3) are not necessarily the same. Define the optimal value of the primal problem as p*. Then for any $\lambda > 0$ and any μ the following holds,

$$d(\lambda, \mu) \leq p^* \tag{3.4}$$

Related papers that utilize LR for resource management are for example [19] and [20].

Subgradient Method

Searching for optimal values of a given optimization problem can be achieved with many different approaches. One of these approaches is the subgradient method as stated in Algorithm 1.

```
Algorithm 1: Subgradient Descent
   Input: $N \in \mathbb{Z}_{>0}, x_0, g_i, \gamma, \epsilon$
   Output: $x^* \in \mathbb{R}$
1  $i = 0$
2  while $g_i < \epsilon$ do
3      $x_{i+1} = x_i - \gamma * g_i$
4      $i = i + 1$
5  end
6  $x^* = x_N$
7  return $x^*$
```

Where $g_k \in \delta f(x_i)$ is any subgradient of $f(x)$, N is the amount of derivative evaluations, x_0 is the initial guess, γ the step size with $\gamma > 0$ and ϵ a value used as stopping criterion. The stopping criterion in this case is reached if the gradient converges to some small value (i.e., $g_i << 1$). Note that a stopping criteria could also be based on a norm of the difference in optimum points or a duration. When the appropriate step size and initial guess are selected convergence is ensured[18].
The main advantage of using the subgradient method is the fact that the objective function does not necessarily need to be differentiable. An additional advantage of using the subgradient method is that in some case it can be combined with LR to develop a simple distributed algorithm for a given problem[21].

ADMM

The combination of LR and the subgradient method offers the possibility to relax an optimization problem into multiple smaller problems that can be solved in parallel. As an alternative to this approach, the Alternating Direction Method of Multipliers could also be applied.
Similarly as with LR, ADMM can split an optimization problem into multiple parts. This

is achieved by decomposing the optimization variable x into two variables x and z. Where the objective function is separable among this decomposition i.e.„

$$\begin{aligned} \min_{x} \quad & f(x) + g(z) \\ \text{subject to} \quad & Ax + Bz = c \\ & \mathbf{h}_j \geq \mathbf{d}_j, \qquad \forall \text{j} \in \text{M}. \end{aligned} \tag{3.5}$$

Consequently, instead of applying LR, the augmented Lagrangian is formulated according to,

$$L(x, z, \lambda) = f(x) + g(z) + \lambda^T (Ax + Bz - c) + \frac{\rho}{2} ||Ax + Bz - c||_2^2 \tag{3.6}$$

with ρ, the augmented Lagrangian parameter. Using an iterative update scheme, the optimal values for x and z are computed[22].
ADMM is well suited for solving optimization problems with a large number of parameters. Some literature related to solving optimization problems in sensor networks using ADMM is provided by [23, 24, 25, 26, 27].
ADMM offers a generalized framework that requires the loss function to be not differentiable and it is simple to implement. However, the convergence rate of ADMM is poor. In addition to that, for any particular problem it is likely that other methods will perform better compared to ADMM[22].

3.1.2 Stochastic Optimization

The Sensor Resource Management problem can be classified as a stochastic optimization problem, which is a term used for methods that perform decision making under uncertainty. Stochastic optimization is not one single field but rather a loose collection of many different approaches in many different applications[28].
The objective in stochastic optimization is to search for a policy that utilizes the available information on random variables to find an optimal action[29].
Although the focus of this thesis is not so much on the subject of stochastic optimization, it is a frequently used element to solve the radar resource management problem. Hence, a comprehensive description will be provided. More specifically, this section describes a stochastic optimization framework and a corresponding stochastic optimization technique called policy rollout. The theory presented in this section is loosely based on the material presented in [28] and [29].

General Problem Components

A stochastic optimization problem generally can be defined by several components. These include,

- State Space, a description of all possible states that the system can reach. This can be modelled with a random state vector $\mathbf{s}_k$ at time step k. The realisation of a system state is denoted by $\mathbf{s}_k \in \mathscr{S}$ where $\mathscr{S}$ is the system state space.
- Actions and Action space, these variables constitute the action or actions that can be taken at a time step k. The transition from one state to the next is influenced by the selection of the respective action. Decisions can be binary, discrete, continuous and categorical. The notation used in this thesis for the realisation of an action is $\mathbf{a}_k \in \mathscr{A}$ where $\mathscr{A}$ is the action space.
- Exogenous Information, any new information that becomes known at some decision step can be defined as exogenous information. Another term often used instead of exogenous information is the observation space. This information can be modeled by a vector $\mathbf{Z}_k$. The realisation of any exogenous information is denoted with $\mathbf{z}_k$.
- State Transition Function $\mathbf{F}$, Describes the evolution of a state from one time step to the next.

- The Cost Function **C**, represents the cost or reward at each respective time step. In general this is a value that can depend on external or internal variables (e.g. system state and actions) and needs to be optimized.

- Discount factor $\gamma \in [0,1]$. Discounts future time steps.

The notation of each of the components is defined in discrete time. Unless specified otherwise, the time will be regarded as discrete for the remainder of this thesis.
A relevant control framework that can be described using these elements is the Markov Decision Process (MDP). The MDP has been used successfully in stochastic environments. At each time step k, the process is in some state $\mathbf{s}_k$. A controller chooses an action $\mathbf{a}_k$ causing the state to transition to a random new state $\mathbf{s}_{k+1}$. A corresponding cost is then returned to the controller.
A prerequisite of utilizing the MDP is that the system state should be completely observable. Meaning in this case that the radar system which functions as an agent should be able to describe the state of the environment with full certainty. The radar system used however is not able to fully describe the system state as a result of noisy measurements. To overcome this, the MDP can be relaxed into a Partially Observable Markov Decision Process (POMDP).

Partially Observable Markov Decision Process

Consider Fig. 3.1, showing a schematic representation of a POMDP. The POMDP system is comprised of an environment and an agent, responsible for sensing the environment. The environment is represented by a set of states. The agent can select from a set of decision variables an action to influence the system state. However, in this thesis it is assumed that the actions cannot change the system state but rather the knowledge about it.
At each time step k, the agent takes an action in a state causing the environment to transition to the next state via the state transition function. Consequently, an observation is made of the state and a corresponding cost is computed. The cost and observation are received by the agent.

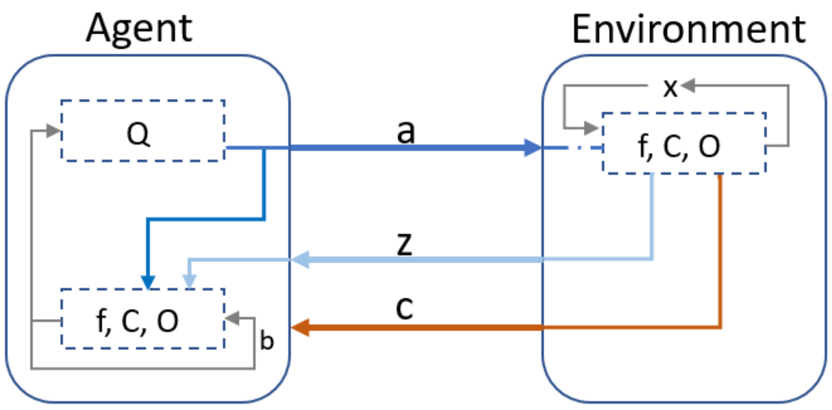


Figure 3.1. Schematic depiction of a POMDP process. The agent chooses an action a, causing a state transition. An observation z and reward r are returned to the agent[30].

The difference between using a POMDP over an MDP has to do with what the agent receives. Where a fully observable MDP observes the next state, the POMDP only receives an observation z of the next state. Define an observation $z \in \Omega$, where Ω represents the set of all possible observations that the agent can receive. This observation, combined with the action is then used to compute a belief state which is expressed as a probability distribution indicating the likelihood of the current state of the environment[31].
The goal of the agent is to choose the best possible actions that minimize the expected

future cost starting at time step k over a time horizon H. Define the cost, V_H, over the time horizon as,

$$V_H = \sum_{k=k_0}^{k_0+H} C(\mathbf{s}_k, \mathbf{a}_k) \tag{3.7}$$

The expected cost given the belief state is expressed as,

$$C_B(\mathbf{b}_k, \mathbf{a}_k) = \sum_{s\in S} \mathbf{b}_k C(\mathbf{s}_k, \mathbf{a}_k) \tag{3.8}$$

The cost can now be expressed as a value function $V_H(\mathbf{b}_k)$ representing the expected future cost,

$$V_H(\mathbf{b}_{k_0}) = E[\sum_{k=k_0}^{k_0+H} C_B(\mathbf{b}_k, \mathbf{a}_k)|\mathbf{b}_{k_0}] \tag{3.9}$$

Using Bellman's equation as defined in [32] the optimal value function for belief state $\mathbf{b}_0$ and action $\mathbf{a}_0$ is expressed as,

$$V_H^*(\mathbf{b}_0) = \min_{\mathbf{a}_0\in\mathbf{A}} (C_B(\mathbf{b}_0, \mathbf{a}_0) + \gamma \cdot E[V_{H-1}^*(\mathbf{b}_1|\mathbf{b}_0, \mathbf{a}_0)] \tag{3.10}$$

The optimal policy is then given by the set of values of $\mathbf{b}_0$ for which the value function is minimized,

$$\pi_0^*(\mathbf{b}_0) = \arg\min_{\mathbf{a}_0\in\mathbf{A}} (C_B(\mathbf{b}_0, \mathbf{a}_0) + \gamma \cdot E[V_{H-1}^*(\mathbf{b}_1|\mathbf{b}_0, \mathbf{a}_0)] \tag{3.11}$$

Define the Q-function as,

$$Q_{H-k}(\mathbf{b}_k, \mathbf{a}_k) = (C_B(\mathbf{b}_k, \mathbf{a}_k) + \gamma \cdot E[V_{H-k-1}^*(\mathbf{b}_1|\mathbf{b}_0, \mathbf{a}_0)] \tag{3.12}$$

The optimal policy can now be expressed as,

$$\pi_k^*(\mathbf{b}_k) = \arg\min_{\mathbf{a}_k\in\mathbf{A}} (Q_{H-k}(\mathbf{b}_k, \mathbf{a}_k)) \tag{3.13}$$

Finding the optimal policy requires the calculation of the Q-values, which is in general infeasible[33].

Policy Rollout for POMDP's

Policy rollout can be used for POMDP's in certain scenarios. The technique itself is based on taking Monte Carlo samples of the expected future. For each action a $\in \mathscr{A}$ a rollout is evaluated starting from initial belief state b_0 up until horizon H. During each rollout, the process starts off with an initial belief state $\mathbf{b}_0$ and a to be evaluated action $\mathbf{a}_0$. Inside the policy rollout observations and belief states are generated based on the initial belief state and action. Specifically in policy rollout, a base policy is applied after the initial belief state until the time horizon is reached. This base policy is arbitrary and can take on many different forms depending on the type of scenario at hand. Selection of the base policy will discussed in Chapter 5.
The expected cost at each time step is summed together similarly as in 3.9. The action corresponding to the lowest cost is chosen to be the best possible action for the next time step. To obtain better results, the policy rollout for each action can be repeated multiple times. Summing all of the outcomes together and taking the average value would in result in the final cost[33].
The Q-value of the policy rollout can now be expressed as,

$$Q_{H-k}^{\pi_{base}}(\mathbf{b}_k, \mathbf{a}_k) = (C_B(\mathbf{b}_k, \mathbf{a}_k) + \gamma \cdot E[V_{H-k-1}^{\pi_{base}}(\mathbf{b}_{k+1}|\mathbf{b}_k, \mathbf{a}_k)] \tag{3.14}$$

The best possible policy is found by minimizing this expression i.e.,

$$\pi_k(\mathbf{b}_k) = \arg\min_{\mathbf{a}_k\in\mathbf{A}} (Q_{H-k}^{\pi_{base}}(\mathbf{b}_k, \mathbf{a}_k)) \tag{3.15}$$

The policy rollout considered is a one step look ahead minimization. In [34] several reasons are mentioned for using policy rollout over other approaches. One reason for using policy rollout is that the policy will be at least as good as the base policy, and in most cases will perform better than the base policy. A second reason is the fact that it can be used online, since they only need to solve a fraction of the POMDP instead of the computing the cost of each possible state. The online property allows the rollout to adapt to variations of the problem. A disadvantage of using policy rollout is when the action space is very large.

3.1.3 Radar Resource Management for a Single Sensor

The Radar Resource Management problem for a single sensor in a multi-target tracking scenario has been considered before. The material provided by Schöpe in [35] and [33] models each target according to the earlier described POMDP and solves the single sensor scenario using a dynamic budget balancing algorithm by exploiting Lagrangian Relaxation and Policy Rollout. This section aims at briefly defining the full existing solution for the single sensor case defined by [33] as the Approximately Optimal Dynamic Budget Balancing (AODB) algorithm.

AODB Algorithm

It is assumed that there are N targets to be tracked in the environment. At time step k, the budget needs to be distributed over the targets in such a way that a certain cost C is minimized while at the same time making sure that total available budget is not exceeded. The RRM problem can be defined as an optimization problem,

$$\begin{array}{ll} \min\limits_{\mathbf{a}_k} & \sum_{n=1}^{N} C(\mathbf{a}_k^n, \mathbf{s}_k^n) \\ \text{s.t.} & \sum_{n=1}^{N} B_k(\mathbf{a}_k^n) \leq b_{max} \end{array} \tag{3.16}$$

With $\mathrm{B}_k(\mathbf{a}_k^n)$ the budget assigned to each target and b_{max}, the total available budget. The budget assigned to each target in this case is related to a selected action. The cost function has been selected to be the position elements of the error-covariance of the Extended Kalman Filter (EKF).

By applying Lagrangian relaxation to problem 3.16, the problem can be decoupled into a sub-optimization problem for each target. The Lagrangian dual is defined as,

$$Z_D = \max_{\boldsymbol{\lambda}} \Big[\min_{\mathbf{a}_k} \Big(\sum_{n=1}^{N} C(\mathbf{a}_k^n, \mathbf{s}_k^n) + \lambda \cdot B_k(\mathbf{a}_k^n) \Big) - \lambda \cdot B_{max} \Big] \tag{3.17}$$

with $\lambda \in \mathbb{R}$ the Lagrangian Multiplier related to the budget constraint. The formulation of 3.17 allows the problem now to be solved for each target in parallel. After each budget allocation the Lagrangian multiplier is updated using the subgradient method. The subgradient g_k to update the Lagrange multiplier λ is defined as,

$$g_k = \frac{d}{d\lambda} \Big[\lambda \cdot (\sum_{n=1}^{N} \frac{\tau_n}{T_n} - B_{max}) \Big] = \sum_{n=1}^{N} \frac{\tau_n}{T_n} - B_{max} \tag{3.18}$$

The Lagrange multiplier is then updated according to,

$$\lambda_{k+1} = \max\{0, \lambda_k + \gamma \cdot g_k\} \tag{3.19}$$

The budget allocation for each target is based on a stochastic optimization framework. The system is modelled according to a POMDP (see Section 3.1.2) and solved using the policy rollout technique. The POMDP can be solved for finite and infinite horizons. In this case, to reduce the computational complexity a finite horizon is selected. The length of the horizon in this case represents the number of considered measurement time steps into the future.

A summary of the AODB algorithm is represented by Fig. 3.2

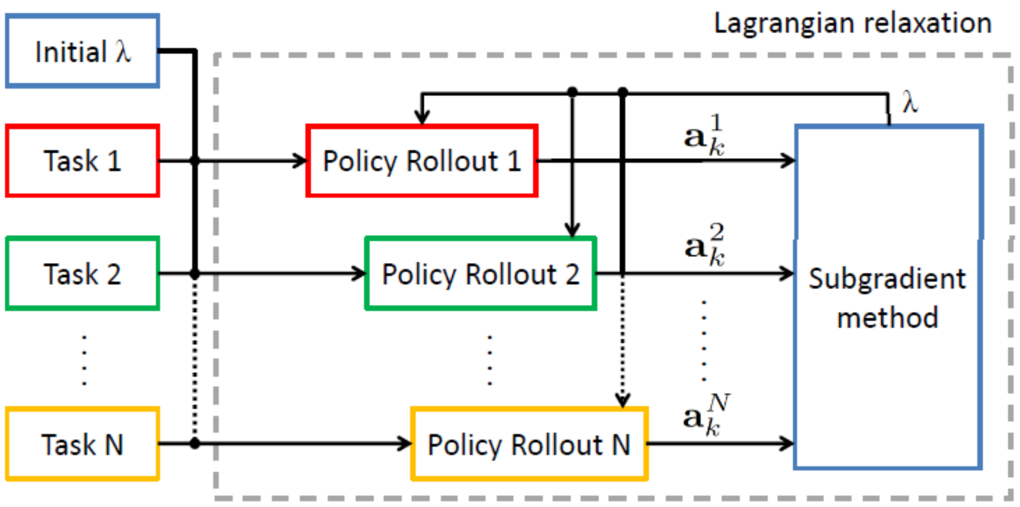


Figure 3.2. Block scheme of the AODB algorithm[33].

3.2 Data Fusion for Sensor Networks

The RRM problem considered in this thesis involves a sensor network of multiple connected radar sensors at different locations. Instead of optimizing each sensor individually, it would be beneficial to optimize the resources jointly for all sensors to obtain some global solution. This can be achieved by for example data fusion. Because there is little literature available on the RRM problem related to sensor networks, this section will mainly focus on data fusion and a tracking filter to which fusion techniques can be applied. Some additional information on alternative but closely related problems (e.g. sensor selection) in sensor networks will be given as well, due to the overlap with the RRM problem defined for this thesis.

3.2.1 Bayes' Theorem

Bayes theorem forms an essential part of all fusion methods. Using Bayes' theorem one can make certain assumptions on an object or environment which is described by a state $\mathbf{x}$, given observations $\mathbf{z}$. Assuming there is a dependency between $\mathbf{x}$ and $\mathbf{z}$, the joint probability is denoted as P($\mathbf{x}$,$\mathbf{z}$). Using the chain rule for conditional probabilities, the joint probability is related to the conditional probabilities according to the following relationship,

$$P(\mathbf{x},\mathbf{z}) = P(\mathbf{x}|\mathbf{z})P(\mathbf{z}) = P(\mathbf{z}|\mathbf{x})P(\mathbf{x}) \tag{3.20}$$

with P(x) and P(z) the prior probabilities of the state and observations respectively. P($\mathbf{z}$|$\mathbf{x}$) models the probability that an observation is made given the state. Bayes' Theorem follows by rearranging the terms slightly resulting in an expression for the state x, given observations z,

$$P(\mathbf{x}|\mathbf{z}) = \frac{P(\mathbf{z}|\mathbf{x})P(\mathbf{x})}{P(\mathbf{z})} \tag{3.21}$$

where P($\mathbf{z}$) is a normalization factor for the posterior probability.
In the case of a tracking process, the interest lies mainly in a filtering process. A filtering process tries to maintain a probabilistic model for a state that is evolving over time and which is periodically observed by a sensor.
In a probabilistic form, the goal is to find the posterior density P($\mathbf{x}_k$|$\mathbf{z}^k$, $\mathbf{x}_0$). With $\mathbf{x}^k$ the observations made up until time step k and $\mathbf{x}_0$ the initial system state.
By exploiting Bayes rule one can define the posterior density as,

$$P(\mathbf{x}_k|\mathbf{z}^k,\mathbf{x}_0) = \frac{P(\mathbf{z}_k|\mathbf{x}_k)P(\mathbf{x}_k|\mathbf{z}^{k-1},\mathbf{x}_0)}{P(\mathbf{z}_k|\mathbf{z}^{k-1})} \tag{3.22}$$

with P($\mathbf{x}_k|\mathbf{z}^{k-1}$, $\mathbf{x}_0$) the predicted probability density of the system based on observations up to time step $k-1$. This term can be rewritten in terms of a state transition model and the joint posterior from $k-1$,

$$P(\mathbf{x}_k|\mathbf{z}^{k-1}, \mathbf{x}_0) = \int P(\mathbf{x}_k|\mathbf{x}_{k-1}) \times P(\mathbf{x}_k|\mathbf{z}^{k-1}, \mathbf{x}_0) d\mathbf{x}_{k-1} \tag{3.23}$$

This equality implies that the dependency of the future state solely depends on the current state and the control input at that specific time[36, 37].

3.2.2 The (Extended) Kalman Filter

This thesis focuses on tracking scenarios. Hence, a tracking filter needs to be applied. In general any filter that calculates the posterior density can be applied. For linear systems this can be a Kalman Filter (KF), while the Extended Kalman Filter (EKF) or particle filter are applicable methods for non-linear systems. For the sake of simplicity only the KF and the EKF will be covered here.
The Kalman filter is an optimal linear filter which calculates an estimate of a system state that is evolving over time based on periodic observations made. The filter uses for both the system state $\mathbf{x}_k$ and the observations $\mathbf{z}_k$ a statistical model.
The Kalman filter is often used in a sensor network due to the explicit description of the system process and observations combined with the fact that an uncertainty measure is consistently used to measure the impact of each sensor.
The standard state-space model for estimating the system state is given by,

$$\mathbf{x}_k = \mathbf{F}_k \mathbf{x}_{k-1} + \mathbf{w}_k \tag{3.24}$$

With $\mathbf{F}_k$ the state transition model, and $\mathbf{w}_k$ the process noise assumed to be zero mean white Gaussian noise with covariance $\mathbf{Q}_k$, i.e., $\mathcal{N}(0,\, \mathbf{Q}_k)$.
The observation model is defined similarly as,

$$\mathbf{z}_k = \mathbf{H}_k \mathbf{x}_k + \mathbf{v}_k \tag{3.25}$$

with $\mathbf{H}_k$ a matrix describing the effect of the state on the observation and $\mathbf{v}_k^s$ the observation noise assumed to be independent and distributed according to $\mathcal{N}(0,\, \mathbf{R}_k)$. Where $\mathbf{R}_k$ is the covariance of the observation noise. Also, the process noise and observation noise are assumed to be uncorrelated.
The Kalman filter can now be defined as a prediction step given by,

$$\mathbf{x}_{k|k-1} = \mathbf{F}_k \mathbf{x}_{k-1|k-1} \tag{3.26}$$

$$\mathbf{P}_{k|k-1} = \mathbf{F}_k \mathbf{P}_{k-1|k-1} \mathbf{F}_k^T + \mathbf{Q}_k \tag{3.27}$$

With $\mathbf{P}$ the error covariance and $\mathbf{Q}$ the covariance of the process noise. Followed by an update step defined as follows,

$$\mathbf{S}_k = \mathbf{H}_k \mathbf{P}_{k|k-1} \mathbf{H}_k^T + \mathbf{R}_k \tag{3.28}$$

$$\mathbf{K}_k = \mathbf{P}_{k|k-1} \mathbf{H}_k^T \mathbf{S}_k^{-1} \tag{3.29}$$

$$\mathbf{P}_{k|k} = (I - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_{k|k-1} \tag{3.30}$$

$$\mathbf{x}_{k|k} = \mathbf{x}_{k|k-1} + \mathbf{K}_k (\mathbf{z}_k - \mathbf{H}_k \mathbf{x}_{k|k-1}) \tag{3.31}$$

Note that for the Kalman filter to be valid, both the state transition model and observation model need to be linear. If these models are non-linear a special form of the Kalman filter can be used called the EKF. The main differences between the EKF compared with the standard Kalman filter is the additional step of the linearization of the process and observation model using the Jacobian. The state prediction for the EKF is defined as,

$$\mathbf{x}_{k|k-1} = f(\mathbf{x}_{k-1|k-1}, \mathbf{w}_k) \tag{3.32}$$

Where f provides a non-linear mapping of the estimated system state and process noise. $\mathbf{F}_k$ and $\mathbf{H}_k$ are replaced according to,

$$\mathbf{F}_k = \nabla f_x \qquad \mathbf{H}_k = \nabla \mathbf{h}_x \tag{3.33}$$

Where ∇ the nabla operator indicates the computation of the Jacobian. Finally, the update step of the system state for EKF is defined as,

$$\mathbf{x}_{k|k} = \mathbf{x}_{k|k-1} + \mathbf{K}_k(\mathbf{z}_k - h(x_{k|k-1})) \tag{3.34}$$

In this research the Kalman filter will be utilized to tackle the radar resource management problem. In literature however, one often also refers to the information filter instead of the Kalman filter[38].
The information filter introduces two additional parameters called the information state variable y_k and information state matrix $\mathbf{Y}_k$. The relation of the additional two parameters with the system state and error covariance is defined by,

$$\mathbf{y}_{i|j} = \mathbf{P}_{i|j}^{-1}\mathbf{x}_{i|j} \tag{3.35}$$

$$\mathbf{Y}_{i|j} = \mathbf{P}_{i|j}^{-1} \tag{3.36}$$

The reason for using the information filter is the simple update step for a multi-sensor case. However, the prediction and update steps for the information filter are mathematically exactly the same as for the Kalman filter[39].

3.2.3 Sensor fusion

A sensor network consists of sensor and processing nodes, possibly co-located. The sensors are capable of observing the environment and consequently take measurements of this environment which are transmitted to a processor node to compute an estimate based on the received information.
It is assumed that each processing node is capable of broadcasting some of its information. Exploiting this system can be done either using a centralized or distributed architecture. In a centralized architecture, local processor nodes communicate information to a global processor to compute an estimate. Although this architecture provides the best possible estimate it does require more communication and has a single point of failure.
The distributed architecture uses multiple processing nodes that are capable of computing estimates offering more robustness compared to the centralized architecture[40].
The information to be communicated can be either the measurements or the corresponding state estimates resulting in measurement fusion or state estimate fusion respectively.
This section will present both measurement fusion and state estimate fusion in their most general form. For a full overview of different fusion approaches for measurement and state estimate fusion one can refer to e.g. [40, 41]. [42] focuses specifically on fusion architectures in a distributed setting.

Fusion Using State Estimates

The theory and derivations given here are loosely based on the material discussed by Kim in [43] and Bar-Shalom in [44]. Given a scenario in which M sensors need to track a single target n with corresponding state $\mathbf{x}_k$ at time step k. At each sensor m, estimates $\hat{\mathbf{x}}_{k|k}^m$ of the state of the target are maintained. Where it is assumed that the state estimates of all sensors in this case correspond to the same target. Which can be deduced for example based on some kind of hypothesis testing[44].
The optimal state estimate fusion is then defined as the conditional mean, given the track estimates in each sensor and the corresponding covariances,

$$\hat{\mathbf{x}}_{k|k}^f = E\Big[\mathbf{x}_k|\hat{\mathbf{x}}^m, \mathbf{P}^m, m = 1, 2, ...M\Big] \tag{3.37}$$

An expression of $\hat{\mathbf{x}}^f_{k|k}$ and $\mathbf{P}^f$ in terms of the individual state estimates and covariances in each sensor is obtained as follows. The posterior density f(x—x$_m$) is proportional to the joint density function,

$$f(\mathbf{x}-\hat{\mathbf{x}}_1,\mathbf{x}-\hat{\mathbf{x}}_2,...,\mathbf{x}-\hat{\mathbf{x}}_M) \tag{3.38}$$

which can be expressed as the likelihood function,

$$f \propto exp\Big[-\frac{1}{2}(\mathbf{x}-\hat{\mathbf{x}}_1,\mathbf{x}-\hat{\mathbf{x}}_2,...,\mathbf{x}-\hat{\mathbf{x}}_M)^T\Sigma^{-1}(\mathbf{x}-\hat{\mathbf{x}}_1,\mathbf{x}-\hat{\mathbf{x}}_2,...,\mathbf{x}-\hat{\mathbf{x}}_M)\Big] \tag{3.39}$$

Under the assumption that the prior, f($\mathbf{x}$) is uniformly distributed and the posterior density is assumed to be Gaussian distributed.
Define e = [I,I,...,I]^T, $\Sigma = E\Big[(\mathbf{x}-\hat{\mathbf{x}}_i)(\mathbf{x}-\hat{\mathbf{x}}_j)^T\Big]$ for $\forall i,j \in M$ and $\mu = (\mathbf{x}-\hat{\mathbf{x}}_1,\mathbf{x}-\hat{\mathbf{x}}_2,...,\mathbf{x}-\hat{\mathbf{x}}_M)^T$. Then the joint density function f can be rewritten into matrix form according to,

$$f(\mu - e\mathbf{x}) \propto exp[-\frac{1}{2}(\mu - e\mathbf{x})^T\Sigma^{-1}(\mu - e\mathbf{x})] \tag{3.40}$$

A maximum likelihood estimate is obtained by maximizing the likelihood function resulting in,

$$\frac{d}{dx}\Big(-\frac{1}{2}(\mu - e\mathbf{x})^T\Sigma^{-1}(\mu - e\mathbf{x})\Big) = -2e^T\Sigma^{-1}(\mu - e\mathbf{x}) = 0 \tag{3.41}$$

Solving this expression for the system state $\mathbf{x}$ gives,

$$\hat{\mathbf{x}}^f = (e^T\Sigma^{-1}e)^{-1}(e^T\Sigma^{-1}\mu) \tag{3.42}$$

and a corresponding fused covariance of,

$$\mathbf{P}_f = (e^T\Sigma^{-1}e)^{-1} \tag{3.43}$$

Note that 3.42 and 3.43 can be used both for a centralized and distributed architecture.

Fusion Using Measurements

The second approach could be to fuse the measurements instead of the state estimates. If it is assumed that each sensor can broadcast their measurements to other sensor nodes then measurement fusion can be defined as a recursive update scheme. If it is assumed that the measurements of all sensors are available at the same time instant, fused estimates of the state and covariance are computed according to Algorithm 2. With $\mathbf{H}$ an observation

Algorithm 2: State and covariance measurement update scheme

```
   Input: $\mathbf{P}^{f_n}_{k|k-1} \in \mathbb{R}^{4x4}$, $\mathbf{s}^{f_n}_{k|k-1} \in \mathbb{R}^{4x1}$ $\mathbf{R}$, $\mathbf{H}$, $\mathbf{h}$
   Output: $\mathbf{P}^{f_n}_{k|k} \in \mathbb{R}^{4x4}$, $\mathbf{s}^{f_n}_{k|k} \in \mathbb{R}^{4x1}$
 1 $\mathbf{P}_n = \mathbf{P}^{f_n}_{k|k-1}$
 2 $\mathbf{s}_n = \mathbf{s}^{f_n}_{k|k-1}$
 3 $m = 1$
 4 while $m < M$ do
 5     $\mathbf{P}_n = \text{update}_{covariance}(\mathbf{R}_m, \mathbf{H}_m, \mathbf{P}_n)$
 6     $\mathbf{s}_n = \text{update}_{state}(\mathbf{R}_m, \mathbf{H}_m, \mathbf{z}_m, \mathbf{h}_m, \mathbf{P}_n, \mathbf{s}_n)$
 7     $m = m + 1$
 8 end
 9 $\mathbf{P}^{f_n}_{k|k} = \mathbf{P}$
10 $\mathbf{s}^{f_n}_{k|k} = \mathbf{s}_n$
11 return $\mathbf{P}^{f_n}_{k|k}, \mathbf{s}^{f_n}_{k|k}$
```

matrix, $\mathbf{h}$ a measurement transformation function and $\mathbf{R}$ the covariance of the observation

noise. The superscript f_n indicates the fused data related to target n. Each iteration, the state estimate of target n is updated based on an observation $\mathbf{z}_m$ made by sensor m and a corresponding estimated error covariance is computed. The resulting estimate is the optimal estimate given all the measurements at the node received up to that time[40].
The fused estimates are then used to compute a prediction of the error covariance.
This form of fusion is less complex to implement compared to state estimate fusion due the fact that measurement fusion does not have to cope with dependent estimation errors[45].

3.2.4 Closely Related Problems

Many previous approaches related to resource management in sensor networks focus on the topic of sensor selection. More specifically, the problem focuses on selecting a sensor or a subset of sensors such that a certain cost is minimized subject to some constraint (e.g. communication or energy constraint) in both myopic and non-myopic settings. For example in [46], the sensor selection problem is formulated as an optimization problem in which the state estimation Mean Square Error (MSE) is minimized.
In [38] optimal sensor activations are searched for by minimizing the trace of the fisher information matrix subject to energy constraints. Some other related literature on the topic of sensor selection can be found in e.g. [47], [48] and [49].
The problem formulation of sensor selection is closely related to the topic of this thesis and potentially some material of a sensor selection solution can be utilized for the resource management problem. However, the final goal of sensor selection and resource management are fundamentally different. The sensor selection problem aims at finding optimal (binary) sensor activations whereas SRM in a sensor network searches for maximum operational capacity of the entire sensor network and can be used with continuous actions.
Some alternative solutions to RRM in sensor networks can be found in for example [50] and [51].
In [50] a myopic RRM solution for single target tracking problems using sensor networks is proposed. Another RRM problem approach to a network scenario has been presented by [51] which aims at decreasing the sensing time of the individual sensors while keeping the sensing performance at a desired level.

3.3 Preliminary Conclusions

Based on the given material in this chapter several conclusions can be made regarding the existing approaches for solving the RRM problem in a multi-sensor network.
First of all, the problems in general are considered to be optimization problems that optimize some objective function. Up until this point the selection of cost function is still under investigation. Either the task-based, information-driven or the risk-based approach are suitable. However, since the measures are mainly task based, the task based approach appears to be the most straightforward to implement.
Parallelization is desired to reduce the complexity of the optimization problem. Parallelization of the problem into multiple sub-optimization problems and solving it can be achieved using multiple procedures. Two possible approaches were mentioned in this thesis. The first approach utilizes LR to decouple the problem and generates a solution using the subgradient method. The second approach is to use ADMM instead. ADMM offers a general framework and is relatively simple to implement. However, the convergence rate is poor compared to LR. Making the combination of LR and the subgradient method the more desired approach.

Solving the problem using ADMM or with the subgradient method solves the problem in a myopic fashion. By using stochastic optimization the problem can be solved non-myopically by looking into the expected future instead of looking a single time step ahead. Although there are multiple ways of stochastic optimization techniques, the choice has been made to use policy rollout mainly due its simple implementation.
To extend the RRM problem to a multi-sensor network, data fusion can be applied. Either

state estimate or measurement fusion can be applied. Measurement fusion is less prone to dependent estimation errors, making it the preferred fusion method over state estimate fusion.

Chapter 4

Algorithms for Radar Resource Management in a Sensor Network

This chapter aims at introducing the newly developed algorithms for a RRM problem in a sensor network for performing multiple tasks jointly. This chapter deals with the optimization of the sensing resources, and the use of the term measurement refers solely to the internal simulation of the expected future in the POMDP.

Section 4.1 will provide a detailed problem description. Implementations to solve the problem will be considered in Section 4.2. Finally, to indicate the relevance and applicability of the developed algorithm an alternative use case will be presented in Section 4.3.

4.1 Detailed Problem Description

Define a sensor network consisting of M sensors where each sensor $m \in M$ in the sensor network behaves according to a MFR system. Every sensor in the sensor network has a maximum resource budget b_{max} available. Portions of the total sensor budget can be allocated to a task $n \in N$, where N is the total number of tasks that need to be executed. Define the portion of the total sensor budget $\mathrm{b}_{m,n}$ allocated to task n by sensor m as a function of the actions $\mathbf{a}_k^{m,n}$ taken at time instant k.

For the sensors under consideration, the resource budget is often not sufficient to support all tasks that need to be executed. Consequently, the sensors often operate at their resource limit i.e., the total number of tasks per sensor requires more sensor budget than available. The SRM problem in general is considered as a scheduling problem, where the goal is to allocate the budget over the different tasks.

More specifically, the scheduling problem can be modeled according to a constrained optimization framework in which the optimal actions need to be found such that a certain cost $\mathbf{C}$ is minimized while at the same time the maximum resource budget b_m^{Max} of each sensor is not exceeded.

At time instant k, the optimization problem for N tasks and M sensors is formulated as,

$$\begin{aligned} \min_{\mathbf{A}} \quad & \mathbf{1}^T\mathbf{C} \\ \text{s.t.} \quad & \underbrace{\begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1N} \\ b_{21} & b_{22} & & \\ \vdots & & \ddots & \\ b_{M1} & & & b_{MN} \end{bmatrix}}_{\mathbf{B}(\mathbf{A})} \cdot \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \leq \underbrace{\begin{bmatrix} b_1^{max} \\ b_2^{max} \\ \vdots \\ b_M^{max} \end{bmatrix}}_{\mathbf{b}_{max}} \end{aligned} \tag{4.1}$$

Where,

$$\mathbf{C} = \begin{bmatrix} \mathbf{C}_1 & \mathbf{C}_2 & \cdots & \mathbf{C}_N \end{bmatrix}^T \tag{4.2}$$

represents the cost related to all tasks, $\mathbf{1} \in \mathbb{R}^{p\times1}$ a vector of all ones and $\mathrm{b}_{m,n}$ is a representation of the budget spend by sensor m on task n. The individual budgets $\mathrm{b}_{m,n}$ inside the resource budget matrix $\mathbf{B}(\mathbf{A})$ represent a percentage of the maximum budget $\mathbf{b}_{max}$ available to each sensor. The optimization variable $\mathbf{A} = [\mathbf{a}^{1,n}, \cdots, \mathbf{a}^{M,n}]$ is a stacked vector representing the actions of all sensors.
In [35] and [33] a method was proposed for solving the RRM problem for a single sensor. This thesis aims at extending the solution to a multi-sensor case. Hence, in the remainder of this thesis, the focus will be on solving a RRM problem.

RRM problem definition

In a RRM problem specifically, each task n is considered to be a target that moves around in some environment and is being tracked by M MFR systems using a tracking filter such as the Kalman filter. In a two-dimensional scenario, every target n can be modelled by a state that describes the current position and velocity in a Cartesian coordinate system at time step k,

$$\mathbf{s}_k^n = [x^n, y^n, \dot{x}^n, \dot{y}^n] \tag{4.3}$$

with $\dot{x}^n$ and $\dot{y}^n$ the velocities in x and y direction respectively. The state of target n at time step $k+1$ can be expressed in terms of the state at time step k according to,

$$\mathbf{s}_{k+1}^n = f_{k+1}(\mathbf{s}_k^n, \mathbf{w}_k^n) \tag{4.4}$$

with $\mathbf{w}_k^n$ the process noise and f_{k+1} some state transition function.
Each sensor takes a measurement of the state at every discrete time step k. For time step k an observation related to target n and sensor m can be characterized as,

$$\mathbf{z}_k^{m,n} = \mathbf{h}(\mathbf{s}_k^n, \mathbf{v}_k^{m,n}, \mathbf{a}_k^{m,n}) \tag{4.5}$$

with $\mathbf{h}$ a measurement function that depends on the state, measurement noise $\mathbf{v}_k$ and the sensor action $\mathbf{a}_k^{m,n}$.

For the remainder of the thesis, the tracking process is based on the sensing information of all sensors. It is assumed that all sensors produce independent measurements which are fused to a global estimate per target using Algorithm 2. This chapter deals with the optimization of the sensing resources, and the use of the term measurement here solely refers to the internal simulation of the expected future of the POMDP.

4.2 Proposed Solutions for the RRM Problem

This section aims at deriving different approaches to tackle the problem described in Section 4.1. The focus here is on describing what happens during the optimization process. The quantitative analysis will be done in chapter 5.

Three different implementations will be considered. These are an independent implementations (4.2.3), a centralized implementations (4.2.4) and a distributed implementation (4.2.5). Prior to the description of the three implementations, two relevant frameworks will be given related to LR and Policy Rollout.

4.2.1 Distribution of Sensor Budgets Using Lagrangian Relaxation

LR is used to include the constraints of problem 4.1 into the cost function similarly as with problem 3.16. This allows the original optimization problem to be decoupled into multiple problems that can be solved in parallel. Define the Lagrangian Dual Problem (LDP) of problem 4.1 as,

$$Z_D = \max_{\boldsymbol{\lambda}} \Big[\min_{\mathbf{A}} \sum_{n=1}^{N} \Big(\mathbf{C}_n + \sum_{m=1}^{M} \lambda_m \cdot b_{m,n} \Big) - \boldsymbol{\lambda}^T \cdot \mathbf{b}_{max} \Big] \tag{4.6}$$

with $\boldsymbol{\lambda} \in \mathbb{R}^{M\times 1}$, the Lagrangian multiplier related to the inequality constraint. Due to the summation in the LDP, the problem can be solved myopically in parallel during each LR iteration. The separate sub-problems are connected through the Lagrangian multiplier, which is updated using the subgradient method,

$$\boldsymbol{\lambda}_{k+1} = \max\{0, \boldsymbol{\lambda}_k + \gamma \cdot \mathbf{g}_k\} \tag{4.7}$$

It is assumed that the cost is based on a one step ahead minimization, hence the solution to this problem would be myopic. In the next section, the policy rollout technique will be used to indicate how the problem can be solved non-myopically.

4.2.2 Finding Approximately Optimal Actions Using Policy Rollout

The system under consideration is modeled according to a POMDP. Inside the POMDP, the tracking process as described in Section 3.2.2 is simulated. As mentioned in Section 3.1.2, the POMDP describes a Markov Decision Process (MDP) in which the state cannot be observed directly, instead an observation is generated that computes a probability distribution over the possible states called the belief state $\mathbf{b}$. Using the belief state and underlying knowledge of the MDP, the POMDP allows the problem to be solved non-myopically by taking into account the expected future.

Similarly as in Section 3.1.3, the actions for each sensor are found using policy rollout for POMDP's. For each action $\mathbf{a} \in \mathscr{A}$, a rollout is evaluated starting from initial belief state $\mathbf{b}_0$ up until horizon H.

4.2.3 Independent Implementation

The most straightforward implementation to the RRM problem in a sensor network is achieved by assuming that each sensor operates independently during the optimization process i.e., the independent implementation applies multi-sensor tracking as explained in section 3.2.3, but is applying the RRM algorithm from [33, 35] as explained in section 3.1.3 for each sensor individually.

Since each sensor operates independently, the cost $\mathbf{C}_n$ related to task n can be decomposed into a summation of costs from multiple sensors,

$$\mathbf{C}_n = C_{1,n} + C_{2,n} + \cdots + C_{M,n} \tag{4.8}$$

Where the individual cost related to sensor m and task n is defined as a function of the action $\mathbf{a}_k^{m,n}$ and state $\mathbf{s}_k^n$,

$$C_{m,n} = (\mathbf{a}_k^{m,n}, \mathbf{s}_k^n) \tag{4.9}$$

Applying LR to problem 4.1 using this definition of the cost results in the following LDP,

$$Z_D = \max_{\boldsymbol{\lambda}} \left[\min_{\mathbf{A}} \left(\sum_{m=1}^{M} \sum_{n=1}^{N} C_{m,n} + \lambda_m \cdot b_{m,n} \right) - \boldsymbol{\lambda}^T \cdot \mathbf{b}_{max} \right] \tag{4.10}$$

The problem can be decomposed into a sub-optimization per task per sensor. Per task and sensor a policy rollout is evaluated to search for the best possible actions. Since each sensor

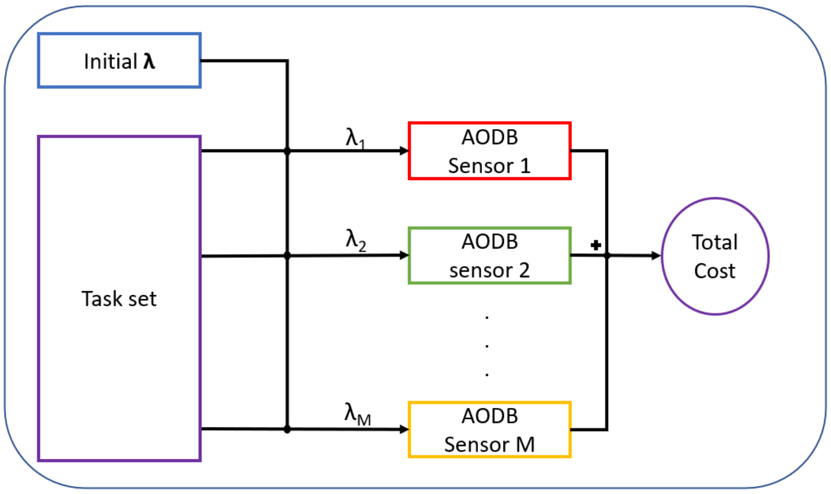


Figure 4.1. Block diagram of a Multi-sensor RRM solution when there is no communication between sensors. Each sensor computes an independent solution based on the AODB algorithm.

operates independently, the AODB algorithm as presented in Section 3.1.3 can be applied to each sensor individually. The outputs of each AODB block are summed together to compute the total cost of the algorithm. A block diagram of this implementation is given in Fig. 4.1.

The independent implementation of AODB to a multi-sensor case is valid. It is not necessarily the optimal way of solving this problem because the information sharing property of sensors is not taken into account. Hence, further improvement can be made to the multi-sensor solution.

4.2.4 Centralized Implementation

The proposed solution is based on a centralized fusion approach, which is considered as the most optimal way of utilizing measurement fusion. A central processing node is used for allocating budgets over the sensors. Consequently, the representation of the cost $\mathbf{C}_n$ related to task n needs to incorporate the actions of multiple sensors i.e.,

$$\mathbf{C}_n = C(\mathbf{a}_k^{1,n}, \mathbf{a}_k^{2,n}, \cdots, \mathbf{a}_k^{M,n}, \mathbf{s}_k^n) \tag{4.11}$$

Applying LR to problem 4.1 using this definition of the cost function results in the following LDP,

$$Z_D = \max_{\boldsymbol{\lambda}} \left[\min_{\mathbf{A}} \left(\sum_{n=1}^{N} C(\mathbf{a}_k^{1,n}, \mathbf{a}_k^{2,n}, \cdots, \mathbf{a}_k^{M,n}, \mathbf{s}_k^n) + \sum_{n=1}^{N} \sum_{m=1}^{M} \lambda_m \cdot b_{m,n} \right) - \boldsymbol{\lambda}^T \cdot \mathbf{b}_{max} \right] \tag{4.12}$$

The problem can be decomposed into N parallel sub-optimization problems and the optimal actions for M sensors related to the n'th task are computed in the central processing node using a global policy rollout.

The centralized implementation utilizes a global policy per task to explore the actions of multiple sensors. For each evaluation of a global policy rollout tracking updates are computed using the EKF.
The choice has been made to use a form of measurement fusion instead of state estimate fusion since it is straightforward to implement and does not have to deal with dependent estimation errors[45]. Hence, Algorithm 2 as described in Section 3.2.3 is utilized.
Since the global policy rollout explores the actions of multiple sensors, the action space will be $\mathbf{g} \times \mathbf{a}^n$, where $\mathbf{g}$ has a length of d^{ML} with d, the number of actions each individual sensor can take related to target n, L the number of optimization variables per sensor and $\mathbf{a}^n = [\mathbf{a}^{1,n}, \cdots, \mathbf{a}^{M,n}]$ the optimization variables for each sensor (see Table I). Hence, increasing the number of sensors will result in an exponential increase of the action space. Because the central processing node has access to all required information

TABLE I. Action space required for the centralized implementation for target n. M is the number of sensors, L is the number of optimization variables per sensor (assumed to be 1 here).

Actions	**Sensor 1**	**Sensor 2**	$\cdots$	**Sensor M**
$\mathbf{g}(1)$	$\mathrm{a}^{1,n}(1)$	$\mathrm{a}^{2,n}(1)$	$\cdots$	$\mathrm{a}^{M,n}(1)$
$\mathbf{g}(2)$	$\mathrm{a}^{1,n}(2)$	$\mathrm{a}^{2,n}(1)$	$\cdots$	$\mathrm{a}^{M,n}(1)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\mathbf{g}(\mathrm{d}^1)$	$\mathrm{a}^{1,n}(d)$	$\mathrm{a}^{2,n}(1)$	$\cdots$	$\mathrm{a}^{M,n}(1)$
$\mathbf{g}(\mathrm{d}^1+1)$	$\mathrm{a}^{1,n}(1)$	$\mathrm{a}^{2,n}(2)$	$\cdots$	$\mathrm{a}^{M,n}(1)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\mathbf{g}(\mathrm{d}^{2\cdot 1})$	$\mathrm{a}^{1,n}(1)$	$\mathrm{a}^{2,n}(d)$	$\cdots$	$\mathrm{a}^{M,n}(1)$
$\mathbf{g}(\mathrm{d}^{2\cdot 1}+1)$	$\mathrm{a}^{1,n}(1)$	$\mathrm{a}^{2,n}(1)$	$\cdots$	$\mathrm{a}^{M,n}(2)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\mathbf{g}(d^{M\cdot 1})$	$\mathrm{a}^{1,n}(1)$	$\mathrm{a}^{2,n}(1)$	$\cdots$	$\mathrm{a}^{M,n}(d)$

(e.g. measurements) the solution to this problem is regarded optimal, but becomes approximately optimal by applying the policy rollout. Also, for a single sensor, the approach given here will reduce to the independent implementation since no measurements can be fused together.

4.2.5 Distributed Implementation

The centralized approach for solving the multi-sensor scenario introduces a heavy computational load on the policy rollout due to the exponential increase of the action space. As a result of this, a practical distributed alternative is proposed in which the information of each sensor is shared amongst the sensors instead of transmitting it to a central node.
Now per sensor and per task a policy rollout is computed. Hence, each individual policy rollout only needs to explore the action space related to a single task and a single sensor. Now each sensor functions as a processing node to compute fused estimates and to find the best possible actions related to that specific sensor.
Each local processor is responsible for computing budget allocations related to the corresponding sensor. Hence, the cost $\mathbf{C}_n$ related to task n can be decomposed into a summation of costs from multiple sensors,

$$\mathbf{C}_n = C_{1,n} + C_{2,n} + \cdots + C_{M,n} \tag{4.13}$$

Where the individual cost related to sensor m and task n is defined as,

$$C_{m,n} = (\mathbf{a}_k^{m,n}, I, \mathbf{s}_k^n) \tag{4.14}$$

Where I is a representation of the information received from all other sensors. Applying LR to problem 4.1 using the cost as defined above results in the following LDP,

$$Z_D = \max_{\boldsymbol{\lambda}} \left[\min_{\mathbf{A}} \left(\sum_{m=1}^{M} \sum_{n=1}^{N} C_{m,n} + \lambda_m \cdot b_{m,n} \right) - \boldsymbol{\lambda}^T \cdot \mathbf{b}_{max} \right] \tag{4.15}$$

The problem is decomposed into a sub-optimization problem per task per sensor similarly as with the independent implementation while at the same time in each sub-problem the information of the other sensors is shared. By doing so, each individual policy rollout only needs to explore the action space related to a single task and a single sensor similarly as with the independent extension while at the same time the information sharing ability is maintained.

As the policy rollout is making predictions of the expected future for a single sensor, it does not have access to the optimized actions of all other sensors during the resource allocation calculation using LR. To maintain a similar performance as the centralized implementation, at the beginning of each policy rollout for a sensor, the last known actions of the other sensors are used as input. Hence, the information term I is defined as the last known actions of the other sensors.
Thus, during a policy rollout, the actions of a single sensor and a single task are explored while the other sensors are assumed to execute the same action. Giving the sensor access to its own generated measurements measurements related.

A disadvantage of sharing the information between sensors is the additional communication overhead. However, one would expect that the required computational resources for the additional communication are significantly less compared to the computational resources required for computing a global policy rollout in the centralized implementation. Hence, the expectation is that the distributed implementation is considered the best possible implementation compared to the other two implementations. It should obtain a similar cost as the centralized implementation using significantly less computational resources. The distributed fusion implementation can be combined with the AODB algorithm. Fig.

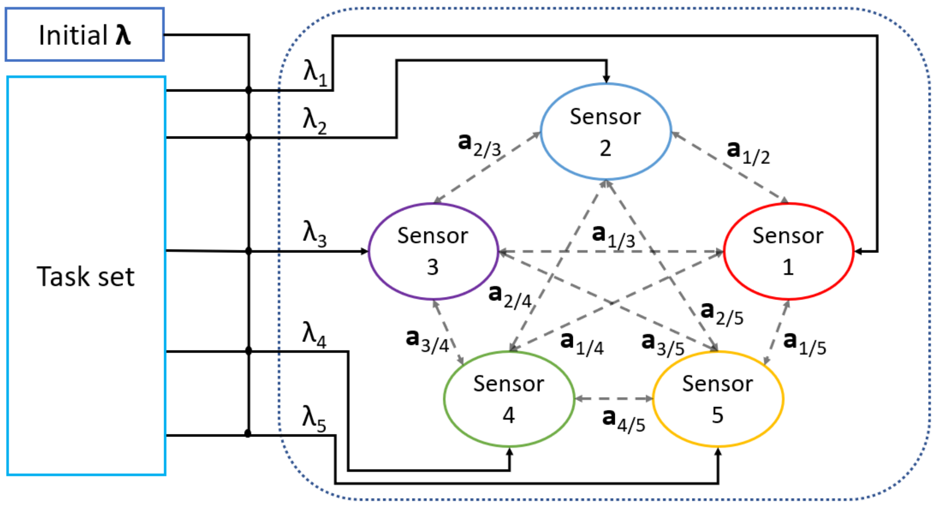


Figure 4.2. Block scheme of the distributed implementation for a scenario with five sensors using the AODB algorithm.

4.2 shows an example involving five tasks. An initial $\boldsymbol{\lambda}$ is selected and the task set is given as input for all sensors. For each sensor the AODB algorithm is computed resulting in an action for the respective sensor. Consequently, the resulting actions from each policy rollout are communicated to all other sensors. The policy rollout during the next time step refines the action of the sensor by incorporating the actions of the other sensor.

4.3 Alternative Use Case

To indicate the general applicability of the proposed solutions an additional example is introduced related to a power distribution network. The goal here is to give a high level approach of the distributed implementation in a field other then radar systems. For this a smart grid is considered (see Fig. 4.3), a distributed network that connects several energy hubs (e.g. solar or wind energy) to a power grid that provides customers such as homes and businesses with electricity. In a smart grid, it is assumed that each energy hub is capable of exchanging information with the other hubs. The ability to exchange information in this case potentially improves the efficiency and reliability regarding the power distribution.
Each energy hub in the network has a certain amount of power available which it needs to allocate over the customers as efficiently and reliable as possible, while at the same time overloads should be prevented.
In theory, this system can be modeled into an optimization problem that is similar to problem 4.1. For example, a cost function **C** could be designed that takes into account the users needs while taking the maximum energy per power hub into account. By looking ahead into the expected future an optimal policy can be found.
Via smart sensors, the network takes measurements at the customer which are used to update the energy allocated by each energy hub.
Hence, this problem could be solved by using either the centralized or distributed implementation as discussed in the previous section.

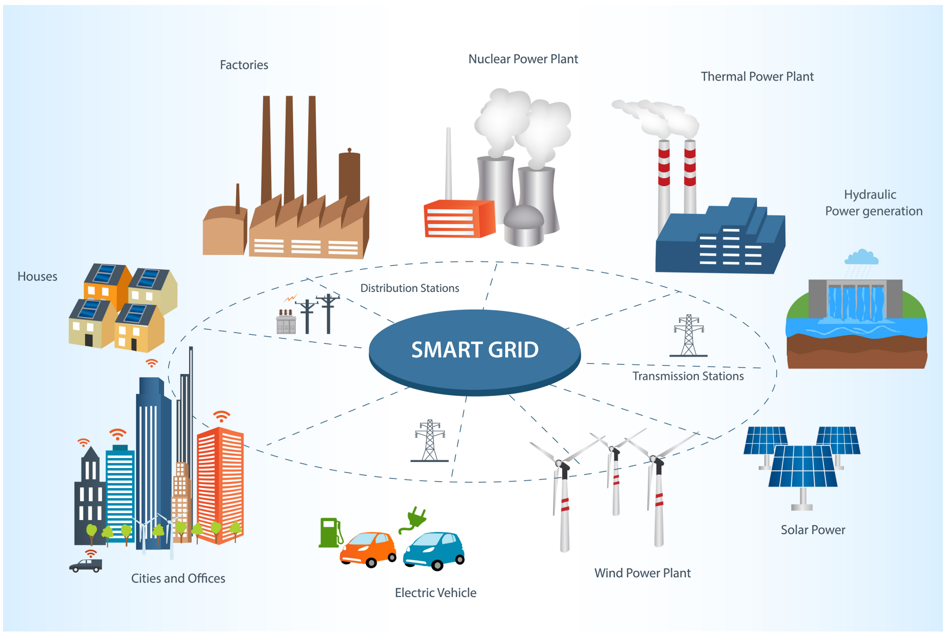


Figure 4.3. Schematic of a smart grid connecting industrial and renewable energies with homes and other businesses to show how the proposed solution can be applied to networks other than radar system networks[52].

Chapter 5

Analysis

In Chapter 4, three implementations to the RRM problem were introduced. This chapter aims at showing the functioning of all three implementations and to verify the distributed implementation as the best possible approach. Multiple two-dimensional radar tracking scenarios will be considered in an environment with multiple targets and stationary sensors. Unless specified otherwise, in the following simulations it is assumed that the revisit time T i.e., the time between consecutive measurements is fixed at one second and the optimization problem focuses only on the dwell time τ i.e., time spent on each target, with $\tau \in [0,1]$. Section 5.1 will introduce the assumptions made on the radar scenarios. A set of general simulation parameter will be given in Section 5.2. Finally, in Section 5.3 results of different simulation scenarios will be provided.

5.1 Modelling

Since the analysis of the algorithms is done in a two dimensional tracking scenario, several assumptions need to be made on the radar system and dynamics of the targets.

Target Dynamical Model

It is assumed that each target moves according to a constant velocity model. Using 4.4, the transition from state k to k+1 can now be described according to,

$$\mathbf{s}_{k+1}^{n} = \mathbf{F}\mathbf{s}_{k}^{n} + \mathbf{w}_{k}^{n} \tag{5.1}$$

with $\mathbf{w}_k^n$ defined as the process noise and state transition model $\mathbf{F}$ given by,

$$\mathbf{F} = \begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.2}$$

The corresponding maneuverability noise covariance $\mathbf{Q}_{w,n}$ for target n is in this case defined as,

$$\mathbf{Q}_{w,k} = \begin{bmatrix} \frac{T^2}{2} & 0 \\ 0 & \frac{T^2}{2} \\ T & 0 \\ 0 & T \end{bmatrix} \begin{bmatrix} \frac{T^2}{2} & 0 & T & 0 \\ 0 & \frac{T^2}{2} & 0 & T \end{bmatrix} \cdot \sigma_{w,k}^2 = \begin{bmatrix} \frac{T^4}{2} & 0 & \frac{T^3}{2} & 0 \\ 0 & \frac{T^4}{2} & 0 & \frac{T^3}{2} \\ \frac{T^3}{2} & 0 & \frac{T^2}{2} & 0 \\ 0 & \frac{T^3}{2} & 0 & \frac{T^2}{2} \end{bmatrix} \cdot \sigma_{w,k}^2 \tag{5.3}$$

with $\sigma_{w,k}^2$ the maneuverability noise variance of a single target.

Observation model

The sensor generates an observation that describes the range and angle with respect to the target. Due to the non-linear relationship between measurements and states, the EKF (see Section 3.2.2) is used to compute the system state.
The tracking of targets during the simulation is realized in cartesian coordinates. Using a measurement transformation function **h**, measurements are generated in range and angle. The observation **z** received at each sensor is given by,

$$\mathbf{z}_k^{m,n} = \mathbf{h}_k^{m,n}(\mathbf{s}_k^n) + \mathbf{v}_k^{m,n} \tag{5.4}$$

with $\mathbf{v}_k^{m,n}$, the measurement noise for target n. Note that the noise related to range and angle are assumed to be independent of each other, i.e.,

$$\mathbf{v}_k^{m,n} = [v_r^{m,n}, v_\theta^{m,n}]^T \tag{5.5}$$

with corresponding variances σ_r^2 and σ_θ^2. where $\mathbf{h}_k^{m,n}(\mathbf{s}_k^n)$ represents the measurement transformation function which transforms the Cartesian measurements into polar measurements defined as,

$$\mathbf{h}_k^{m,n}(\mathbf{s}_k^n) = \begin{bmatrix} \sqrt{(x_k^n - x'_m)^2 + (y_k^n - y'_m)^2} \\ \text{atan2}(y_k^n - y'_m, x_k^n - x'_m) \end{bmatrix} \tag{5.6}$$

with x$'_m$ and y$'_m$ the location of sensor m in Cartesian coordinates respectively.
The observation model $\mathbf{H}_k^{m,n} \in \mathbb{R}^{2x4}$ for target n and sensor m is defined as the Jacobian of the measurement transformation function **h** evaluated at the current state of target n,

$$\mathbf{H}_k^{m,n} = \frac{\delta \mathbf{h}^m}{\delta \mathbf{s}}\Big|_{\mathbf{s}_k^n} \tag{5.7}$$

SNR Model

Computation of the Signal-to-Noise Ratio (SNR) is done according to 5.8, which is based on the theory provided by Koch in [53].

$$SNR_k = SNR_0 \cdot \frac{\sigma_n}{\sigma_0} \cdot \frac{\tau_n}{\tau_0} \cdot \left(\frac{r_n}{r_0}\right)^4 \cdot \exp(-2\Delta\alpha) \tag{5.8}$$

with SNR$_0$, the SNR for a reference target, σ_n the Radar Cross Section (RCS) of target n, τ_n the dwell time of target n and r$_n$ the actual range of target n at time step k. The values in the denominator are corresponding to a RCS, dwell time and range of a reference target. The term in the exponential is called the relative beam positioning error, which is defined in this case according to,

$$\Delta\alpha = \frac{(\theta_{m,n} - \hat{\theta}_{m,n})^2}{\Gamma^2} \tag{5.9}$$

with $\theta_{m,n}$ the real target angle with respect to sensor m, $\hat{\theta}_{m,n}$ the predicted target angle in azimuth and Γ, the one sided beam-width in azimuth. In the rest of this section it is assumed that there is no beam positioning error i.e., $\Delta\alpha = 0$
The variance in range and angle related to the measurement noise can now be defined as,

$$(\sigma_{r/\theta}^{m,n})^2 = \frac{(\sigma_{r/\theta}^{m,0})^2}{SNR_k} \tag{5.10}$$

The corresponding measurement covariance matrix is, due to the independent measurements, defined as the following diagonal matrix,

$$R_k^{m,n} = \begin{bmatrix} (\sigma_r^{m,n})^2 & 0 \\ 0 & (\sigma_\theta^{m,n})^2 \end{bmatrix} \tag{5.11}$$

σ_r^2 and σ_θ^2 are computed based on a SNR value and measurement noise variances($\sigma_{r,0}^2, \sigma_{\theta,0}^2$) that are related to a reference target with parameters defined according to Table I.

TABLE I. Parameters of reference measurement

Parameter	**Value**
Reference SNR (SNR_0)	1
Reference RCS (ζ_0)	$10\,\mathrm{m}^2$
Reference dwell time (τ_0)	$1\,\mathrm{s}$
Reference range (r_0)	$50\,\mathrm{km}$

5.2 Assumed Radar Scenario

This section describes the prerequisites for evaluating different radar tracking scenarios. A set of general parameters are given followed by a set of parameters related to the system and the target. Prior to this a small note will be made on how the optimization problem is evaluated and on the initialization of the algorithms. Unless specified otherwise, the parameters described here are the parameters used throughout all different scenarios.

Cost function & Constraint

In both the centralized and distributed approach the cost ($\mathbf{C}_n$), related to target n is based on the predicted error covariance at time step $k+1$. Define the current predicted error covariance at time step k as,

$$\mathbf{P}^*_{k+1|k} = \mathbf{F}\mathbf{P}^*_{k|k}\mathbf{F}^T + \mathbf{Q}_k \tag{5.12}$$

The * indicates that the error covariance can be either the fused predicted error covariance computed in the central processing node or the predicted error covariance computed locally in sensor m depending on the chosen approach.
The cost is defined to be the trace of the positional elements of the error covariance,

$$\mathbf{C}_n = \begin{bmatrix}1 & 1 & 0 & 0\end{bmatrix} \cdot diag(\mathbf{P}^*_{k+1|k}) \tag{5.13}$$

The formulation of the cost for the independent, centralized and distributed approach are formulated slightly differently. A normalization factor of $\frac{1}{N}$ is added for the centralized approach since this approach already takes into account the number of sensors in the cost $\mathbf{C}_n$. Similarly, for the independent and distributed approach a normalization term of $\frac{1}{MN}$ is added.

The individual budgets $\mathrm{b}_{m,n}$ are defined to a be a ratio of the dwell time over the revisit time per sensor per target,

$$b_{m,n} = \frac{\tau_{m,n}}{T} \tag{5.14}$$

Since the revisit time is fixed at one second and the revisit time can take one values between 0 and 1 second, the individual budget $\mathrm{b}_{m,n}$ represents a percentage of the total budget used by the sensor.

Initialization

Prior to computing budget allocations an initialization process is defined. During this initialization step the error covariance is initialized as a square diagonal matrix $\mathbf{P}^*_{k+1|k} = diag(100, 100, 100, 100)$. The targets are tracked using the EKF using Algorithm 2 over multiple simulation steps during which the predicted state and error covariance are updated to guarantee reasonable initial values of the error covariance as soon as the budget allocation algorithm process has started.
Note that the initial error covariances of the centralized and distributed approach are set to be the same.

General Simulation parameters

Table II shows the general simulation parameters used for the simulations. The maximum budget for each sensor is set to be 1. The budget allocation is recalculated every 20 seconds, and there is a total of 10 budget updates. In between budget updates, measurements are taken using the current allocated budgets.
As discussed in Section 3.1.2, the policy rollout uses a belief state which is used after the initial belief state until the time horizon is reached. For simplicity, the applied base policy is defined as the evaluated action at each step in the policy rollout ($\pi_{base} = \mathbf{a}_k$). The policy rollout has a horizon length of 15 time steps. For each policy selection four possible rollouts are computed and the result is defined as the average of those four outcomes.
The action to be optimized is defined to be the dwell time (τ). The actions are selected from a one dimensional discrete actions space $\mathscr{A}$. The discretization for the dwell time ($\Delta\tau$) is defined to be 0.01 seconds. The revisit time T is fixed at 1 second.

TABLE II. General Simulation Parameters

Parameter	**Value**
Maximum budget B_{max}	1
Budget update t_B	20 s
Number of budget updates	10
Beam positional error $\Delta\alpha$	0
Probability of detection	1
Step size subgradient γ	100

Parameter	**Value**
Action discretization $\mathscr{A} = [\Delta\tau]$	0.01 s
Revisit time T	1 s
Precision of Lagrangian relaxation ϵ	0.05
Number of rollouts	4
Horizon length H	15

Assumed Radar System

Each radar system in the sensor network generates independent measurements in range and angle. The corresponding variances in range and angle are described using a two-dimensional square matrix $\mathbf{R}$ with the variances placed on the diagonal.
Measurements in range and angle are picked from normal distributions $\mathscr{N}(r,\, \sigma_r^2)$ and $\mathscr{N}(\theta,\, \sigma_\theta^2)$ respectively. Variances σ_r^2 and σ_θ^2 are computed based on a Signal-to-Noise Ratio (SNR) value (see 5.8) and measurement noise variances ($\sigma_{r,0}^2$, $\sigma_{\theta,0}^2$) that are related to a reference target. Table III shows the system parameters of the assumed radar systems with respect to the reference measurement.

TABLE III. General system parameters of the assumed radar systems with respect to the reference measurement

Parameter	**Value**
Noise variance in range ($\sigma_{r,0}^2$)	$25\,\mathrm{m}^2$
Noise variance in angle ($\sigma_{\theta,0}^2$)	$4\mathrm{e}^{-4}\,\mathrm{rad}^2$

General Target Parameters

The standard scenario under consideration consists of six targets and two sensors. Each target has a certain RCS (ζ) and observation noise variance (σ_w^2). The received RCS at the sensors per target differ, due to the dependency of frequency on the RCS. For each target, the RCS and observation noise are picked according to Table IV. Since the focus is not on what the specific values of the RCS and observation noise are, the values are picked arbitrary.

TABLE IV. General target parameters

Parameter	Target 1	Target 2	Target 3	Target 4	Target 5	Target 6
ζ_1 (m^2)	12.5	5	22.5	20	5	25
ζ_2 (m^2)	50	20	90	80	20	100
σ_w^2	13	23.6	15.2	22.2	16.9	11.45

5.3 Simulation Results

The performance of the algorithm is investigated via simulated results in Matlab. Several scenarios will be considered to try to understand the working of the algorithm. To show the performance difference between the distributed implementation and the other two implementations both a static scenario (Section 5.3.1) and a dynamic scenario (Section 5.3.2) will be considered. Section 5.3.3 will provide some additional special cases of the distributed implementation. All results will be analyzed based on the budget allocation, runtime and primal ($Z_P = \mathbf{1}^T\mathbf{C}$) and dual ($Z_D$) cost. The primal ($\mathbf{1}^T\mathbf{C}$) and dual ($Z_D$) cost refer to the original and relaxed optimization problem respectively.

5.3.1 Static Scenario

The first scenario under consideration is a static scenario involving 2 stationary sensors and 6 stationary targets. The targets and sensors are located in a two dimensional grid according to Fig. 5.1. At time t = t_0 all three algorithmic solutions are applied to this

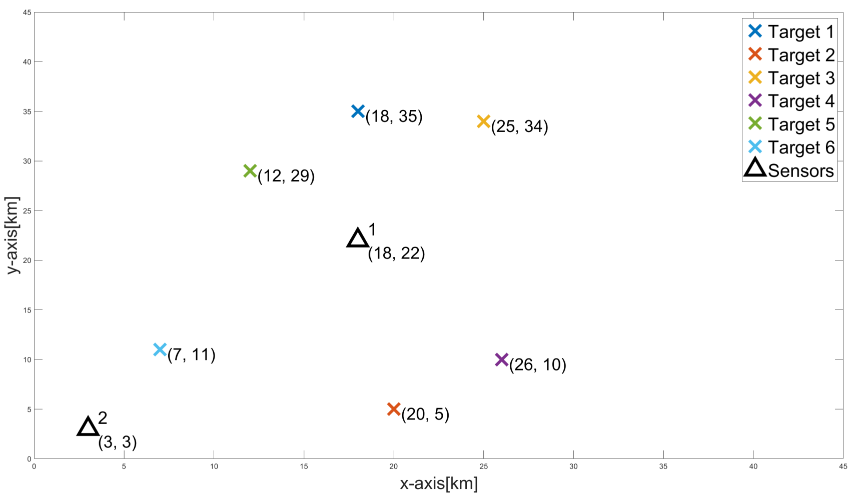


Figure 5.1. Visualization of stationary tracking scenario. Triangles indicate sensors and crosses indicate targets.

scenario using the simulation parameters described in Section 5.2. The results will be evaluated for each of the three solutions based on the cost and resulting dwell times.

Independent implementation

The simulation results of the independent implementation for the multi sensor scenario can be found in Fig. 5.2 and Fig. 5.3 showing the cost after the first budget allocation step and after multiple simulation steps. Fig. 5.2 shows the convergence during a single policy rollout. Note that the number of iterations required is quite large. This is mainly

TABLE V. Overview of resulting budget allocations over target one till six by sensor 1 and 2 of the Independent implementation

Sensor	Target 1	Target 2	Target 3	Target 4	Target 5	Target 6	Total Budget
1	12%	38%	12%	15%	9%	14%	100
2	26%	9%	26%	11%	23%	1%	96%

due to the initial pick of the Lagrangian multipliers and the selected step size. Also, the primal function remains constant during the policy rollout because it is independent of the optimization parameter τ. Fig. 5.3 shows the primal and dual cost over multiple simulation steps. Both the primal and dual cost converge to their respective optimal values over multiple simulation steps. The primal cost now also decreases in value due to the dependency on the dwell time during budget updates in between the budget allocation algorithm. The resulting budget allocation for each target per sensor (i.e., $\mathrm{b}^{n,m} = \frac{\tau^{n,m}}{T}$)

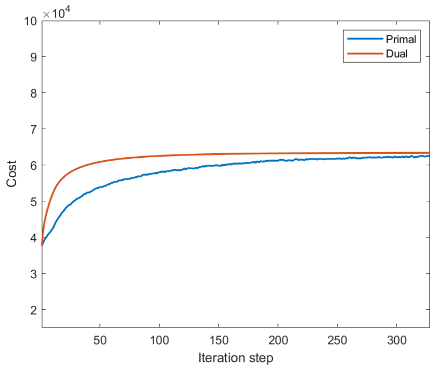


Figure 5.2. Evolution of the primal and dual cost of the independent implementation during the first budget allocation update process.

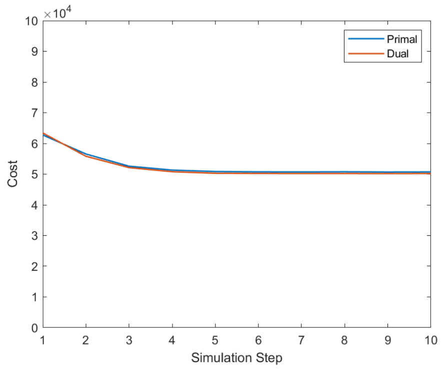


Figure 5.3. Evolution of the cost of the independent implementation over multiple simulation steps.

after the algorithm has converged over multiple simulation steps can be found in Table V. The sum of the budgets for both sensors are close to their maximum resource allocation. Due to the predefined error margin (ϵ) of five percent, the algorithm is allowed to stop as soon as the budget is within 5% of the maximum resource budget. This is the reason why the sum of the budgets for sensor 2 is 96% instead of 100%.

Centralized implementation

The simulation results of the centralized implementation are presented in Fig. 5.4 and Fig. 5.5 showing the cost after the first budget allocation step and after multiple simulation steps. Fig. 5.4 shows the convergence of the primal and dual cost during the first policy rollout. Note how the algorithm now requires significantly less iteration steps compared to the independent implementations. The evolution of the primal and dual cost over multiple simulation steps is given in Fig. 5.5. As can be seen from the figure, both the primal and dual cost remains more or less constant over the entire simulation. Indicating that the approximately optimal values are reached during the first budget allocation process. The resulting budget allocations for each target per sensor after the last simulation step can be found in Table VI. Since the centralized implementation takes the communication factor into account, the allocations are different compared to the independent implementation. For example, the budget allocation for target 3 is mainly covered by sensor 1 with 46% of its total budget whereas sensor 2 only spends 1% of its budget on target 3. As a result of this, sensor 2 can spend more budget on other relevant targets (e.g. target 6).
The reasoning behind the individual budget allocations is quite difficult due to the depen-

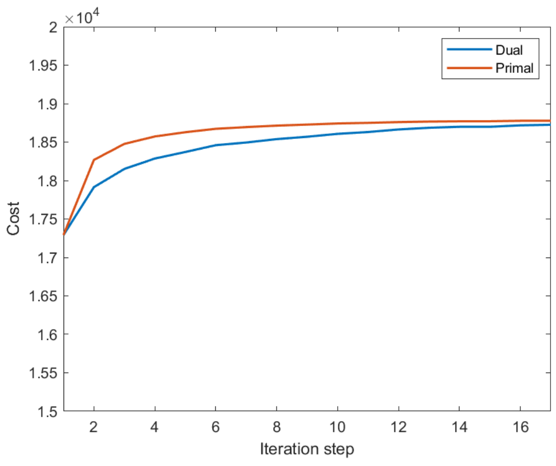


Figure 5.4. Evolution of the primal and dual cost of the centralized implementation during the first budget allocation update process.

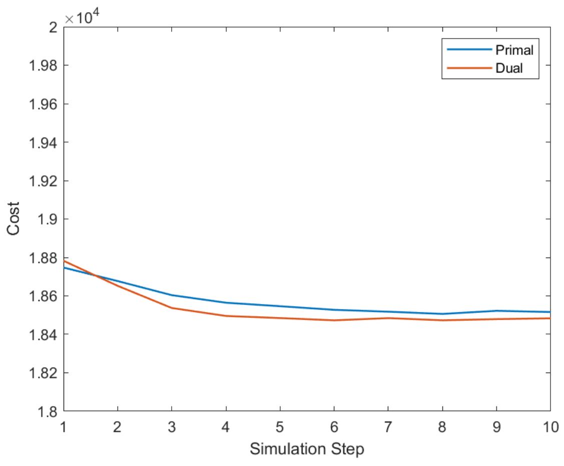


Figure 5.5. Evolution of the cost of the centralized implementation over multiple simulation steps.

dency on multiple factors (e.g. range, RCS or process noise), however according to the cost function they are approximately optimal.
Again, due to the predefined error margin both sensors do not exactly reach their maximum resource allocation.

TABLE VI. Overview of resulting budget allocations over target one till six by sensor 1 and 2 of the centralized implementation

Sensor	**Target 1**	**Target 2**	**Target 3**	**Target 4**	**Target 5**	**Target 6**	**Total Budget**
1	17%	18%	46%	8%	9%	1%	99%
2	35%	13%	1%	13%	24%	11%	97%

Distributed implementation

The simulation results of the distributed implementation are shown in Fig. 5.6 and Fig. 5.7 showing the convergence of the cost during the first budget allocation step and the evolution of the cost over multiple simulation steps.
Fig. 5.6 indicates that more iteration steps are required to reach convergence during the first budget allocation step due to different initial starting values. Based on the evolution of both the primal and dual cost over multiple simulation steps one can conclude that the distributed implementation converges to a value that is similar to the centralized implementation. The dwell times at the end of the simulation are given in Table VII. The budget allocations are approximately equal compared to the dwell times of the centralized implementation. Note how the total budget for sensor 2 exceeds 100%. Hence, the constraint as defined in problem 4.1 is not met. This can occur due to the fact that the predefined error margin (ϵ) is set to 5%. As soon as the sum of the budget allocations is within 5% of the maximum resource budget, the algorithm will stop even if the sum of the budgets is above 100%.

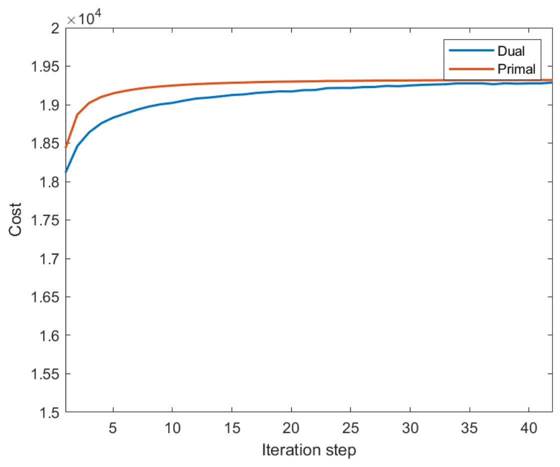


Figure 5.6. Evolution of the primal and dual cost of the distributed implementation during the first budget allocation update process.

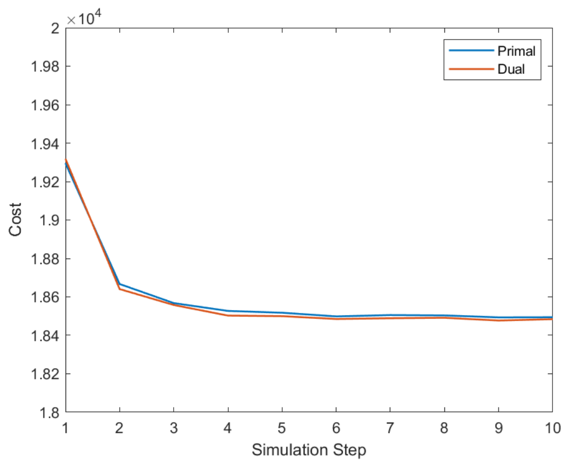


Figure 5.7. Evolution of the primal and dual cost of the distributed implementation over multiple simulation steps.

TABLE VII. Overview of resulting budget allocations over target one till six by sensor 1 and 2 of the distributed implementation

Sensor	**Target 1**	**Target 2**	**Target 3**	**Target 4**	**Target 5**	**Target 6**	**Total Budget**
1	16%	18%	44%	8%	9%	1%	96%
2	37%	14%	1%	14%	25%	11%	102%

Comparison

To compare the three implementations four static scenarios are defined. Each scenario comprises of two sensors and six arbitrarily placed targets around the sensors (see Appendix A.1). All three implementations are applied to the four different scenarios and are allowed to converge in 10 simulation steps. For the sake of better comparison each scenario is averaged over 10 consecutive runs.

Fig. 5.8 and Table VIII show the resulting averaged primal and dual cost and the average runtime for all three approaches at the end of each run. Both the centralized and distributed implementations outperform the independent implementation with respect to the primal and dual cost by a factor two to three. As expected, due to the exponential

TABLE VIII. Runtime comparison between the independent, centralized and distributed implementation averaged over multiple static scenarios.

Approach	**Runtime in seconds**
Independent	90
Distributed	34
Centralized	812

increase of the action space for an increasing number of sensors, the average runtime of the centralized implementation is significantly larger with respect to the other two implementations. The average runtime of the distributed implementation is more then 20 times less compared to the centralized implementation.

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.

Note that the costs of the centralized and distributed implementation are approximately

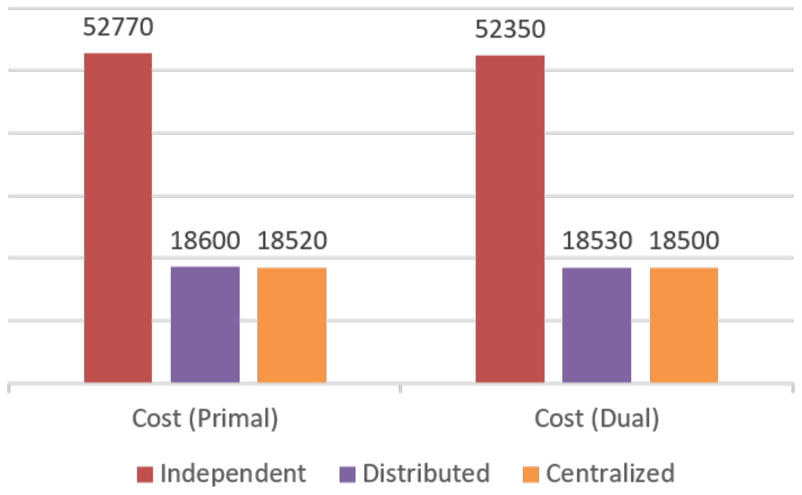


Figure 5.8. Cost comparison between the independent, centralized and distributed extension averaged over multiple static scenarios. The results are compared based on the average primal cost (left) and average dual cost (right)

equal. This implies that both implementations computed more or less the same budget allocations. To verify this, the percentage difference in average budget allocation is computed for each considered target in one of the comparison scenarios (see Fig. 5.9).
The maximum time difference in budget allocation is roughly 2.1% which is well within the previously defined error margin (ϵ) of 5%. From this, one can conclude that indeed the centralized and distributed implementations are approximately equal and over time both implementations will converge to more or less the same results in a static scenario.

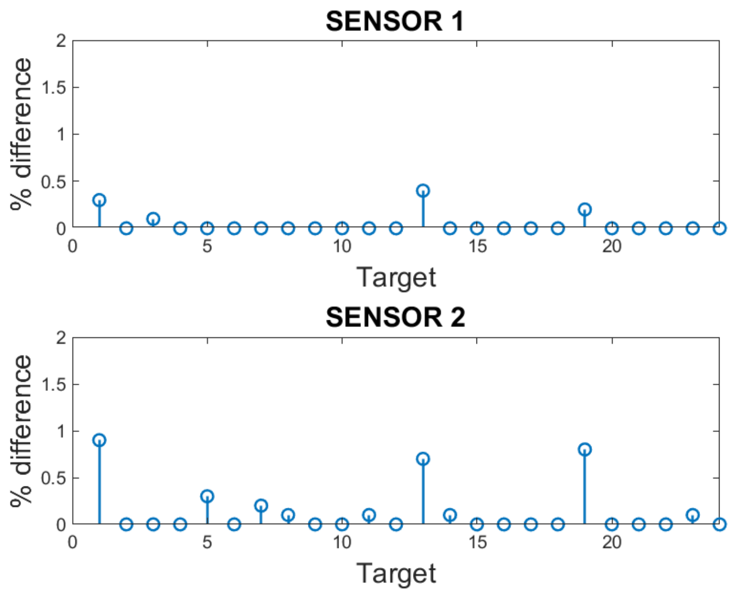


Figure 5.9. Percentage difference in average budget allocation between the centralized and distributed implementations over 24 arbitrarily placed targets for sensor 1 (top) and sensor 2 (bottom).

5.3.2 Dynamic Scenario

To further explore the workings of the derived algorithms, a dynamic scenario is considered. For this scenario, the same number of sensors (two) and targets (six) are used. However, for this scenario it is assumed that the targets move according to a linear motion model as described in Section 5.1.
The simulation consists of ten simulation steps, in which ten consecutive policy rollouts are applied during which the optimal budget are computed. To reduce the computational complexity, the action space consists of the dwell time alone. The revisit time is fixed at 1 second for all sensors. A visualization of the dynamic tracking scenario is shown in Fig. 5.10. The initial velocities of targets one till six are given in Table IX.

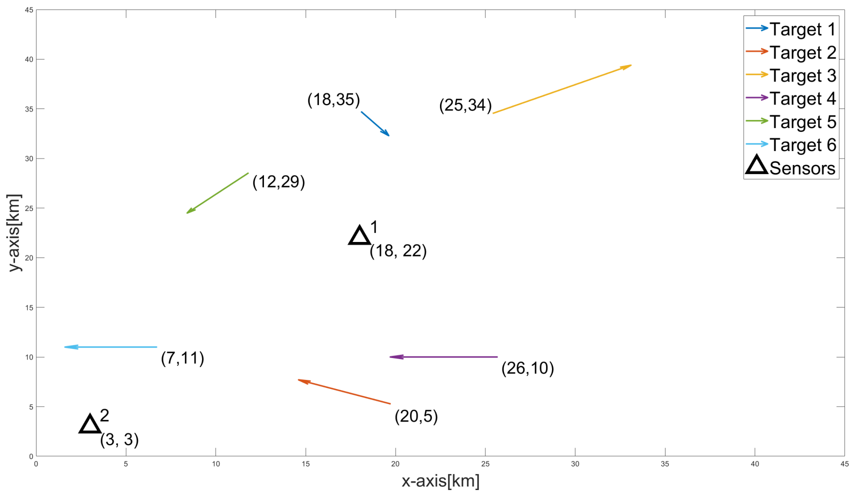


Figure 5.10. Visualization of dynamic tracking scenario

TABLE IX. Target velocities

Target	V_x(m/s)	V_y(m/s)
1	9	-15
2	-30	15
3	45	30
4	-35	0
5	-20	-25
6	-30	0

Independent implementation

The simulation results of the independent implementation are shown in Fig. 5.11 and Fig. 5.12 showing the convergence of the primal and dual cost during the first budget allocation step and the budget allocation over multiple simulation steps. Note how the cost converges within approximately 900 iterations. Again, the initial pick of the Lagrangian multipliers cause the algorithm to converge relatively slow.
Ideally, sensor 1 would spend significantly more budget on target 3 compared to sensor 2 due to the placement of target 3. However, due to the lack of communication, both sensor 1 and sensor 2 spend allocate a large portion of their budget to target 3. The resulting budget allocations of the independent implementation are given in Table X. Again the total budget of a sensor is exceeded here due to the way the error margin is defined.

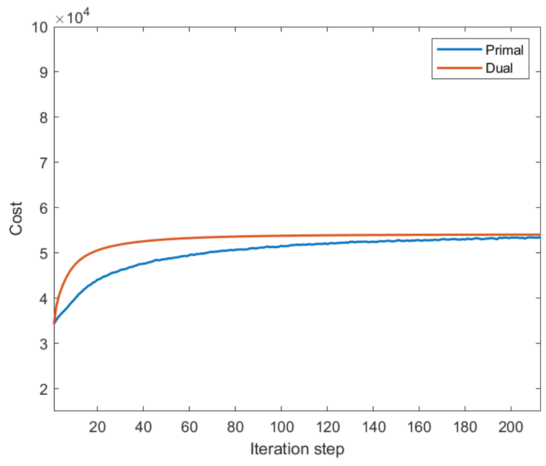


Figure 5.11. Evolution of the primal and dual cost of the Independent implementation during the first budget allocation update process

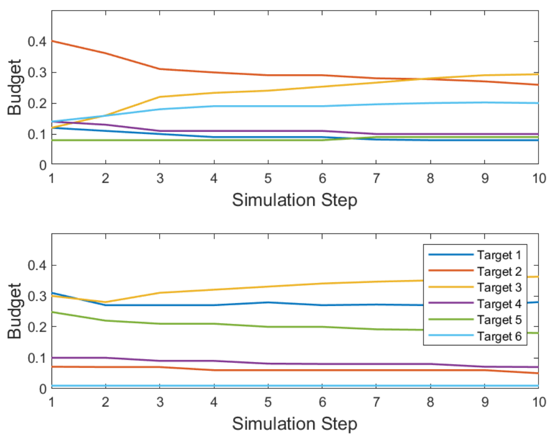


Figure 5.12. Budget allocation of the Independent implementation over multiple simulation steps

Sensor	Target 1	Target 2	Target 3	Target 4	Target 5	Target 6	Total Budget
1	6%	23%	34%	9%	9%	22%	103%
2	27%	5%	40%	6%	16%	1%	95%

TABLE X. Overview of resulting budget allocations over target one till six by sensor 1 and 2 of the Independent implementation in a dynamic scenario at the end of the simulation

Centralized implementation

The simulation results of the centralized implementation are shown in Fig. 5.13 and Fig. 5.14 showing the convergence of the primal and dual cost during the first budget allocation step and the budget allocation over multiple simulation steps. The policy rollout now converges in significantly less iterations.
The budget allocations indicate that sensor 1 spends the biggest portion of its budget on target 3, while sensor 2 allocates its resource over the other targets. Purely based on the placement of the targets one would expect such a behaviour. The resulting budget allocations at the end of the simulation are given in Table XI

TABLE XI. Overview of resulting budget allocations of the centralized implementation to target one till six by sensor 1 and 2

Sensor	Target 1	Target 2	Target 3	Target 4	Target 5	Target 6	Total Budget
1	9%	9%	76%	4%	4%	1%	103%
2	38%	16%	1%	11%	24%	10%	100%

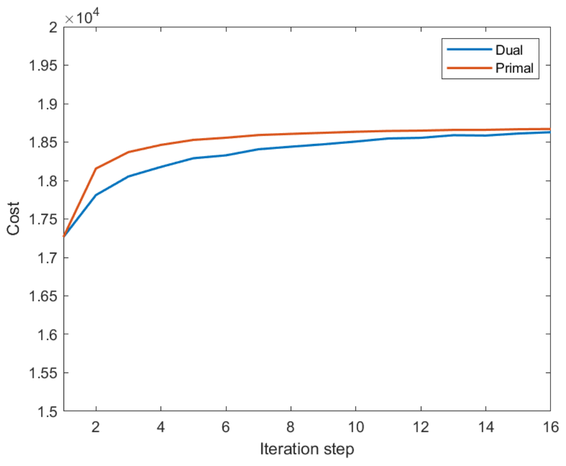


Figure 5.13. Evolution of the primal and dual cost of the centralized implementation during the first budget allocation update process

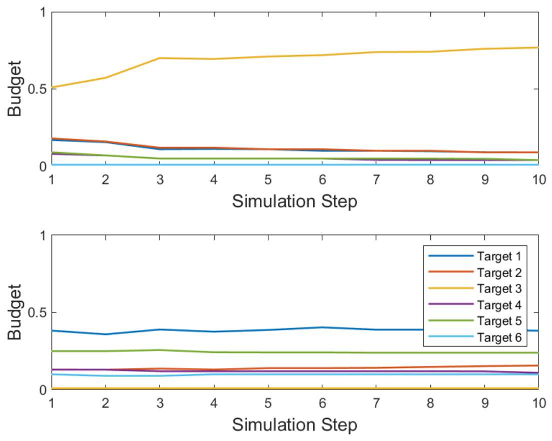


Figure 5.14. Evolution of the primal and dual cost of the centralized implementation over multiple simulation steps in a dynamic scenario

Distributed implementation

The simulation results of the distributed implementation are given in Fig. 5.15, showing the evolution of the primal and dual cost during the first budget allocation step and Fig. 5.16 showing the budget allocation over multiple time steps. Note that the number of iterations required for the cost to converge is approximately 36. Which is about twice as much compared to the centralized implementation. Indeed, due to the sharing of the last known dwell times, the initial starting values for each policy rollout are different. As a result of this, more iteration steps are required to reach convergence for the first budget allocation process. The resulting cost value however, is close to the converged cost of the centralized implementation.

The budget allocation over multiple time steps show a similar results as with the centralized implementation. This is also verified by looking at Table XII showing the resulting budget allocations at the end of the simulation.

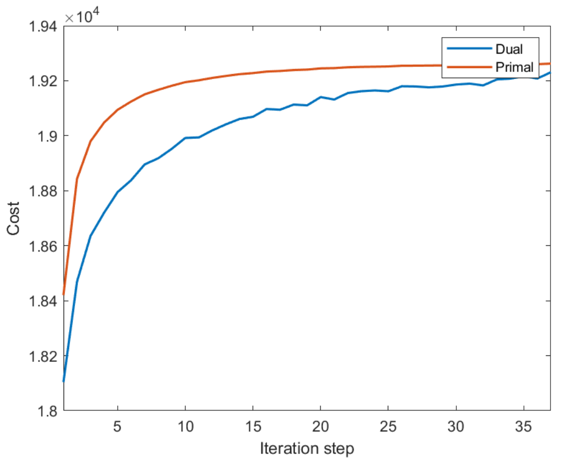


Figure 5.15. Evolution of the primal and dual cost of the distributed implementation during the first budget allocation update process.

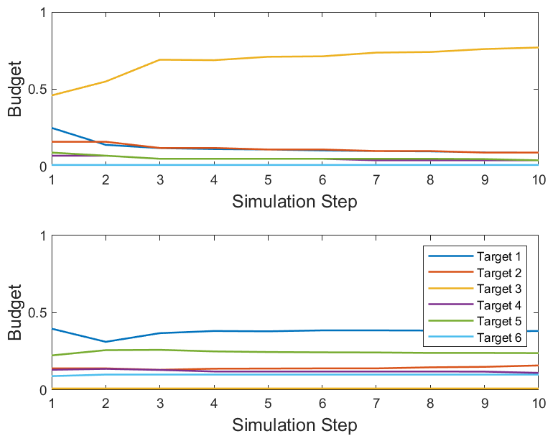


Figure 5.16. Budget allocation using the distributed implementation over multiple simulation steps in a dynamic scenario.

TABLE XII. Overview of resulting budget allocations of the distributed implementation to target one till six by sensor 1 and 2

Sensor	Target 1	Target 2	Target 3	Target 4	Target 5	Target 6	Total Budget
1	9%	9%	77%	4%	4%	1%	104%
2	38%	16%	1%	11%	24%	10%	100%

Comparison

A comparison is made between the three implementations in a dynamic setting. Four dynamic scenarios are considered consisting of 6 dynamic targets with arbitrary starting locations and 2 stationary sensors (see Appendix B.1). All three implementations are applied to the four different scenarios and are allowed to converge in 10 simulation steps. Each scenario is again averaged over 10 consecutive runs.
Fig. 5.17 and Table XIII show the average primal and dual cost and the average runtime over the entire simulation for the three implementations. Note how the relative differences between the primal and dual cost are more or less equal to the static case indicating that the distributed implementation has the best overall performance. The cost of the distributed implementation is more or less equal to the centralized implementation and the average runtime is approximately ten times as less with respect to the centralized implementation.

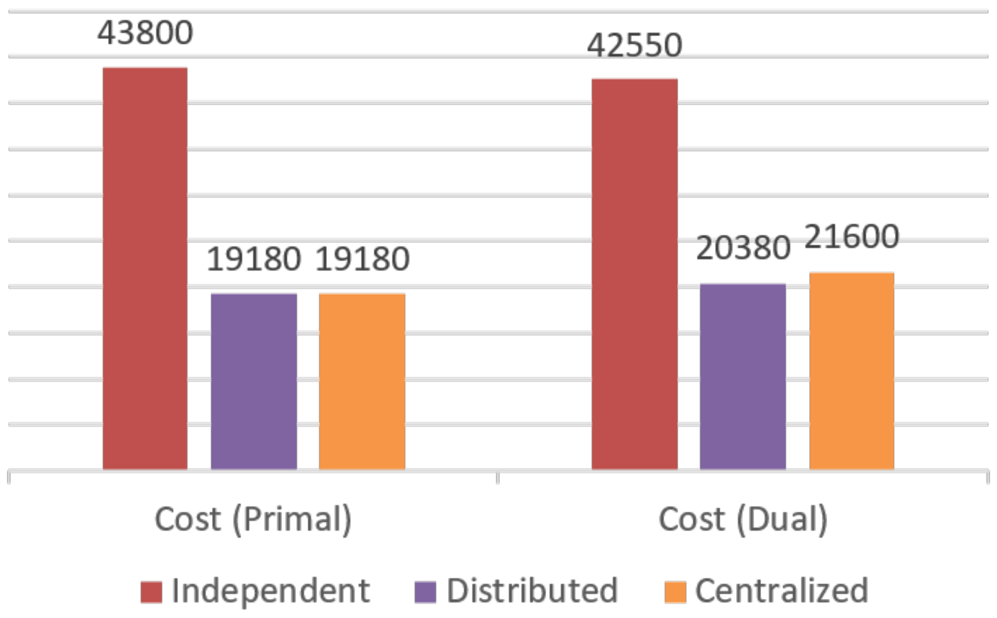


Figure 5.17. Cost comparison between the independent, centralized and distributed extension averaged over multiple dynamic scenarios. The results are compared based on the average primal cost (left) and average dual cost (right)

To verify whether the distributed implementation has more or less the same budget allocations, the percentage difference in average budget is compared between each of the dynamic targets considered in one of the scenarios. Since the targets will be displaced in between budget allocation updates, the last known dwell times will not exactly correspond to the desired dwell times of the centralized implementation. Hence, one would expect to see a larger percentage difference in average dwell times compared to the static case. The resulting percentage difference in dwell times are plotted in Fig. 5.18.
Note how the percentage difference is larger compared to the static case, however the

TABLE XIII. Runtime comparison between the independent, centralized and distributed extension averaged over multiple dynamic scenarios.

Approach	Runtime in seconds
Independent	158
Distributed	113
Centralized	1187

difference are still well within the predefined error margin of 5%.

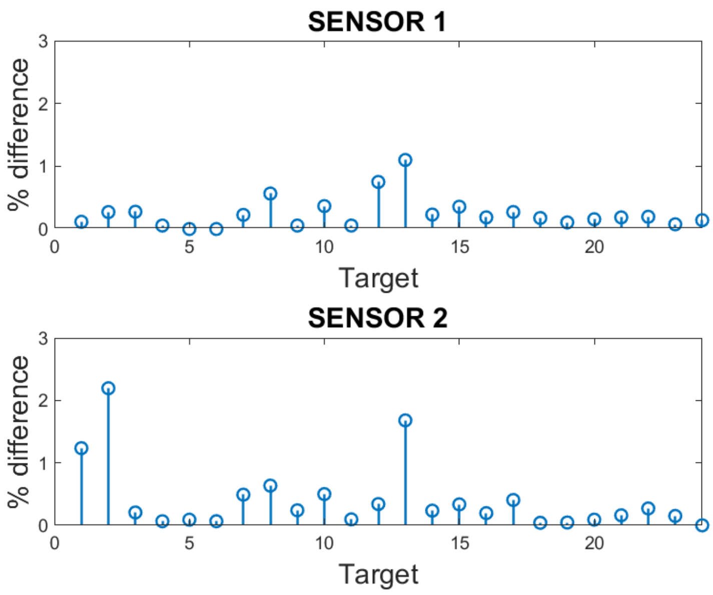


Figure 5.18. Percentage difference in dwell times for 24 randomly placed targets between the centralized and distributed implementation in a dynamic scenario.

5.3.3 Special Cases

Based on the results of the static and dynamic scenarios, the conclusion can be made that the overall performance i.e., resulting cost and runtime, of the distributed implementation is better compared to the centralized and independent implementation. Hence, the rest of this section the distributed implementation will only be considered.
There is a vast amount of other possible scenarios that can be considered. Most scenarios however will not produce significantly alternative results compared to the results obtained in sections 5.3.1 and 5.3.2. This section shows several cases in which the distributed fusion implementation might perform sub-optimal or does not work at all.
The simulation parameters for all special cases are equal to the ones used in the previous section unless specified otherwise. Each simulation is repeated 10 times and averaged.

Targets at the Same Location

The first special case under consideration is one where the targets are overlapping with each other. The scenario consists of 5 stationary targets and 2 stationary sensors placed according to Fig. 5.19.
Although the targets have different properties with respect to RCS and observation noise the algorithm might still struggle with allocating budget over the targets in an efficient manner in this scenario. The resulting budget allocations for sensor 1 and sensor 2 are

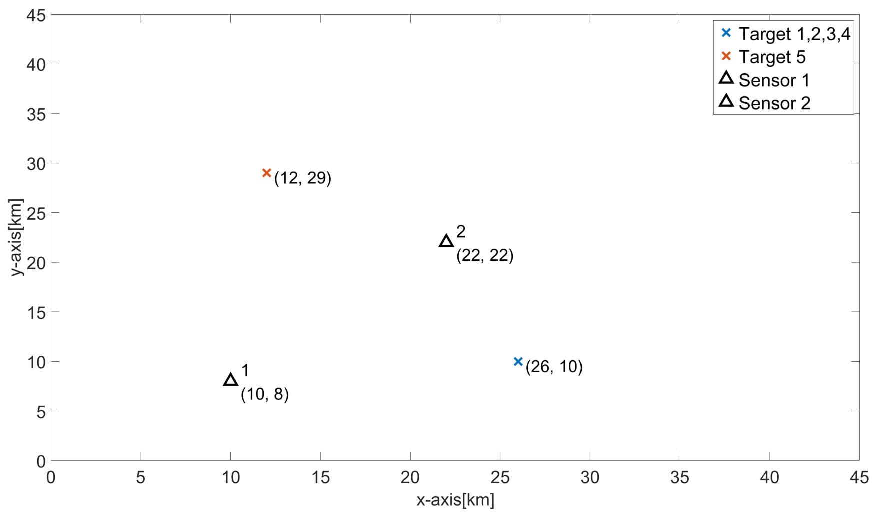


Figure 5.19. Tracking scenario with overlapping targets.

presented in Fig. 5.20 and Fig. 5.21 respectively.
Both figures indicate a fluctuation for target 5 from simulation step to simulation step. It appears as if the sensors do not have a single optimal solution in this case. This is due to the placement of target 5 with respect to the sensors.
The resulting dwell times after the last simulation step are given in Table XIV.

TABLE XIV. Dwell times allocated at the end of the simulation for a scenario with overlapping targets

Sensor	Target 1	Target 2	Target 3	Target 4	Target 5	Total Budget
1	17%	28%	14%	16%	21%	96%
2	12%	19%	10%	11%	45%	97%

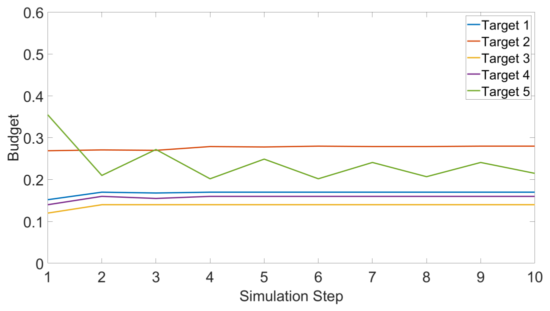


Figure 5.20. Budget allocation for sensor 1 in a scenario with overlapping targets.

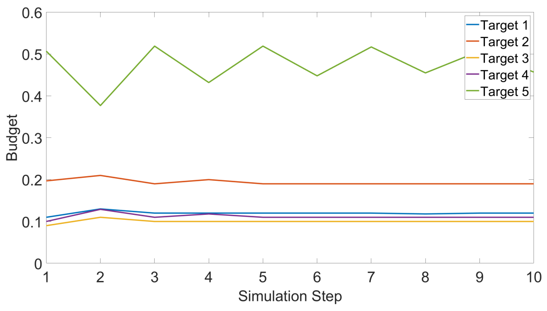


Figure 5.21. Budget allocation for sensor 2 in a scenario with overlapping targets.

Sensors at the Same Location

The second case to consider is a case where the sensors are overlapping with each other. The scenario consists of 6 stationary targets and 2 overlapping stationary sensors placed according to Fig. 5.22. The only distinction in this scenario that can be made between

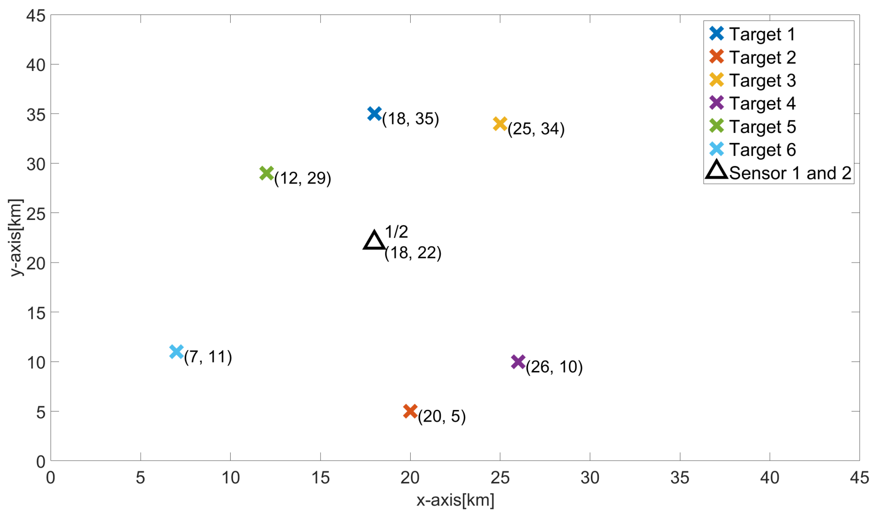


Figure 5.22. Tracking scenario with overlapping sensors

both sensors is the operating frequency since the range and process noise will be equal. Consequently, the main difference in tracking parameters is the RCS. Since the other parameters in both targets are the same convergence might be more difficult to reach. Fig. 5.23 and Fig. 5.24 show the budget allocations for sensor 1 and sensor 2 over multiple simulation steps. Note how the budget allocation for both sensors fluctuates in between simulation steps.

For sensor 1 this results in either a high budget allocation to target 2 and low budget allocation to the other sensors or a more balanced budget allocation between all targets. A similar effect can be seen with the budget allocation of sensor 2.

A possible argument for this to be happening is due to the placement of target 2. Target 2 is located the farthest away from the sensors and the observation noise and RCS are considerable. Consequently, both sensors want to spend a significant amount of budget on target 2. After the first iteration, this is indeed what can be observed, target 2 receives from both sensors the largest budget. After the first iteration the information between both sensors is shared, which allows both sensors to perceive that the other sensor already allocated a large amount of budget to target 2. Then both sensors reduce the budget allocation with respect to target 2 which in turn results in the opposite effect.

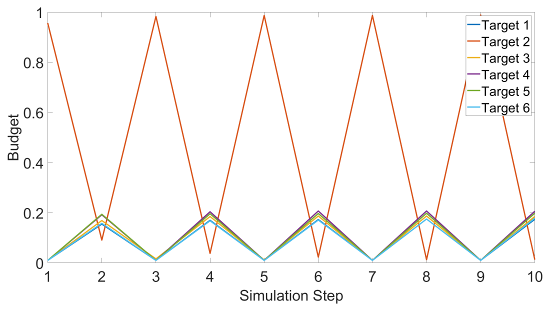


Figure 5.23. Budget allocation for sensor 1 in a scenario with overlapping sensors.

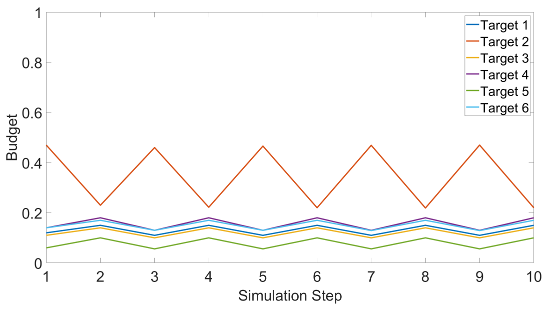


Figure 5.24. Budget allocation for sensor 2 in a scenario with overlapping sensors.

To verify whether this is a recurring issue, target 2 was removed from the simulation (see Appendix C.1). Results indicated that even when target 2 is removed, the fluctuating of budgets between simulation steps remains. The resulting dwell times at the end of the simulation are shown in Table XV.

TABLE XV. Dwell times at the end of the simulation for a scenario with overlapping sensors

Sensor	Target 1	Target 2	Target 3	Target 4	Target 5	Target 6	Total Budget
1	21%	1%	18%	21%	18%	16%	95%
2	15%	22%	14%	18%	10%	17%	96%

Dropping Targets

The third scenario under consideration is one where one of the sensors drops several targets during the simulation. For this example, seven dynamic targets and two sensors are considered (see Fig. 5.25). In this example specifically, sensor 2 is not able to track target 6 and 7 after simulation step 10 while sensor 1 is able to track all targets. Fig. 5.26 and

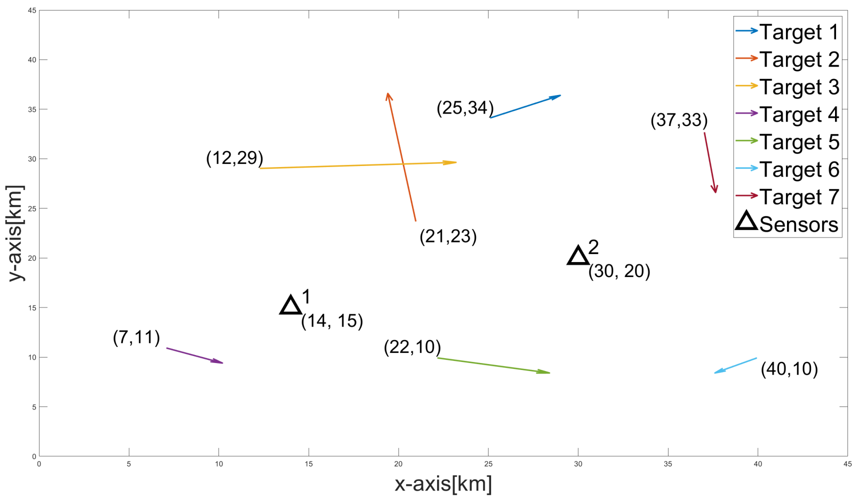


Figure 5.25. Tracking scenario where sensor 2 is not able to track targets 6 and 7 after simulation step 10.

Fig. 5.27 show the budget allocations for sensor 1 and sensor 2 over 40 simulation steps. At simulation step 10 a clear transition is visible. As soon as sensor 2 drops target 6 and

7, a compensation is made by sensor 1 by allocating a significant amount of its budget on target 6 and 7 indicating a form of robustness of the algorithm to cope with sudden changes in the tracking process. Sensor 2 on the other hand redistributes its resources over the other targets. Also, since the last known dwell times are shared with the other sensors, it takes an additional simulation step for sensor 1 to make the adjustment of the dwell times. The dwell times at the end of the simulation are given in Table XVI.

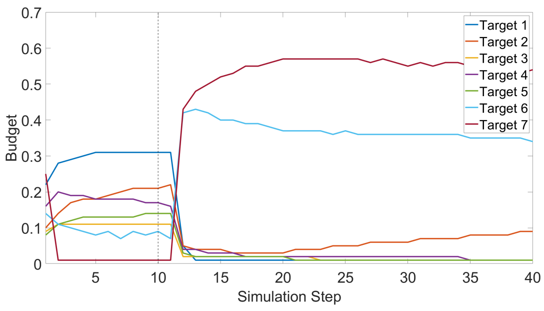


Figure 5.26. Budget allocation for sensor 1.

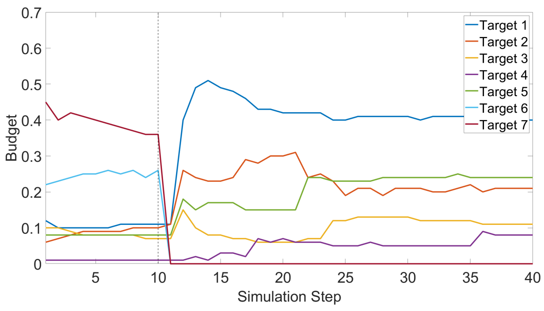


Figure 5.27. Budget allocation for sensor 2. At time $k = 10$, sensor 2 is not able to track targets 6 and 7 anymore.

TABLE XVI. Dwell times at the end of the simulation in a scenario where sensor 2 can track a limited a limited amount of targets

Sensor	Target 1	Target 2	Target 3	Target 4	Target 5	Target 6	Target 7	Total
1	1%	9%	1%	1%	1	34%	54%	101%
2	40%	21%	11%	8%	24%	0%	0	104%

Varying Budgets

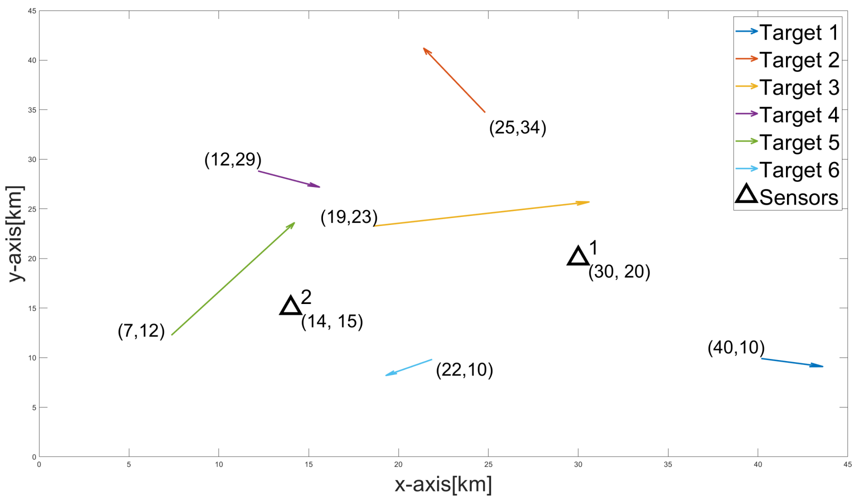


Figure 5.28. Tracking scenario where the sensors have a limited amount of budget available

Both sensors do not necessarily have the same amount of budget at their disposal. To simulate this effect a scenario is considered where the maximum budget of one of the sensors changes during the simulation. More specifically, the maximum budget of sensor 1 is set to 70% after simulation step 10 whereas sensor 2 keeps the maximum budget of 100%. The considered scenario is given in Fig. 5.28.

Fig. 5.29 and Fig. 5.30 show the budget allocations for sensor 1 and sensor 2 over 20 simulation steps. The budget allocated over time shows clearly how the algorithm is capable of adjusting its dwell times based on the change in maximum resources available. The

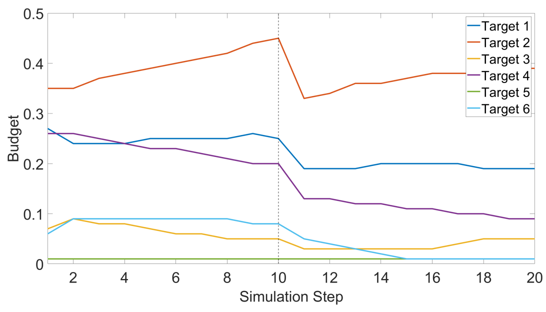


Figure 5.29. Budget allocation for sensor 1. At time $k = 10$, the maximum budget is reduced to 0.7

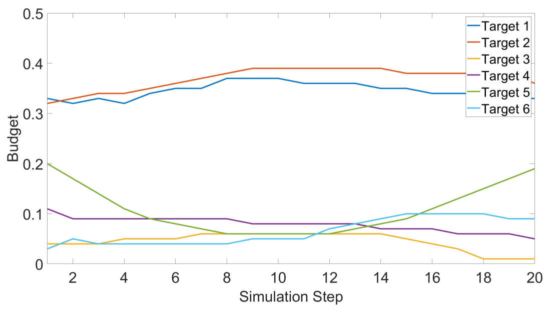


Figure 5.30. Budget allocation for sensor 2.

dwell times at the end of the simulation are given in Table XVII. Both sensors allocated their budget such that they both operate at their respective resource limit.

TABLE XVII. Dwell times at the end of the simulation in a scenario where sensor 2 has less available resources compared to sensor 1.

Sensor	**Target 1**	**Target 2**	**Target 3**	**Target 4**	**Target 5**	**Target 6**	**Total Budget**
1	19%	39%	5%	9%	1%	1%	74%
2	33%	36%	1%	5%	19%	9%	103%

Big Scenario

To verify whether the algorithm works in a scenario with more than two sensor, a scenario is considered with four sensors and ten stationary targets. The resulting budget allocations and cost can be found in Appendix C.2. Both the cost and budget allocations converge similarly as with the scenario involving two sensors.

Chapter 6

Conclusion

An approximately optimal solution to the RRM problem in a multi-sensor network has been presented. The RRM problem was formulated as a multi-sensor constrained optimization problem in which the optimal actions need to be found for multiple tasks. The cost, defined as the predicted error covariance, needs to be minimized while at the same time the maximum resource budget of each sensor is not exceeded.

A novel framework was developed for solving the RRM problem in a sensor network. The proposed framework exploits an already existing solution for RRM in a single sensor. The different tasks are modeled as a POMDP and the problem is solved non-myopically using a combination of Lagrangian relaxation and policy rollout. The multi-sensor implementation is achieved by exploiting communication between sensors.

Two novel multi-sensor implementations, namely a centralized and distributed implementation have been developed to control the actions of the sensors at an affordable computational cost. The centralized implementation, defined as the approximately optimal solution, utilizes a global policy per task. As such, the policy rollout for a single target needs to explore the actions of multiple sensors. For each global policy rollout significant computational resources are required.
As a practical alternative to the centralized implementation, the distributed implementation has been developed to reduce the computational complexity. Now per sensor and per task a policy rollout is computed. Hence, each individual policy rollout only needs to explore the action space related to a single target and a single sensor. To maintain a similar performance as the centralized implementation, at the beginning of each policy rollout the last known actions of the other sensors are shared. Resulting in an increase of computational resources required due to the added communication overhead. However, this is assumed to be negligible compared to the overall reduction in computation time for the distributed implementation.
An additional third independent implementation is considered. The independent implementation uses no communication during the optimization process and is considered to be the implementation with the lowest performance with respect to the cost.

Using two-dimensional simulated radar tracking scenarios all implementations were verified. The targets were assumed to move according to a linear model based on the EKF. The EKF should be capable of dealing with non-linear movement of targets. However, this has not been taken into account in this thesis.

The results of the three implementations were verified by analyzing the convergence of the cost and the resulting budget allocations. By comparing the budget allocation of multiple randomly placed targets in static and dynamic scenarios it has been shown that over time the distributed implementation converges to approximately similar results as the centralized implementation.

Both the novel centralized and distributed implementation have been compared to the independent implementation. The simulations show that the centralized and distributed implementation outperform the independent implementation with respect to the average cost. The independent implementation has a cost that is more than twice as large as the other two implementations. The cost of the centralized and distributed implementation converged to approximately equal values.

To indicate the difference in computational resources required for the centralized and distributed implementation, the average runtime has been compared. It has been shown that the centralized implementation requires a computational load that is more than ten times as big as the distributed implementation for both a static and dynamic scenario. This is mainly due to the modelling of the policy rollout. Because the centralized implementation, defined as the approximately optimal solution, utilizes a global policy per task to explore the actions for multiple sensors, the action space will be significantly larger compared to the policy rollout used for the distributed implementation. For an increasing number of sensors the action space of the centralized implementation will grow exponentially, whereas a policy rollout for the distributed implementation remains unchanged for an increasing number of sensors.

Several additional special cases have been provided to show the validity of the distributed implementation under more extreme conditions. In general these results indicated that there is still room for improvement on the algorithm. For example, when stacking sensors on top of each other, the budget allocations might fluctuate heavily for both sensors between fixed values. This behaviour is undesired since it implies a kind of indecisiveness of the algorithm.
On the other hand, when the budget is changed during optimization or when a sensor drops some of the targets the algorithm showed that it can deal with those sudden changes. In addition to that it has been shown that the algorithm still performs as desired when increasing the number of sensors.

The novel centralized implementation results in the approximately optimal solution for the RRM but requires a heavy computational load making it not a desired solution. The alternative distributed implementation shows promising results since it converges to similar results as the centralized implementation using significantly less computational resources.

Because the results are based on simulations, the question arises what the performance of the algorithm will be in a real life scenario. Hence, in future work, it would be interesting to look into real-world scenarios to test the derived algorithm. Furthermore, it would be interesting to take into account different revisit times or to optimize it jointly or even to include other aspects such as searching and classification. Also, in this thesis no communication constraints were defined for the communication between sensors. It would be interesting to look at problems that do include those constraints. Finally, the selection of the cost function and the method of comparing needs to be investigated further.

Bibliography

[1] A. O. Hero and D. Cochran. Sensor management: Past, present, and future. *IEEE Sensors Journal*, 11(12):3064–3075, 2011.

[2] D. Cochran & K. Kastella A.O. Hero, D. Castanon. *Foundations and Applications of Sensor Management.* Springer US, 2008.

[3] A. Charlish, F. Hoffmann, C. Degen, and I. Schlangen. The development from adaptive to cognitive radar resource management. *IEEE Aerospace and Electronic Systems Magazine*, 35:8–19, 06 2020. doi: 10.1109/MAES.2019.2957847.

[4] S. Haykin. Cognitive radar: a way of the future. *IEEE Signal Processing Magazine*, 23(1):30–40, 2006. doi: 10.1109/MSP.2006.1593335.

[5] 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*, 34(12):20–29, Dec 2019. ISSN 1557-959X. doi: 10.1109/MAES.2019.2953802.

[6] S. Brüggenwirth, M. Warnke, S. Wagner, and K. Barth. Cognitive radar for classification. *IEEE Aerospace and Electronic Systems Magazine*, 34(12):30–38, Dec 2019. ISSN 1557-959X. doi: 10.1109/MAES.2019.2958546.

[7] J. Scheer & W.A. Holm M. A. Richards. *Principles of modern radar.* Raleigh, N.C. : Scitech Pub, 2010.

[8] C. A. Balanis. *Antenna Theory: Analysis and Design, 4th Edition.* Wiley, 2016.

[9] L. Nicolaescu and T. Oroian. Radar cross section. In *5th International Conference on Telecommunications in Modern Satellite, Cable and Broadcasting Service. TELSIKS 2001. Proceedings of Papers (Cat. No.01EX517)*, volume 1, pages 65–68 vol.1, 2001. doi: 10.1109/TELSKS.2001.954850.

[10] K. Benson. Phased array beamforming ics simplify antenna design. *Analag Dialogue*, 53, 2019.

[11] A. Charlish. Autonomous agents for multi-function radar resource management. 11 2011.

[12] Ömer Ç. Radar resource management techniques for multi-function phased array radars. 2014.

[13] F. Katsilieris. *Sensor management for surveillance and tracking: An operational perspective.* PhD thesis, 03 2015.

[14] H. Zhang, J. XIE, B. Zong, W. Lu, and C. Sheng. Dynamic priority scheduling method for the air defense phased array radar. *IET Radar, Sonar & Navigation*, 11, 02 2017. doi: 10.1049/iet-rsn.2016.0549.

[15] F. B. Abdelaziz and H. Mir. An optimization model and tabu search heuristic for scheduling of tasks on a radar sensor. *IEEE Sensors Journal*, 16(17):6694–6702, 2016. doi: 10.1109/JSEN.2016.2587730.

[16] F. Katsilieris, J.N. Driessen, and A. Yarovoy. Threat-based sensor management for target tracking. *IEEE Transactions on Aerospace and Electronic Systems*, 51(4):2772–2785, 2015. doi: 10.1109/TAES.2015.140052.

[17] C. Kreucher, K. Kastella, and A. O. Hero. Sensor management using an active sensing approach. 85(3), 2005. ISSN 0165-1684. doi: 10.1016/j.sigpro.2004.11.004. URL `https://doi-org.tudelft.idm.oclc.org/10.1016/j.sigpro.2004.11.004`.

[18] S. Boyd and L. Vandenberghe. *Convex Optimization.* Cambridge University Press, USA, 2004. ISBN 0521833787.

[19] J. Wintenby and V. Krishnamurthy. Hierarchical resource management in adaptive airborne surveillance radars. *IEEE Transactions on Aerospace and Electronic Systems*, 42(2):401–420, 2006.

[20] 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*, pages 8393–8393, 2012. doi: 10.1117/12.919552. URL `https://doi.org/10.1117/12.919552`.

[21] S. Boyd and Jaehyun P. Subgradient methods. *lecture notes of EE364b, Stanford University, Spring 2013-2014*, 2014, 05 2014.

[22] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. *Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers.* 2011.

[23] S. Jiang, Y. Lei, S. Wang, and D. Wang. An asynchronous admm algorithm for distributed optimization with dynamic scheduling strategy. *IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems*, pages 1–8, 11 2019.

[24] S. Park, Y. Min, J. Ha, D. Cho, and H. Choi. A distributed admm approach to non-myopic path planning for multi-target tracking. *IEEE Access*, PP:1–1, 11 2019. doi: 10.1109/ACCESS.2019.2952235.

[25] Y. Deng, Z. Chen, X. Yao, S. Hassan, and J. Wu. Task scheduling for smart city applications based on multi-server mobile edge computing. *IEEE Access*, 7:14410–14421, 2019. doi: 10.1109/ACCESS.2019.2893486.

[26] G. Zhang and R. Heusdens. Distributed optimization using the primal-dual method of multipliers. *IEEE Transactions on Signal and Information Processing over Networks*, 4(1):173–187, 2018. doi: 10.1109/TSIPN.2017.2672403.

[27] J. Giesen and S. Laue. Distributed convex optimization with many convex constraints. 10 2016.

[28] W. Powell. A unified framework for stochastic optimization. *European Journal of Operational Research*, 275, 07 2018. doi: 10.1016/j.ejor.2018.07.014.

[29] A. Charlish, K. Bell, and C. Kreucher. Implementing perception-action cycles using stochastic optimization. In *2020 IEEE Radar Conference (RadarConf20)*, pages 1–6, 2020. doi: 10.1109/RadarConf2043947.2020.9266338.

[30] M. Littman. A tutorial on partially observable markov decision processes. *Journal of Mathematical Psychology*, 53:119–125, 06 2009. doi: 10.1016/j.jmp.2009.01.005.

[31] M. van Otterlo and M. Wiering. *Reinforcement Learning: State-of-the-Art.* Adaptation, Learning, and Optimization 12. Springer-Verlag Berlin Heidelberg, 1 edition, 2012. ISBN 978-3-642-27644-6,978-3-642-27645-3. URL `http://gen.lib.rus.ec/book/index.php?md5=b52a9923923ff555110ae81ad0d384a7`.

[32] R. Bellman. *Dynamic Programming.* Princeton University Press, USA, 2010. ISBN 0691146683.

[33] M. I. Schöpe, J.N. Driessen, and A. Yarovoy. Multi-task sensor resource balancing using lagrangian relaxation and policy rollout. In *2020 IEEE 23rd International Conference on Information Fusion (FUSION)*, pages 1–8, 2020. doi: 10.23919/FUSION45008.2020.9190546.

[34] D. Bertsekas. Multiagent reinforcement learning: Rollout and policy iteration. *IEEE/-CAA Journal of Automatica Sinica*, 8(2):249–272, 2021. doi: 10.1109/JAS.2021.100 3814.

[35] M. I. Schöpe, J.N. Driessen, and A. Yarovoy. Optimal balancing of multi-function radar budget for multi-target tracking using lagrangian relaxation. In *2019 22th International Conference on Information Fusion (FUSION)*, pages 1–8, 2019.

[36] B. D. O. Anderson and J. B. Moore. *Optimal Filtering.* Dover Publications, 2005.

[37] Z. Chen. Bayesian filtering: From kalman filters to particle filters, and beyond. *Statistics*, 182, 01 2003. doi: 10.1080/02331880309257.

[38] 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*, 64(13):3509–3522, 2016. doi: 10.1109/TSP.2016.2550005.

[39] N. Assimakis, M. Adam, and A. Douladiris. Information filter and kalman filter comparison: Selection of the faster filter. *International Journal of Information Engineering (IJIE)*, 2:1–5, 01 2012.

[40] C. Chong. Forty years of distributed estimation: A review of noteworthy developments. In *2017 Sensor Data Fusion: Trends, Solutions, Applications (SDF)*, pages 1–10, 2017. doi: 10.1109/SDF.2017.8126358.

[41] B. Khaleghi, A. Khamis, F. O. Karray, and S. N. Razavi. Multisensor data fusion: A review of the state-of-the-art. *Information Fusion*, 14(1):28–44, 2013. ISSN 1566-2535. doi: https://doi.org/10.1016/j.inffus.2011.08.001. URL `https://www.sciencedirect.com/science/article/pii/S1566253511000558`.

[42] M.E. Liggins, Chee-Yee Chong, I. Kadar, M.G. Alford, V. Vannicola, and S. Thomopoulos. Distributed fusion architectures and algorithms for target tracking. *Proceedings of the IEEE*, 85(1):95–107, 1997. doi: 10.1109/JPROC.1997.554211.

[43] K. H. Kim. Development of track to track fusion algorithms. In *Proceedings of 1994 American Control Conference - ACC '94*, volume 1, pages 1037–1041 vol.1, 1994. doi: 10.1109/ACC.1994.751905.

[44] Y. Bar-Shalom and X.R. Li. *Multitarget-multisensor Tracking: Principles and Techniques.* Yaakov Bar-Shalom, 1995. ISBN 9780964831209. URL `https://books.google.nl/books?id=GfOoMQEACAAJ`.

[45] C. Chong, K. Chang, and S. Mori. Comparison of optimal distributed estimation and consensus filtering. In *2016 19th International Conference on Information Fusion (FUSION)*, pages 1034–1041, 2016.

[46] L. F. O. Chamon, G. J. Pappas, and A. Ribeiro. Approximate supermodularity of kalman filter sensor selection. *IEEE Transactions on Automatic Control*, 66(1):49–63, 2021. doi: 10.1109/TAC.2020.2973774.

[47] H. Wang, K. Yao, and D. Estrin. Information-theoretic approaches for sensor selection and placement in sensor networks for target localization and tracking. *Journal of Communications and Networks*, 7(4):438–449, 2005. doi: 10.1109/JCN.2005.6387986.

[48] J. L. Williams, J. W. Fisher, and A. S. Willsky. Approximate dynamic programming for communication-constrained sensor network management. *IEEE Transactions on Signal Processing*, 55(8):4300–4311, 2007. doi: 10.1109/TSP.2007.896099.

[49] D. Zhang., M. Li, F. Zhang, and M. Fan. New gradient methods for sensor selection problems. *International Journal of Distributed Sensor Networks*, 15(3): 1550147719839642, 2019. doi: 10.1177/1550147719839642. URL `https://doi.org/10.1177/1550147719839642`.

[50] 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*, 9(8):1427–1439, 2015. doi: 10.1109/JSTSP.2015.2465304.

[51] 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)*, pages 1–8, 2016. doi: 10.1109/WIOPT.2016.7492941.

[52] J. Locke. The smart grid could hold the keys to electric vehicles @ONLINE. `https://innovationatwork.ieee.org/the-smart-grid-could-hold-the-keys-to-electric-vehicles/`, 2018.

[53] W. Koch. Adaptive parameter control for phased-array tracking. In Oliver E. Drummond, editor, *Signal and Data Processing of Small Targets 1999*, volume 3809, pages 444 – 455. International Society for Optics and Photonics, SPIE, 1999. doi: 10.1117/12.364042. URL `https://doi.org/10.1117/12.364042`.

Appendices

Appendix A

Static Scenarios

A.1 Comparison Scenarios

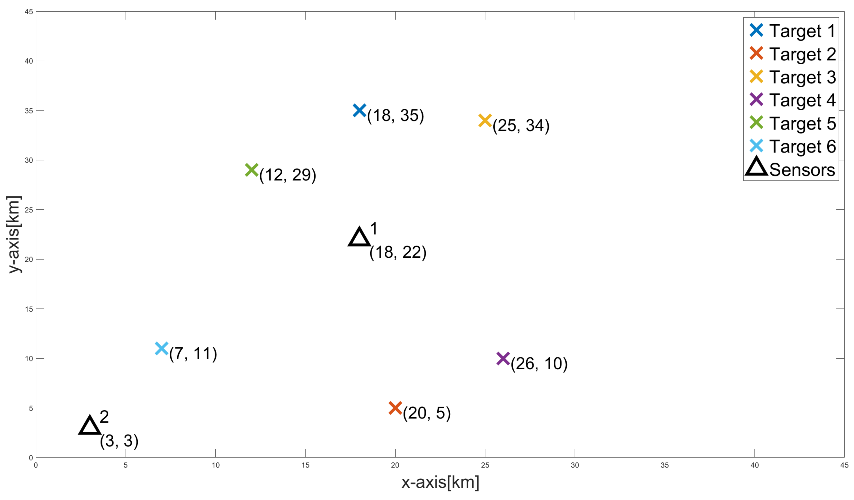


Figure A.1. Static Comparison scenario 1

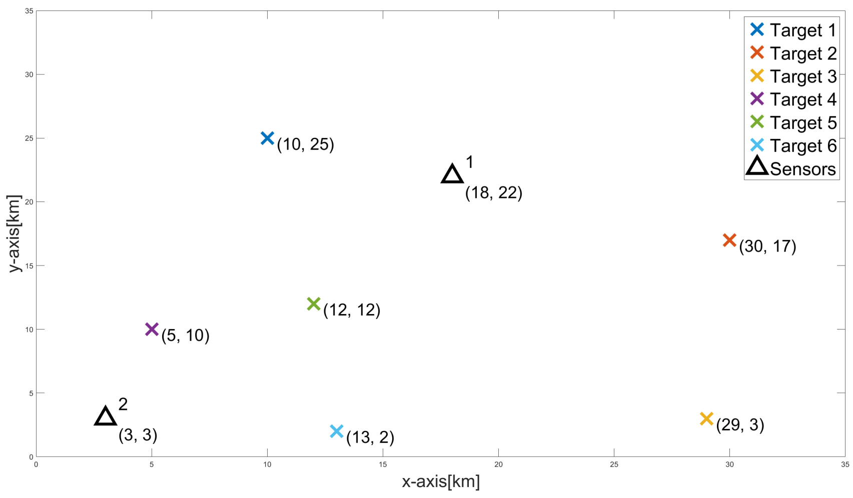


Figure A.2. Static Comparison scenario 2

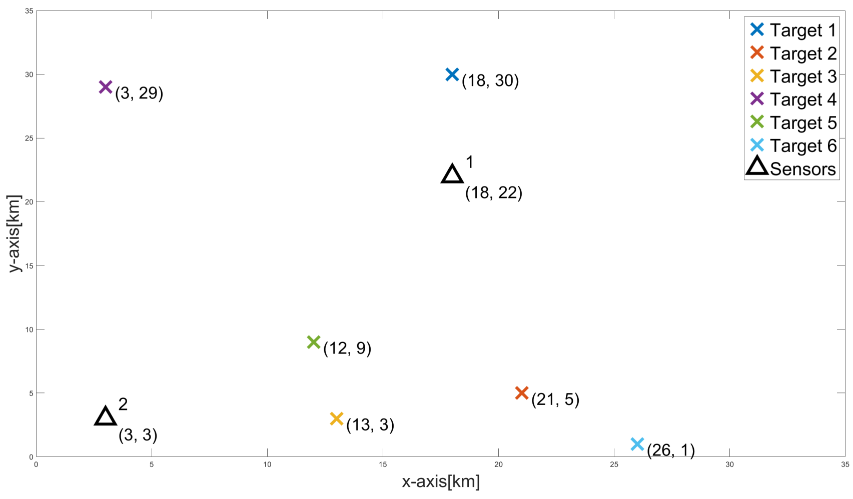


Figure A.3. Static Comparison scenario 3

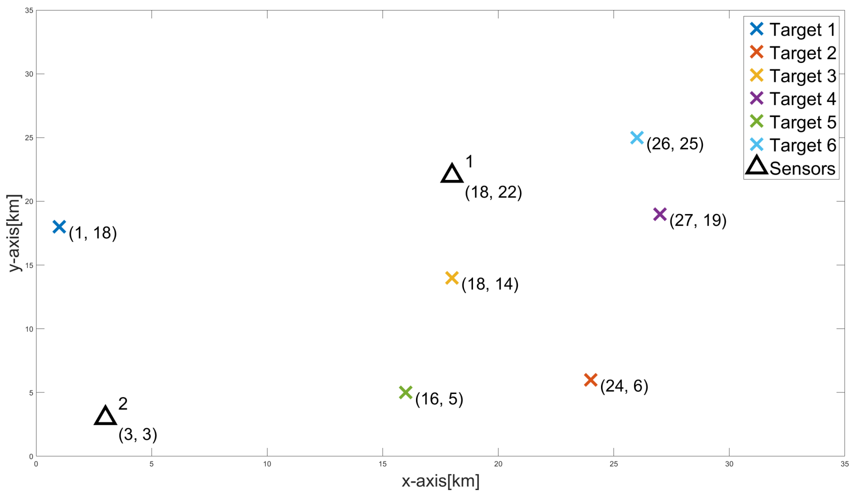


Figure A.4. Static Comparison scenario 4

Appendix B

Dynamic Scenarios

B.1 Comparison Scenarios

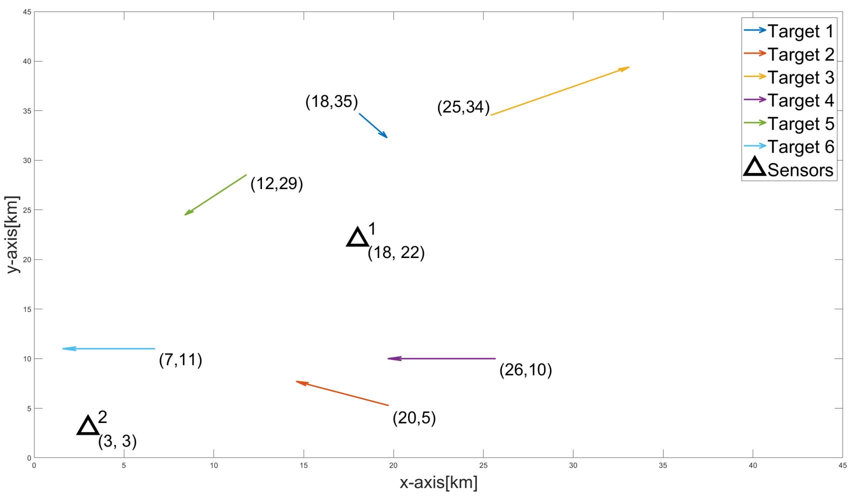


Figure B.1. Dynamic comparison scenario 1

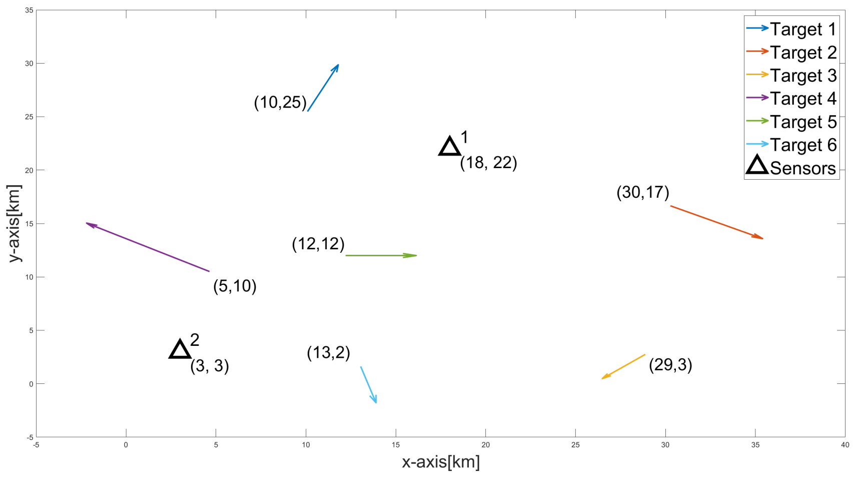


Figure B.2. Dynamic comparison scenario 2

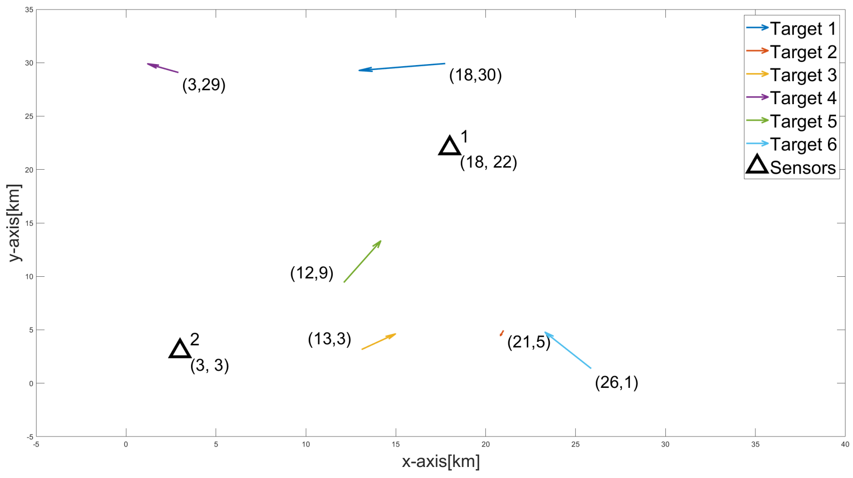


Figure B.3. Dynamic comparison scenario 3

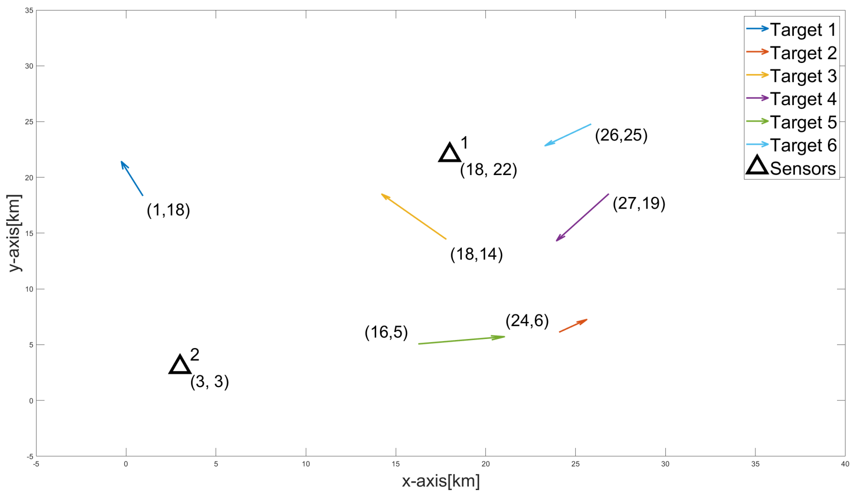


Figure B.4. Dynamic comparison scenario 4

Appendix C

Limiting Cases

C.1 Sensors at the Same Location

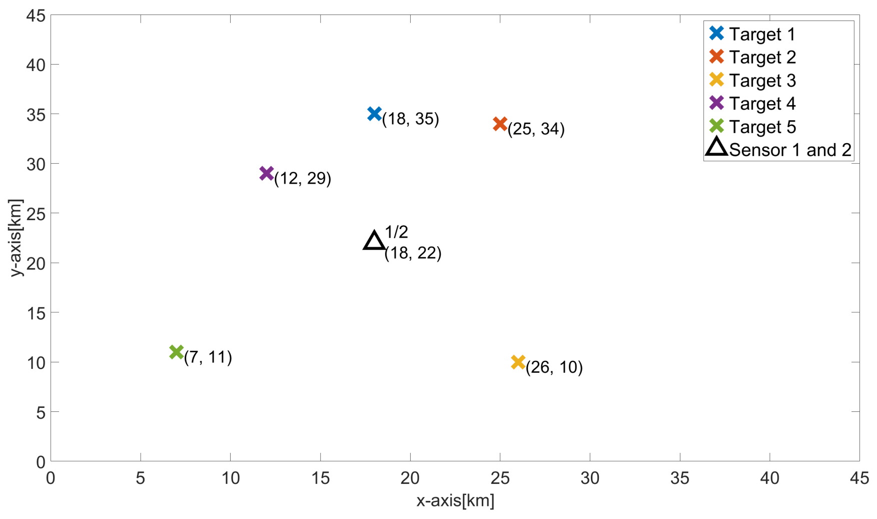


Figure C.1. Tracking scenario with overlapping sensors where target 2 is removed.

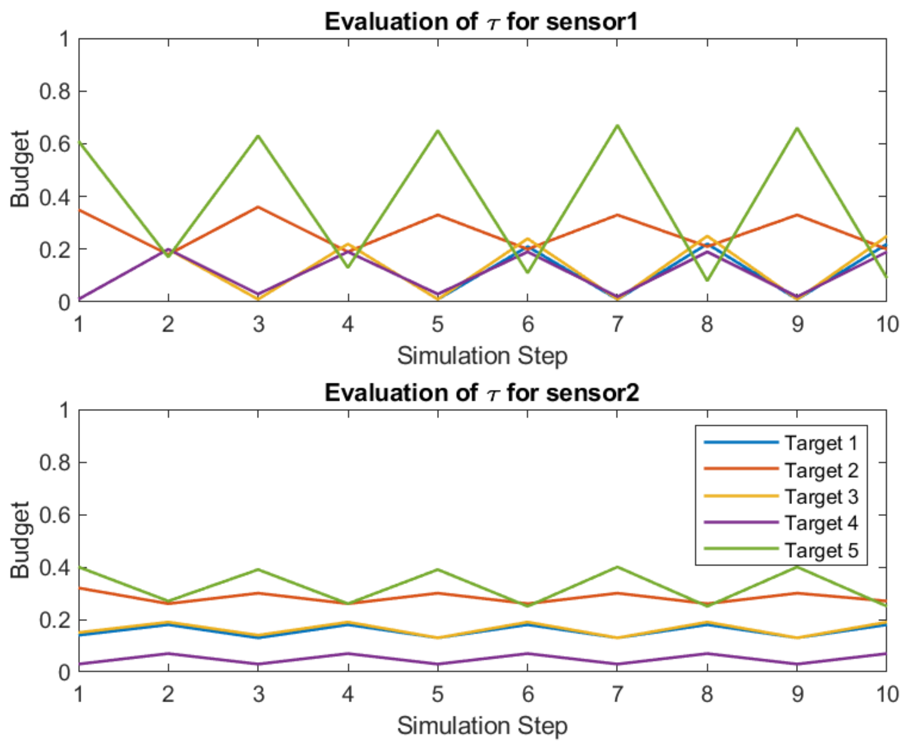


Figure C.2. Resulting budget allocation with overlapping sensors. Even when target 2 is removed a similar fluctuation between simulation steps can be observed.

C.2 Big Scenario

TABLE I. Overview of resulting budget allocations for targets one till five

Sensor	Target 1	Target 2	Target 3	Target 4	Target 5
1	10%	1%	9%	1%	27%
2	1%	39%	1%	29%	1%
3	1%	21%	13%	9%	1%
4	26%	1%	1%	1%	29%

TABLE II. Overview of resulting budget allocations for targets six till ten plus the total budget allocation per sensor

Sensor	sensor 6	Target 7	Target 8	Target 9	Target 10	Total Budget
1	1%	47%	1%	1%	1%	99%
2	15%	1 %	1%	1%	13%	102%
3	1%	1%	45%	6%	1%	99%
4	10%	3%	1%	1%	1%	101%

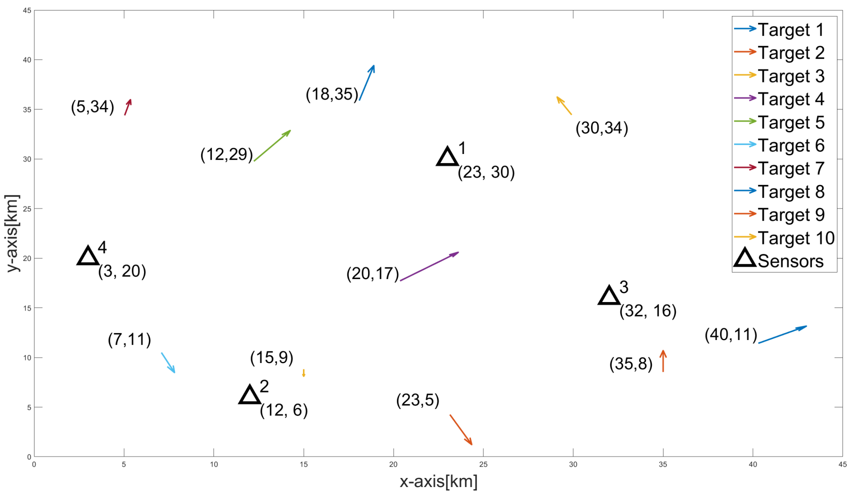


Figure C.3. Tracking scenario with a scenario involving four sensors and 10 targets.

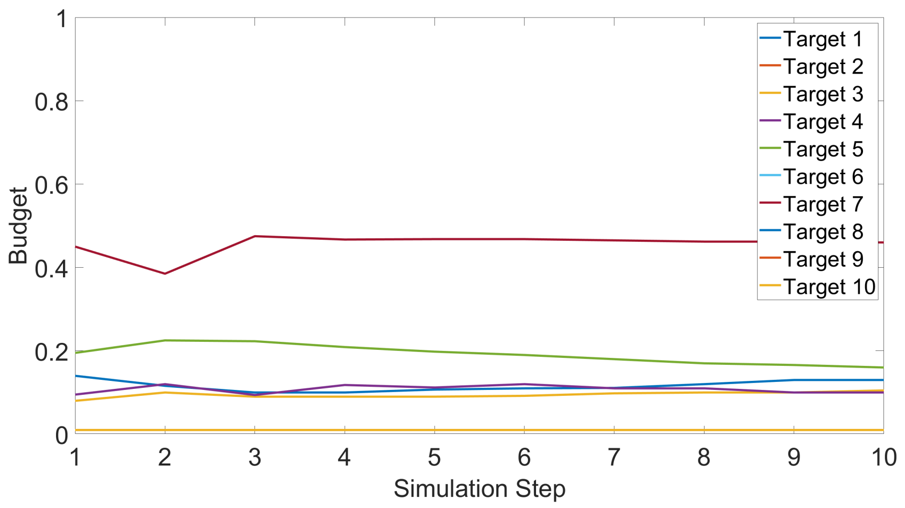


Figure C.4. Resulting budget allocation for sensor 1.

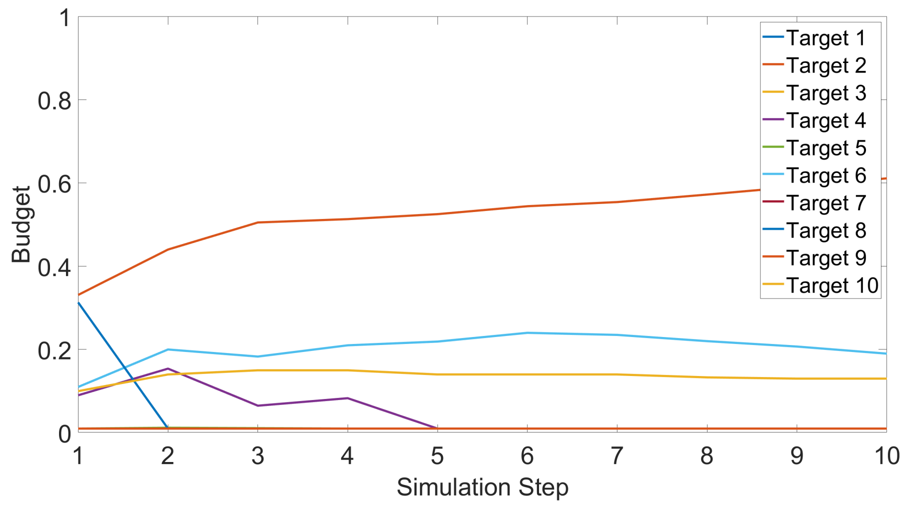


Figure C.5. Resulting budget allocation for sensor 2.

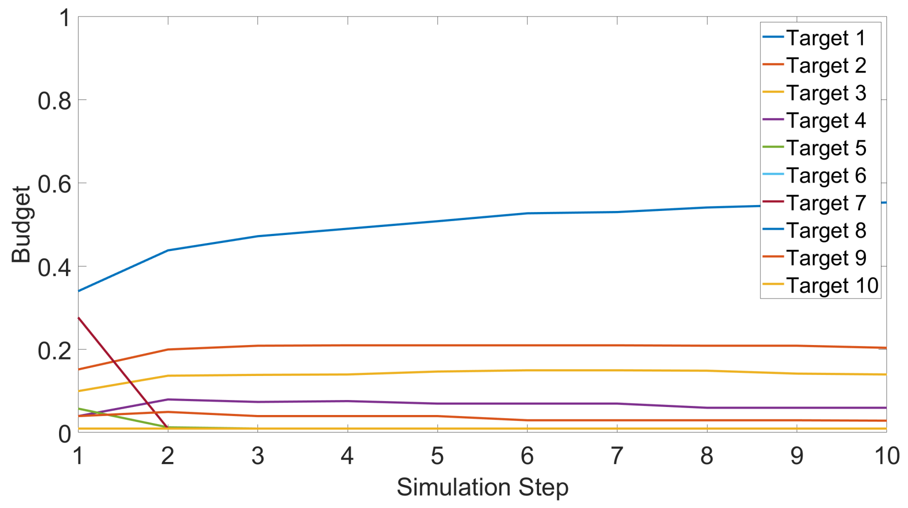


Figure C.6. Resulting budget allocation for sensor 3.

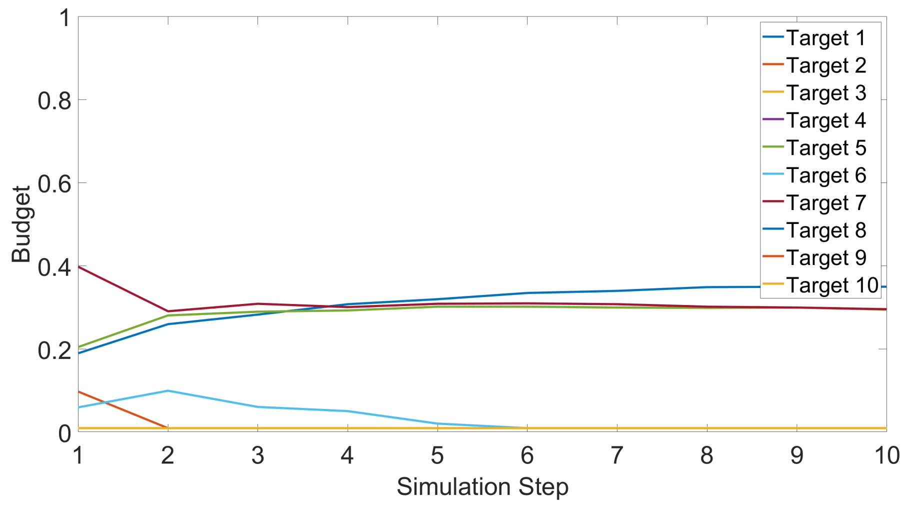


Figure C.7. Resulting budget allocation for sensor 4.

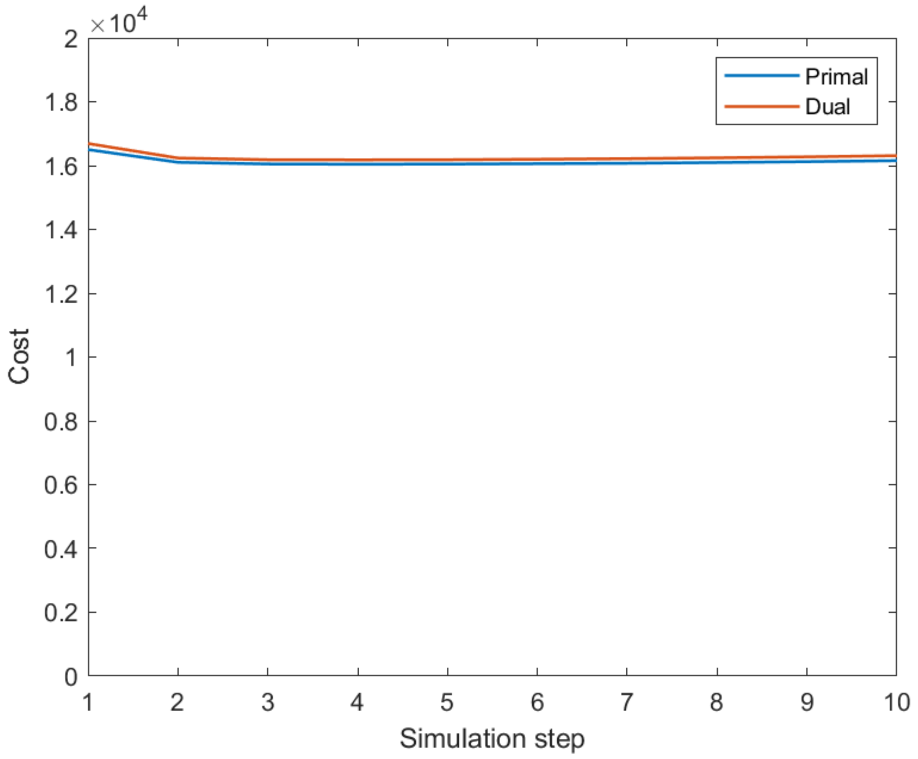


Figure C.8. Resulting cost of the bigger scenario.

Appendix D

Matlab Code

D.1 Main

```
function [runtime,T_out, tau_out] = Main_comparison(M,N,plotting)
% Setup
freqs = [12e9, 3e9];                                                        % Frequencies of radar systems [X band; S band]
lambda  = 3e8./freqs;
RCS = ([500; 200; 900; 800; 200; 1000;400]./lambda)';                                    % Radar Cross Sections(m2)
var_w = [12.9803   23.6233   15.1691   22.1923   16.9110   11.4533 9.2228];          % Process Noise

sensor_location = [14e3 30e3  32e3 3000;...
                   15e3 20e3  16e3 20e3];   % Sensor Locations
sensor_location = sensor_location(:,1:M);
% Simulation parameters
Sim_length = 40;
Budget_Update = 5;
[l_roll, n_roll] = deal(15,4);                                                    % policy rollout params
[B, precision, step, lambda] = deal([1,1], 0.05, 40, 1e2*ones(1,M));             % subgradient params
[T,tau, lambda_end, state] = deal(cell(Sim_length,1));



K = 10;
[f,f_dual, runtime] = deal(zeros(K,1));
[f_res_save, f_res_dual_save] = deal(zeros(Sim_length,K));
[~, ~, F, ~] = handler_functionsV3();
tau_prior = 1/N;
T_prior = 1;
for k = 1:10
    k
    lambda = 1e2*ones(1,M);
    cov_pred(1:M,1:N) = {eye(4)*100};
    [state_prior, tmp_pred] = set_stateV2(N);
    state_pred(1:M) = {tmp_pred};
    for it = 1:50
        for n = 1:N
            % True location of target w.r.t. sensor
            state_prior(:,n) = F(T_prior)*state_prior(:,n);
            for m = 1:M
                [angle, range, ~] = sensor_state(state_prior
                    (:,n), sensor_location(:,m));
                [meas{m,n}, R{m,n}] = input_kalman(RCS(m,n),
                    tau_prior, range, angle);

            end
            for m = 1:M
                [cov_pred{m,n}, state_pred{m}(:,n)] = ...
                    Ext_Kalman(R(:,n), meas(:,n), var_w(n),
                        cov_pred{m,n}, state_pred{m}(:,n),
                        sensor_location, T_prior);
            end
        end
    end
    state{1} = state_prior;
    tau_in = tau_prior.*ones(M,N);
    %% Main Algorithm
    % Note that f_res_dual is the sum of the individual cost
        of each sensor
    [f_res_dual, f_res] = deal(zeros(Sim_length,1));
    time = zeros(1,Sim_length);
    for Q = 1:Sim_length
        tic

        [T{Q}, tau{Q}, f_res_dual(Q), f_res(Q), lambda,
            Primal_cost{Q},Dual_cost{Q}] = ...
            Budget_allocation(cov_pred, state_pred, tau_in,
                sensor_location, var_w, RCS, lambda, l_roll,
                n_roll, step, B, precision, R, plotting);
        time(Q) = toc;
        tau_in = tau{Q};
        lambda_end{Q} = lambda;


        % Limited Budget
        if Q == 10
            B = [0.8 1];
            % Limited Angle
        elseif Q > 25
            if size(state_pred{2},2) > 5
                state_pred{2}(:,6:7) = [];
                cov_pred(2,6:7) = {[]};
                tau{Q}(2,6:7) = 0;
            end
            % Update tracks
            for n = 1:N
                meas = [];
                if n > 5              % Hard coded for
                    simplicity
                    M = 1;
                else
                    M = 2;
                end
                % Share info each x seconds.
                P = 1;
                while 1*P < Budget_Update
                    state{Q}(:,n) = F(T{Q}(1,n))*state{Q}(:,n)
                       ; % -> only works if T's are equal
                    for m = 1:M
                        [angle, range, ~] = sensor_state(state
                           {Q}(:,n), sensor_location(:,m));
                        [meas{m}(:,P), R{m,n,P}] =
                           input_kalman(RCS(m,n), tau{Q}(m,n),
                            range, angle);
                    end
                    P = P+1;
                end

                % Measurement fusion
                if M == 2
                    for m = 1:M
                        [cov_pred{m,n}, state_pred{m}(:,n)] =
                           ...
                            Ext_Kalman(squeeze(R(:,n,:)), meas
                               , var_w(n), cov_pred{m,n},
                               state_pred{m}(:,n),
                               sensor_location, 1);
                    end
                else
                    [cov_pred{m,n}, state_pred{m}(:,n)] = ...
                        Ext_Kalman(squeeze(R(1,n,:))', meas,
                           var_w(n), cov_pred{m,n}, state_pred
                           {m}(:,n), sensor_location(:,1), 1);
                end
                M = 2;
            end
        else
            % Update tracks
            for n = 1:N
                meas = cell(1,M);
                % Share info each x seconds.

                P = 1;
                while T{Q}(1,n)*P < Budget_Update
                    state{Q}(:,n) = F(T{Q}(1,n))*state{Q}(:,n)
                       ; % -> only works iif T's are equal
                    for m = 1:M
                        [angle, range, ~] = sensor_state(state
                           {Q}(:,n), sensor_location(:,m));
                        [meas{m}(:,P), R{m,n,P}] =
                           input_kalman(RCS(m,n), tau{Q}(m,n),
                            range, angle);
                    end
                    P = P+1;
                end

                % Measurement fusion
                for m = 1:M
                    [cov_pred{m,n}, state_pred{m}(:,n)] = ...
                            Ext_Kalman(squeeze(R(:,n,:)), meas
                               , var_w(n), cov_pred{m,n},
                               state_pred{m}(:,n),
                               sensor_location, T{Q}(m,n));
                end
            end
        end

        state{Q+1} = state{Q};
    end
    tau_out{k} = tau;
    f_res_save(:,k) = f_res;
    f_res_dual_save(:,k) = f_res_dual;
    f(k) = f_res(end);
    f_dual(k) = f_res_dual(end);
    Primal_cost_save{k} = Primal_cost;
    Dual_cost_save{k} = Dual_cost;
    runtime(k) = sum(time);
end
runtime_ave = sum(runtime)/K;

%% Save Relevant Values

% Set T in budget allocation as well
T_out = T{end};

%% Plotting
if plotting == 1
    close all
    figure(1)
    p1 = plot(round(f_res*100)/100,'linewidth',1.5); hold on;
    p2 = plot(round(f_res_dual*100)/100,'linewidth',1.5); hold
         on;
    xlabel('Cost');
    title('Function value(dual)');
    xlim([1 Sim_length])
    legend([p1 p2], 'Primal', 'Dual');



    tmp = cell2mat(lambda_end);
    figure(2)
    for m = 1:M
        plot(tmp(:,m),'linewidth',1.2);hold on;
    end
    title('Evaluation of Lambda')
    xlabel('Simulation step');
    xlim([1 Sim_length]);
    legend(arrayfun(@(lambda) sprintf('Lambda %d', lambda), 1:
       M, 'UniformOutput', false))



    tmp = cell2mat(tau);
    figure(3)
    for n = 1:N
        for m = 1:M
            tau_in = tmp(m:M:end,n);
            subplot(M,1,m)
            p(n) = plot(tau_in, 'linewidth',4); hold on;
            set(gca,'FontSize',20)
            title(['Sensor ',num2str(m)],'FontSize', 34);
            xlabel('Simulation Step','FontSize', 34);
            ylabel('Budget','FontSize', 34);
            xlim([1 Sim_length]);
            ylim([0 .7])
        end
    end
    subplot(2,1,1)
    xline(10,'--'); hold on
    xline(26,'--'); hold on
    subplot(2,1,2)
    xline(10,'--'); hold on
    xline(26,'--'); hold on
    h = legend(arrayfun(@(target) sprintf('Target %d', target)
        , 1:N, 'UniformOutput', false),'Position',[0.946 0.6
        0.015 0.3]);
    legend('Orientation','Vertical')
    hl = findobj(h,'type','line');
    legend('boxon')
    set(h,'FontSize',26);
    set(hl,'LineWidth',2.5);






    state_in = cell2mat(state);
    state_x = state_in(1:4:end,:);
    state_y = state_in(2:4:end,:);

    figure(4)
    for n = 1:N
%           p(n) = plot(state_x(:,n)/1000, state_y(:,n)/1000,'x
    ','linewidth',5,'MarkerSize',2); hold on
        start = [state_x(1,n)/1000,state_y(2,n)/1000];
        ending = [state_x(end,n)/1000,state_y(end,n)/1000];
        diff = ending-start;
        h(n) = quiver(start(1),start(2),diff(1),diff(2),0,'
            MaxHeadSize',2.25/norm(diff),'linewidth',2,'
            MarkerSize',20); hold on
        if(h(n).UData < 0)
            textString = sprintf('(%.f,%.f)', state_x(1,n)
                /1000, state_y(1,n)/1000);
            text(state_x(1,n)/1000+0.2, state_y(1,n)/1000-1,
                textString, 'FontSize', 25);
        else
```

```
            textString = sprintf('(%.f,%.f)', state_x(1,n)/1000, state_y(1,n)/1000);
            text(state_x(1,n)/1000-3, state_y(1,n)/1000+1.2, textString, 'FontSize', 25);
        end
    end
    for m = 1:M
       p(N+m) = plot(sensor_location(1,m)./1000, sensor_location(2,m)./1000,'k^','linewidth',4,'MarkerSize',23); hold on;
       textString = sprintf('(%.f, %.f)', sensor_location(1,m)./1000, sensor_location(2,m)./1000);
       text(sensor_location(1,m)./1000+0.55, sensor_location(2,m)./1000-1.2, textString, 'FontSize', 25);hold on;
       textString = sprintf('%d', m);
       text(sensor_location(1,m)./1000+0.55, sensor_location(2,m)./1000+1.2, textString, 'FontSize', 28);hold on;
    end
    xlim([0 45]);
    ylim([0 45]);
    xticks([0 5 10 15 20 25 30 35 40 45])
    yticks([0 5 10 15 20 25 30 35 40 45])


    xlabel('x-axis[km]', 'FontSize', 50);
    ylabel('y-axis[km]', 'FontSize', 50);
    set(gcf, 'Position', get(0, 'Screensize'));
    legendstrings = cell(1, N+1);
    for target = 1:N
        legendstrings{target} = sprintf('Target %d', target);
    end
    for sensor = N+1:N+1
        legendstrings{sensor} = sprintf('Sensors');
    end
    h = legend(legendstrings);
    set(h,'FontSize',30);
%     title('Sensor Locations & Tracks');

end


end
```

D.2 Budget allocation algorithm

```
function [T_out, tau_out, f_res_dual_out,f_res_out, lam,
   Primal_cost, Dual_cost] = ...
                      Budget_allocation(cov_pred, state_pred,
                         tau_old, s_loc, var_w, RCS, lam, l_roll
                         , n_roll, step, B, precision,R,
                         plotting)

    [N,M] = deal(size(cov_pred,2), size(lam,2));
    [info_line, f_res_old] = deal([],0);
    C = zeros(M,N);
    y = size(state_pred{2},2);
    iter = 1;
    while 1
        for m = 1:M
           for n = 1:size(state_pred{m},2)
               if n > y
                   state_in = state_pred{1}(:,n);
                   RCS2 = [];
                   s_loc2 = [];
                   tau2 = [];
                   M = 1;
               else
                   state_in = [state_pred{1}(:,n),state_pred
                      {2}(:,n)];
                   M = 2;
                   tau2 = tau_old([1:m-1 m+1:end],n);
                   s_loc2 = s_loc(:,[1:m-1 m+1:end]);
                   state_tmp = state_in(:,[1:m-1 m+1:end]);
                   RCS2 = RCS([1:m-1 m+1:end],n);
               end
               [T{iter}(m,n), tau{iter}(m,n), Primal_cost(m,n)
                  , Dual_cost(m,n)] = ...
                      Policy_rollout(M,cov_pred{m,n}, state_pred
                         {m}(:,n), state_in, tau2, s_loc(:,m),
                         s_loc2, RCS(m,n),RCS2, var_w(n), lam(m)
                         , l_roll, n_roll);
               C(m,n) = Cost(T{iter}(m,n), var_w(n),
                  state_pred{m}(:,n), cov_pred{m,n}, R(:,n),
                  s_loc);
            end
       end
%         tau_old = tau{iter};
       [lambda(:,iter), lam, grad(:,iter)] = SubGradient(ones(
          M,N),M,tau{iter},step,B,lam);



        % Sum Cost & Compute Budget
        C_sum = sum(C,'all')/(M*N);
                                          % average Cost
        f_res(iter) = C_sum;
        f_res_dual(iter) = f_res(iter) + sum(lam*(tau{iter}./
           ones(M,N))) - sum(lam.*B);
        budget_tot(iter, :) = sum(tau{iter}./ones(M,N),2);
```

```matlab
        budget_ind{iter} = tau{iter}./ones(M,N);

        primal_plot(iter) = sum(sum(Primal_cost));
        dual_plot(iter) = sum(sum(Dual_cost)) - l_roll*sum(lam.*B);

        if  max(abs(grad(:,iter)),[],'all')  <= precision %&& max(abs(f_res_dual(iter)-f_res_old)) <= precision
            fprintf('\n')
            break
        end

        %print info & prepare next iteration
        del_line = sprintf(repmat('\b',1,length(info_line)));
        info_line = sprintf(['Iteration: %i -->' ' df = ' repmat('%f ', 1, 1) ' grad = ' repmat('%f ', 1, M) 'Lam = ' repmat('%f ', 1, M)],iter,f_res_dual(:,iter)-f_res_old,abs(grad(:,iter)),lambda(:,iter));
        fprintf([del_line info_line])
        f_res_old = f_res_dual(1,iter);
        iter = iter + 1;
    end
    tau_out = tau{end};
    T_out = T{end};
    f_res_dual_out = f_res_dual(end);
    f_res_out = f_res(end);
end
```

D.3 Policy Rollout

```matlab
function [T, tau, primal_out, dual_out] = Policy_rollout(M, P_old, state_old, state2, tau_old, s_loc1,s_loc2, RCS, RCS2, var_w, lambda, l_roll, n_roll)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Name: Bas van der Werk
% Date: 25-2-2021
% Description: Compute a policy rollout related to a single target and a
% single sensor
%
%     Inputs:
%     Policy_rollout(M,P_old, state_old, state2, tau_old, s_loc1,s_loc2, RCS, RCS2, var_w, lambda, l_roll, n_roll)
%
%     M:          Number of sensors                                      RCS:    Radar cross section
%     P_old:      Last known error covariance                    RCS2:   Radar cross section of other sensors
%     state_old:  Last known predicted(!) state                var_w:  variance of Process Noise
%     state2:     Last known predicted states of other sensors         lambda: Lagrangian Multiplier
%      tau_old:   Last known revisit times of the other sensors      l_roll: Length of rollout
%      s_loc1:    sensor location of selected senors      n_roll: number of rollouts
%      s_loc2:    sensor location of other sensors
%
%      output: T: Selected revisit times - MxN
%              tau: Selected dwell times - MxN;
%              acc_cost: accumulated average cost - Ax1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


    [~, ~, F, ~] = handler_functionsV3();
    actions = set_actionsV2(0.01, 1);
    L = size(actions,2);
    acc_cost = zeros(1,L);
    primal = zeros(1,L);
    T = 1;

    % Run rollout for every possible action
    parfor it_a=1:L
        % Average over multiple iterations
        for it_b = 1:n_roll
            chosen_action = actions(:,it_a);
            cov_pred = P_old;
            state = state_old;
            state_pred = state;
            state_in = state2;
            tau_in = tau_old;
            for it_c = 1:l_roll
                % Update selected sensor
                [angle, range, ~] = sensor_state(state, s_loc1);
                [meas, R] = input_kalman(RCS, chosen_action(2), range, angle);

                [cov_est, state_est] = ...
                            Kalman_est(R, meas, cov_pred, state_pred, s_loc1);


                % Update state & covariance using info of other sensors(tau_in).
                for m = 1:M-1
                    [angle, range, ~] = sensor_state(state_in(:,m), s_loc2(:,m));
                    [meas, R] = input_kalman(RCS2(m), tau_in(m), range, angle);

                    [cov_est, state_est] = ...
                            Kalman_est(R, meas, cov_est, state_est, s_loc2(:,m));
                end
                % assume T is equal for all sensors
                state = F(T)*state;
                state_in = F(T)*state_in;

                [C, cov_pred, state_pred] = Kalman_pred(T,
                   var_w, cov_est, state_est);


                % Relaxation for selected sensor
                % only
                Relaxation = lambda*chosen_action(2)./
                   chosen_action(1);
                acc_cost(it_a) = acc_cost(it_a) + C +
                   Relaxation;
                primal(it_a) = primal(it_a) + C;
            end
        end
        acc_cost(it_a) = acc_cost(it_a)/n_roll;
        primal(it_a) = primal(it_a)/n_roll;
    end

    A = acc_cost;

    [~,I] = min(A);
    tau = actions(2,I);
    T = actions(1,I);
    primal_out = primal(I);
    dual_out = acc_cost(I);
end
```

D.4 Subgradient

```
function [lambda,lam_out,grad] = SubGradient(T,M,tau,steps,B,
   lam_in)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Name: Bas van der Werk
% Date: 25-2-2021
% Description: Compute subgradient to update Lagrangian
   Multipliers
%
%     Inputs:
%     SubGradient(T,M,tau,steps,B,lam_in)
%     T: Revisit time
%     M: Number of sensors
%     tau: Dwell times
%     steps: Stepsize
%     B: Maximum Budget
%     lam: Last known Lambda
%
%     output: lambda: used for plotting - 1xM
%             lam_out: used for algorithm - 1xM;
%             grad: Gradient - 1xM
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    for m = 1:M
        grad(m) = sum(tau(m,:)./T(m,:))-B(m);
    end

    % Increment lambda based on the step size
    lam_out = lam_in + steps.*grad;
    lambda = lam_out; % Save lambdas for plotting
end
```

D.5 Additional Functions

```
function [P_est, state_est] = Kalman_est(R, meas, P, state,
    s_loc)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Name: Bas van der Werk
% Date: 25-2-2021
% Description: Compute Kalman Estimate
%
%     Inputs:
%     Kalman_est(R, meas, P, state, s_loc)
%     R: covariance of observation noise
%     meas: Measurement of sensor(s)
%     P: Error covariance
%     state: Current state of target
%     s_loc: sensor locations
%
%     output: P_est: Estimated covariance - 4x4
%             state_est: Estimated state of target - 4x1;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    [h,H,~,~] = handler_functionsV3();
    I = eye(4);
    [h, H, ~, ~] = handler_functionsV3();
    % Estimation step
    K = P*H(state, s_loc)'/(H(state, s_loc)*P*H(state, s_loc)'
        + R);
    P_est = (I - K*H(state, s_loc))*P;
    state_est = state + K*(meas - h(state, s_loc));
end
```

```
function [C, P_pred, state_pred] = Kalman_pred(T, var_w, P_est
    ,state_est)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Name: Bas van der Werk
% Date: 25-2-2021
% Description: Compute Kalman Prediction
%
%     Inputs:
%     Kalman_pred(T, var_w, P_est,state_est)
%     T: Revisit time
%     var_w: Process noise
%     P_est: Estimated covariance
%     state_est: Estimated state of target
%
%     output: C: Cost - Scalar
%             P_pred: Predicted covariance - 4x4
%             state_pred: Predicted state of target - 4x1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```matlab
    [~, ~, F, Q] = handler_functionsV3();
    P_pred = F(T)*P_est*F(T)' + Q(T,var_w);
    state_pred = F(T)*state_est;
    C = P_pred(1,1)+P_pred(2,2) + 1000/T.^2;
end
```

```matlab
function [P_pred, state_pred] = Ext_Kalman(R, meas, var_w, P,
    state, s_loc, T)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Name: Bas van der Werk
% Date: 25-2-2021
% Description: Compute recursive update of the Global kalman
    filter
% based on measurement and covariance of sensor
%
%     Inputs:
%     Ext_Kalman(R, meas, var_w, P, state, s_loc)
%     R: covariance of observation noise
%     meas: Measurement of sensor(s)
%     var_w: Process noise
%     P: Error covariance
%     state: Current state of target
%     s_loc: sensor locations
%
%     output: P_pred: Predicted covariance - 4x4
%             state_pred: Predicted state of target - 4x1;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    [h,H,F,Q] = handler_functionsV3();
    I = eye(4);
    M = size(R,1);
    updates = size(R,2);            % R defines how many track
        updates there are
    state_est = state;
    P_est = P;

    for u = 1:updates
        for m = 1:M
            if ~isempty(R{m,u})
                K = P_est*H(state_est, s_loc(:,m))'/(H(
                    state_est, s_loc(:,m))*P_est*H(state_est,
                    s_loc(:,m))' + R{m,u});
                P_est = (I - K*H(state_est, s_loc(:,m)))*P_est
                    ;
                state_est = state_est + K*(meas{m}(:,u) - h(
                    state_est, s_loc(:,m)));
            end
        end
        P_pred = F(T)*P_est*F(T)' + Q(T,var_w);
        state_pred = F(T)*state_est;
    end
end
```

```matlab
function [h, H, F, Q] = handler_functionsV3()
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Name: Bas van der Werk
% Date: 25-2-2021
% Description: Define helper functions for tracking process. Note that a
% compensation is made in the linearization based on the location of the
% sensor.
%
%     Inputs:
%     handler_functionsV3()
%
%     output: h: Measurement model
%             H: Linearized observation model;
%             F: State transition model
%             Q: Covariance of the process noise
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
h = @(s, s_loc)[sqrt((s(1)-s_loc(1))^2+(s(2)-s_loc(2))^2); atan2(s(2)-s_loc(2),s(1)-s_loc(1))];

H = @(s, s_loc) [(s(1)-s_loc(1))/((s(1)-s_loc(1))^2+(s(2)-s_loc(2))^2)^(1/2) (s(2)-s_loc(2))/((s(1)-s_loc(1))^2+(s(2)-s_loc(2))^2)^(1/2) 0 0;
          -(imag((s(1)-s_loc(1)))+real((s(2)-s_loc(2))))/((imag((s(1)-s_loc(1)))+real((s(2)-s_loc(2))))^2+(imag((s(2)-s_loc(2)))-real((s(1)-s_loc(1))))^2)...
              -(imag((s(2)-s_loc(2)))-real((s(1)-s_loc(1))))/((imag((s(1)-s_loc(1)))+real((s(2)-s_loc(2))))^2+(imag((s(2)-s_loc(2)))-real((s(1)-s_loc(1))))^2) 0 0];
H_old = @(s) [(s(1))/((s(1))^2+(s(2))^2)^(1/2) (s(2))/((s(1))^2+(s(2))^2)^(1/2) 0 0;
          -(imag((s(1)))+real((s(2))))/((imag((s(1)))+real((s(2))))^2+(imag((s(2)))-real((s(1))))^2)...
              -(imag((s(2)))-real((s(1))))/((imag((s(1)))+real((s(2))))^2+(imag((s(2)))-real((s(1))))^2) 0 0];


% Predict
F = @(T)[1, 0,  T,0;0, 1,  0,   T; 0, 0,  1,   0;0, 0,  0, 1];
Q = @(T, var_w)[T^4/4, 0,        T^3/2,0;0,        T^4/4, 0,  T^3/2;
              T^3/2, 0        T^2,  0;0,        T^3/2, 0,        T^2]*var_w;

end
```

```matlab
function [measurement,R] = input_kalman(RCS,tau,range,angle)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Name: Bas van der Werk
% Date: 25-2-2021
% Description: Take measurement & compute a covariance R based on RCS,
% dwell times, range and angle.
%
%     Inputs:
%     input_kalman(RCS,tau,range,angle)
%     RCS: radar cross section
%      tau: Dwell times
%      range: Distance to target w.r.t. sensor
%      angle: Angle to target w.r.t. sensor
%
%      output: measurement: measurement - 2x1
%              R: Covariance of observation noise - 4x4;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    std_r0 = 25;
    std_th0 = sqrt(4e-4);

    SNR = calc_snr(0,RCS,tau,range); % delta b = 0!
    var_range = (std_r0./sqrt(SNR)).^2;
    var_angle = (std_th0./sqrt(SNR)).^2;
    R = diag([var_range var_angle]);


    % Take n samples and average
    measurement = [normrnd(range,sqrt(R(1,1))); normrnd(angle,
        sqrt(R(2,2)))];
end
```

```
function [angle, range, vel] = sensor_state(state,
   sensor_location)
    %     [angle, range, vel] = sensor_state(state,
        sensor_location)
    angle = atan2(state(2)-sensor_location(2),state(1)-
        sensor_location(1))';
    range = sqrt((state(1)-sensor_location(1)).^2+(state(2)-
        sensor_location(2)).^2)';
    vel = (state(1)*state(3)+state(2)*state(4))./sqrt(state(1)
        .^2+state(2).^2);
end
```

```
function [A] = set_actionsV2(discretization,M)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Name: Bas van der Werk
% Date: 25-2-2021
% Description: Define a discrete action space based on the
   number of
% sensors and a custom discretization step
%
%      Inputs:
%      set_actionsV2(discretization,M)
%      discretization: steps
%      M: Number of sensors
%
%      output: A: Discrete action space - 2M x y
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    tau = discretization:discretization:0.9;
    L = length(tau);
    T = 1*ones(1,L^M);                                       %
        Fixed T for all sensors
    A = [];
    for i = 1:M
        tmp = repelem(tau, L^(M-i));
        actions = repmat(tmp, 1, L^(i-1));
        A = cat(1, A, T, actions);
    end
end
```