Multi-Task Sensor Resource Balancing Using Lagrangian Relaxation and Policy Rollout

Schöpe, M.I.; Driessen, Hans; Yarovoy, Alexander

DOI
10.23919/FUSION45008.2020.9190546

Publication date
2020

Document Version
Accepted author manuscript

Published in
2020 23rd International Conference on Information Fusion (FUSION)

Multi-Task Sensor Resource Balancing Using Lagrangian Relaxation and Policy Rollout

Max Ian Schöpe, Hans Driessen and Alexander Yarovoy
Microwave Sensing, Signals and Systems (MS3)
Delft University of Technology
Delft, The Netherlands
{m.i.schope, j.n.driessen, a.yarovoy}@tudelft.nl

***Abstract*—The sensor resource management problem in a multi-object tracking scenario is considered. In order to solve it, a dynamic budget balancing algorithm is proposed which models the different sensor tasks as partially observable Markov decision processes. Those are being solved by applying a combination of Lagrangian relaxation and policy rollout. The algorithm converges to a solution which is close to the optimal steady-state solution. This is shown through simulations of a two-dimensional tracking scenario. Moreover, it is demonstrated how the algorithm allocates the sensor time budgets dynamically to a changing environment and takes predictions of the future situation into account.**

***Index Terms*—Sensor Resource Management, Lagrangian Relaxation, Partially Observable Markov Decision Process, Policy Rollout**

I. Introduction

In recent years, the degrees of freedom of sensors like radars, lidars or cameras have increased tremendously due to technological improvements in hardware and software [1]. For radar, the development has led to multi-function radar (MFR) systems that can handle a variety of different tasks, often practically in parallel. This has been made possible e.g by using phased array antennas, applying digital beamforming (DBF) and the digital generation of a large variety of waveforms [2]. As a result, sensors are becoming more flexible and an adaption to the environment during run-time becomes possible. This paper focuses mainly on single adaptive sensors like MFR systems or cameras. However, our approach is generic and is in principle applicable to all kinds adaptive sensor systems.

A. Sensor Resource Management

One of the most challenging problems of modern adaptive sensors is how to control them while they are in operation. For radar, a lot of research aims for automatic adaption to changing situations, like changing weather conditions, interference or quickly changing numbers of observed objects (see e.g. [3]–[6]). The automatic adaptation of radar resources to a variable environment or mission is often called radar resource management (RRM) or more generally speaking sensor resource management (SRM). Possible applications of these management approaches include automotive scenarios, such as autonomous driving or traffic monitoring, (maritime) surveillance and air traffic control, for example.

Much of the work on SRM and RRM presented in literature (see for example the overview by Hero and Cochran in [1]) focuses on a single task, e.g. keeping a constant track quality even under objects maneuvers. This usually means regulating the time budget spent on a certain task. However, sensors like MFR systems are mostly operating at their sensor time budget limit, so increasing the budget for one task means at the same time decreasing the budget of the others. This leads to a deterioration of their performance. In this work which continues the research in [7], the SRM problem is seen as a budget or performance balancing act over the various tasks. We strive to formulate the problem as an optimal control problem and solve it in an approximately optimal manner with respect to the optimal steady-state solution. It has been suggested that a truly optimal solution could possibly lead to a significant improvement of the performance of adaptive sensors [8].

B. Markov Decision Processes in Resource Management

Markov decision processes (MDP) and partially observable Markov decision processes (POMDP) are an attractive framework for SRM problems. They allow to formulate a dynamic problem with different states in which the optimal actions can be found through optimizing a cost or reward function. In addition to that, they also allow a prediction of the future state of the tasks.

It is possible to apply this framework to single tasks, which has been done for instance by Charlish and Hoffmann in [9] or by Krishnamurthy in [10]. The former applied policy rollout, while the latter used a stochastic dynamic programming algorithm. Both methods optimize the time between consequent measurement operations. Other single object POMDP approaches have been published e.g. in [11] and [12].

Another problem formulation is to apply a constrained (PO)MDP, meaning that there is a limit on the available resources or budgets that can be distributed over all tasks. Possible applications are sensor networks or single sensors with multiple tasks. In order to deal with the potential large computational complexity in these problems, Lagrangian relaxation (LR) has been suggested to decouple the main optimization problem into smaller and easier to solve sub-problems. One LR approach for sensor networks has been published by Williams et al. in [13]. Some notable LR approaches for multi-task radar scenarios are e.g. [14] by Wintenby and Krishnamurthy and

[15] by White and Williams. Wintenby and Krishnamurthy apply a Markov chain consisting of performance states for each tracking task and solve it with a combination of LR and approximate dynamic programming. We believe that in order to include other kinds of tasks such as classification, a more general framework than Markov chains needs to be applied. White and Williams assume that the state space is discretized and a fully observable MDP can be defined and solved by the use of dynamic programming. This approach requires perfect measurements, which is too big of a simplification in our eyes. We see the potential in such solutions, but we believe that a POMDP is needed to properly describe the problem. Other publications that suggest the use of POMDPs can for example be found in [16]–[19].

C. Previous Contribution

Previously, we have already shown the optimal balancing of sensor budget in a linear time invariant (LTI) setting [7]. We introduced an optimal steady-state budget balancing (OSB) algorithm which uses LR to distribute the available budgets over multiple independent tasks. In this simplified setting, an optimal solution can easily be calculated. Since most problems cannot be described in an LTI setting, the results in [7] just give a small indication of what is possible with this kind of approach.

D. Novelty in this Paper

In this paper, we extend the previous approach to generic dynamical problems by taking advantage of the POMDP framework and introduce an approximately optimal dynamic budget balancing (AODB) algorithm. It is approximately optimal with respect to the steady-state error-covariance of a Kalman filter (see e.g. [20]). The SRM problem is solved non-myopically by using an online Monte Carlo technique called policy rollout, which stochastically predicts the future. This technique has been described e.g. by Bertsekas et al. in [21]–[23]. In this paper, we will demonstrate a way to apply this technique to a simplified tracking scenario. We noticed that an overall solution to the SRM problem has not been presented so far, so our long term goal is to develop and evaluate generic SRM approaches that can be applied in surveillance applications, such as tracking, classification and detection.

In practice, it is important to define an operationally relevant cost function to efficiently make use of these techniques. This topic will not be covered in this paper. An example of an operationally relevant cost function has been discussed by Katsilieris et al. in [24].

E. Structure of the Paper

The rest of the paper is structured as follows. Section II defines the problem as a constrained optimization problem in a POMDP framework, while section III explains how the combination of LR and policy rollout can be applied to that problem. In section IV we describe a two-dimensional tracking scenario that is solved using our proposed method. We show the corresponding simulation results in section V, while section VI comprises the conclusions.

II. SRM Problem Definition

In this section, the considered SRM problem is introduced. For illustration purposes, it is based on a two-dimensional tracking scenario. Firstly, the motion model, the measurement model and possible tracking algorithms are introduced. Secondly, the sensor budget optimization problem is formulated.

A. Motion Model

At every moment in time k, each object that moves within this model can be assigned a state based on its position in x and y direction within a Cartesian coordinates system. For object n this state is defined as

$$\boldsymbol{s}_k^n = [x_k^n \quad y_k^n \quad \dot{x}_k^n \quad \dot{y}_k^n]^T, \tag{1}$$

where x_k^n, y_k^n and $\dot{x}_k^n$, $\dot{y}_k^n$ are the position and velocity of object n in x and y respectively. It assumed that the object is tracked every time step k with sensor action $\boldsymbol{a}_k^n \in \mathbb{R}^m$, where m is the amount of adjustable action parameters. The future state can be predicted at every time step k following a certain function

$$\boldsymbol{s}_{k+1}^n = f\left(\boldsymbol{s}_k^n, \boldsymbol{w}_k^n; \boldsymbol{a}_k^n\right), \tag{2}$$

where $k+1$ is the next following state and $\boldsymbol{w}_k^n \in \mathbb{R}^4$ is the maneuverability noise for object n at time k. The state evolution equation (2) directly defines the evolution probability density function which is given as

$$p\left(\boldsymbol{s}_{k+1}^n | \boldsymbol{s}_k^n; \boldsymbol{a}_k^n\right). \tag{3}$$

B. Measurement Model

We assume a sensor that can make noisy observations of certain elements of the state $\boldsymbol{s}_k^n$ and does so at every time step k. This could comprise e.g. range measured by a radar sensor or x and y position observed by a camera. A typical measurement of object n at time step k can therefore be characterized by the measurement function

$$\boldsymbol{z}_k^n = h\left(\boldsymbol{s}_k^n, \boldsymbol{v}_k^n; \boldsymbol{a}_k^n\right), \tag{4}$$

where $\boldsymbol{v}_k^n \in \mathbb{R}^p$ is the measurement noise for object n and p is the amount of measurement parameters. The measurement equation (4) directly defines the measurement probability density function which can be written as

$$p\left(\boldsymbol{z}_k^n | \boldsymbol{s}_k^n; \boldsymbol{v}_k^n\right). \tag{5}$$

C. Tracking Algorithm

The tracking algorithm should aim at computing the posterior density. For linear systems, a Kalman filter can be adopted as exact solution. For non-linear systems, approximate algorithms need to be considered, e.g. particle filter algorithms.

D. Budget Optimization Problem

As mentioned in section I, the sensor is assumed to have a limited maximum budget of any kind, which will be called θ_{max}. For action $\boldsymbol{a}^n$ that is executed for each task n, a certain amount of θ_{max} is required. Typical actions are time, frequency or energy allocations for instance. In an overload situation, all the current tasks of the sensor require more budget than is available at the moment. In that case, the available budget has to be distributed over the tasks in a way that minimizes the cost which is related e.g. to the uncertainty of the current situation. Therefore, the SRM problem is formulated as a minimization problem.

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

$$\begin{aligned} \underset{\boldsymbol{a}}{\text{minimize}} \quad & \sum_{n=1}^{N} c(\boldsymbol{a}_k^n, \boldsymbol{s}_k^n) \\ \text{subject to} \quad & \sum_{n=1}^{N} \theta_k^n \leq \theta_{max}, \end{aligned} \tag{6}$$

where $\theta_k^n \in [0,1]$ is the budget for task n at time k, $c(\cdot)$ is the used cost function and $\theta_{max} \in [0,1]$ is the maximum available budget. A value of $\theta_k^n = 0$ means no assigned budget, while $\theta_k^n = 1$ means that all available budget is assigned.

III. Proposed Solution for SRM Problem

In order to solve the problem defined in section II, an approach involving a POMDP framework and a combination of LR and policy rollout is proposed. This section explains the solution approach in detail.

A. Distribution of Sensor Budgets Using LR

This paper is an extension of our previous research [7], where we used LR to include the constraints into the cost function. By doing so, the original optimization problem is decoupled into smaller ones, one for each task. This leads to the so-called Lagrangian dual (LD) which needs to be optimized. This optimization problem is called the Lagrangian dual problem (LDP) and is formulated as

$$Z_D = \max_{\lambda_k} \left(\min_{\boldsymbol{a}} \left(\sum_{n=1}^{N} (c(\boldsymbol{a}_k^n, \boldsymbol{s}_k^n) + \lambda_k \cdot \theta_k^n) \right) - \lambda_k \cdot \theta_{max} \right), \tag{7}$$

where $\lambda_k \in \mathbb{R}$ is the Lagrange multiplier for the budget constraint. In the beginning ($l = 0$), an initial Langrange multiplier value λ_0 is chosen, for which the LD is minimized with respect to θ_k^n and the resulting budget values are saved. Then, in order to maximize the LD, λ_{l+1} is determined by the use of the subgradient method. The LD is again minimized with this new Lagrange multiplier value. This process is repeated iteratively until the solution converges with the desired precision. Thanks to the sum in the LDP, the minimization problem with resprect to $\boldsymbol{a}$ can be solved for each object n independently. The exact procedure is shown in [7] and can be summarized as follows:

1) $l = 0$: Set initial Lagrange multiplier ($\lambda = \lambda_0$).
2) Solve $Z_D(\lambda)$ and save resulting $\boldsymbol{a}_l^n$ and θ_l^n.
3) Choose subgradient for Lagrange multiplier as $\mu_l^\lambda = \sum_{n=1}^{N} \theta_l^n - \theta_{max}$.
4) Check if $\mu_l^\lambda \approx 0$ with desired precision. If it is, stop the process. The current budget values are the final solution.
5) Set $\lambda_{l+1} = max\{0, \lambda_l + \gamma_l \mu_l^\lambda\}$, where γ_l is the LR step size at time l.
6) Go to step 2 and set $l = l + 1$.

Note that here an internal index l is used for the iterations within the LR process. The final λ_l is the LR solution for λ_k at time step k.

B. Definition of a POMDP

The novelty of this paper is the combination of the above mentioned LR approach with policy rollout for a POMDP framework. A POMDP describes an MDP in which the state cannot be observed directly. Instead, an observation is taken which generates a probability distribution over the possible states. This is called the belief state. Based on the belief states and the knowledge of the underlying MDP, a POMDP allows to solve optimization problems non-myopically, meaning that it takes the expected future into account.

The solution of a POMDP is a set of optimal actions which minimizes the cost, the so-called optimal policy. It is defined as the action $\boldsymbol{a}$ that minimizes the so-called Q-value:

$$\pi_t^*(\boldsymbol{b}_t) = \arg\min_{\boldsymbol{a} \in \boldsymbol{A}} (Q_{H-t}(\boldsymbol{b}_t, \boldsymbol{a})), \tag{8}$$

where H and t are the horizon and time step within this horizon respectively and $\boldsymbol{b}_t \in \mathbb{R}^q$ is the so-called belief state for time step t, with q being the number of elements of the underlying state vector. A more detailed definition of a POMDP is given in appendix A. Many publications consider rewards rather than costs. In that case, the value is maximized and not minimized, as it is done here. Many different solution methods have been introduced in order to solve POMDPs. There are both online, as well as offline approaches.

Most offline methods are based on the so-called Value Iteration (VI), which iteratively calculates the cost/reward values of all possible states. There are exact approaches to VI (e.g. One-Pass algorithm [25]), as well as approximate point-based algorithms (e.g. PBVI or Perseus [26]). The former techniques often lead to very complicated optimization problems, while the latter ones require many points (and therefore memory and computational effort) in order to converge towards the exact solution. The advantage of offline solutions is that the POMDP is solved only once, but the solution is always valid afterwards. Unfortunately, those methods are already infeasible for a small dimensional state space.

Contrary to that, online algorithms only solve a small part of the POMDP that is relevant at the current moment. This makes them less accurate, but much easier and faster to compute. Some of the online approaches involve approximate tree methods (see for example the overview in [17]) or Monte Carlo sampling (e.g. policy rollout).

Since an exact and complete solution of the POMDP is usually infeasible in real scenarios, this paper focuses on the

implementation of policy rollout as an approximate solution. The general structure of our proposed algorithm is illustrated in figure 1. The output of the algorithm are the converged budgets for each task.

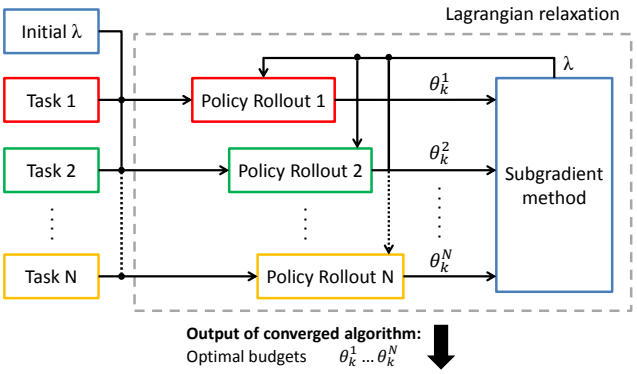

Fig. 1: High level block scheme of the proposed algorithm.

C. Policy Rollout for POMDPs

The basic idea of policy rollout is to take Monte Carlo samples of the expected future. Each sample of one possible future is called a rollout. Within a rollout, observations and belief states are generated for a given initial belief state and a given action. There is at least one rollout per action $\boldsymbol{a}$ in the action space $\boldsymbol{A}$. The action to be evaluated is taken in the first step of the rollout, while a so-called base policy π_{base} is used for every following step, until the horizon H is reached. In each rollout, the total cost is summed up. If several rollouts are run per action, those cost results are averaged. This final cost is the expected value of the evaluated action. Finally, the action that produces the lowest cost is chosen for the next time step. It has been shown that policy rollout leads to a policy that is at least as good as the base policy with a very high probability, if enough samples are provided [23]. The choice of the base policy and the amount of samples to be taken is therefore crucial to the performance of the algorithm.

The policy rollout can be expressed mathematically as shown in (9) and (10). The Q-value is defined as

$$Q^{\pi_{base}}(\boldsymbol{b}_t, \boldsymbol{a}) = C_B(\boldsymbol{b}_t, \boldsymbol{a}) + E\left[V^{\pi_{base}}(\boldsymbol{b}_{t+1})|\boldsymbol{b}_t, \boldsymbol{a}\right], \quad (9)$$

where $E[\cdot]$ is the expectation and $C_B(\boldsymbol{b}_t, \boldsymbol{a}) = \sum_{\boldsymbol{s}\in\boldsymbol{S}} \boldsymbol{b}_t(\boldsymbol{s})c(\boldsymbol{s}, \boldsymbol{a})$ is the expected cost given belief state $\boldsymbol{b}_t$. The best policy can then be found by applying

$$\pi_t(\boldsymbol{b}_t) = \underset{\boldsymbol{a}\in\boldsymbol{A}}{\arg\min}(Q^{\pi_{base}}(\boldsymbol{b}_t, \boldsymbol{a})). \quad (10)$$

This technique does not necessarily lead to the optimal policy, rather to an improvement with respect to the chosen base policy π_{base}.

IV. Two-dimensional Tracking Scenario

In this section, a two-dimensional tracking example is introduced that illustrates the performance of the proposed algorithm. The application is set in a two-dimensional LTI system and the applied tracking algorithm is a simple Kalman filter. In the following subsections, the assumed situation is described explicitly.

A. Assumptions

A constant velocity model is assumed. The distance between the time steps k is defined through the action vector $\boldsymbol{a}_k^n \in \mathbb{R}^2$. It consists of the dwell time τ and T_k^n, which is the revisit interval for object n, as used e.g in radar applications. Therefore (2) can explicitly be written as

$$\boldsymbol{s}_{k+1}^n = \boldsymbol{F}_k^n \cdot \boldsymbol{s}_k^n + \boldsymbol{w}_k^n, \quad (11)$$

with $\boldsymbol{F}_k^n \in \mathbb{R}^{4\times4}$ defined as

$$\boldsymbol{F}_k^n = \begin{bmatrix} 1 & 0 & T_k^n & 0 \\ 0 & 1 & 0 & T_k^n \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (12)$$

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

$$\begin{aligned} \boldsymbol{Q}(T_k^n) &= \begin{bmatrix} \frac{(T_k^n)^2}{2} & 0 \\ 0 & \frac{(T_k^n)^2}{2} \\ T_k^n & 0 \\ 0 & T_k^n \end{bmatrix} \begin{bmatrix} \frac{(T_k^n)^2}{2} & 0 & T_k^n & 0 \\ 0 & \frac{(T_k^n)^2}{2} & 0 & T_k^n \end{bmatrix} \sigma_{w,n}^2 \\ &= \begin{bmatrix} \frac{(T_k^n)^4}{4} & 0 & \frac{(T_k^n)^3}{2} & 0 \\ 0 & \frac{(T_k^n)^4}{4} & 0 & \frac{(T_k^n)^3}{2} \\ \frac{(T_k^n)^3}{2} & 0 & (T_k^n)^2 & 0 \\ 0 & \frac{(T_k^n)^3}{2} & 0 & (T_k^n)^2 \end{bmatrix} \sigma_{w,n}^2, \end{aligned} \quad (13)$$

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

In this example we assume that the sensor produces independent measurements in x and y direction. The measurement equation in (4) for object n at time step k is then

$$\boldsymbol{z}_k^n = \boldsymbol{H} \cdot \boldsymbol{s}_k^n + \boldsymbol{v}_k^n, \quad (14)$$

where $\boldsymbol{H} \in \mathbb{R}^{2\times4}$ is the measurement matrix, defined as

$$\boldsymbol{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \quad (15)$$

and $\boldsymbol{v}_n^k \in \mathbb{R}^2$ is the measurement noise for object n which has independent x and y components. It defined as

$$\boldsymbol{v}_k^n = [v_k^{x,n} \quad v_k^{y,n}]^T \quad (16)$$

with variances $\sigma_{x,n}^2$ and $\sigma_{y,n}^2$. It is assumed that the measurement variances are influenced by the so-called dwell time τ through a factor $1/\tau$. Dwell time is a radar-related term that defines the time the antenna beam spends on an object. Due to independent measurements in x and y direction, the measurement covariance can be defined as

$$\boldsymbol{R}^n = \begin{bmatrix} \sigma_{x,n}^2 & 0 \\ 0 & \sigma_{y,n}^2 \end{bmatrix} \frac{1}{\tau}. \quad (17)$$

The two-dimensional tracking example in this paper is based on the scenario used in our previous work [7]. It is assumed that there are N objects in the environment which are being

tracked by a Kalman filter. The revisit interval T needs to be optimized, while the dwell time τ is constant. Therefore the SRM problem can be expressed as

$$\begin{aligned} \underset{\boldsymbol{T}}{\text{minimize}} \quad & \sum_{n=1}^{N} c(T_k^n, \boldsymbol{s}_k^n) \\ \text{subject to} \quad & \sum_{n=1}^{N} \frac{\tau}{T_k^n} \leq \theta_{max}, \end{aligned} \tag{18}$$

where $\theta_{max} \in [0, 1]$ is the total available budget. The term budget refers to a ratio of dwell time τ and revisit interval T.

Furthermore, it is assumed that every observed object always generates a correctly assigned detection and that there are no false alarms. The measurement variances are constant for the observed area, but can differ from object to object. This means that no influence of SNR is taken into account.

B. Cost Function

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

$$\boldsymbol{P}_{k|k-1}^n(T_{k-1}^n) = \boldsymbol{F}_{k-1}^n \boldsymbol{P}_{k-1|k-1}^n(T_{k-1}^n)(\boldsymbol{F}_{k-1}^n)^T + \boldsymbol{Q}(T_{k-1}^n), \tag{19}$$

where $\boldsymbol{F}_{k-1}^n$ is the transition matrix with interval length T_{k-1}^n as defined in (12), $\boldsymbol{P}_{k-1|k-1} \in \mathbb{R}^{4\times4}$ is the last filtered error-covariance matrix and $\boldsymbol{Q}(T_{k-1}^n)$ is the maneuverability at time step $k-1$ as defined in (13).

Another estimation and prediction cycle is then applied to the error-covariance, in order to end up with the error-covariance $\boldsymbol{P}_{k+1|k} \in \mathbb{R}^{4\times4}$ for time step $k+1$:

$$\boldsymbol{P}_{k+1|k}(T_k^n) = \boldsymbol{F}_k^n \boldsymbol{P}_{k|k,n}(T_k^n)(\boldsymbol{F}_k^n)^T + \boldsymbol{Q}(T_k^n), \tag{20}$$

This expression will be used in the next section to define the cost functions.

V. Simulations

In the following subsections, simulations are presented to highlight some aspects of the proposed algorithm.

A. General Simulation Parameters

The general simulation parameter values are mentioned in table I. If other values are used for the simulations, it is explicitly mentioned in the corresponding subsection. The implemented base policy is simply to apply the evaluated action in every step of the policy rollout. Therefore $\pi_{base,k} = \boldsymbol{a}$. A constant LR step size is applied in all simulations. Within the policy rollout, a Kalman filter is used for generating samples of the expected future.

TABLE I: General simulation parameters.

Paramter	**Value**
Maximum budget θ_{max}:	1
Lagrange step size γ_k:	1000
Initial Lagrange multiplier λ_0:	100
Precision of LR δ_{LR}:	0.001
Fixed value of dwell time τ:	50ms

B. Simulation 1: Comparison OSB and AODB in LTI scenario.

In order to prove the validity of the proposed AODB algorithm, a comparison is conducted with the OSB algorithm as proposed in [7]. In the LTI case, the steady-state error-covariance given the optimal revisit interval T can easily be calculated, for instance by using the equations by Kalata in [20]. The OSB algorithm is used as explained in [7] with the general simulation parameters in table I. It calculates the optimal steady-state budget allocation. The AODB algorithm implements the sum of the predicted error-covariance for the position in x and y direction as cost function. Using (20) this can be expressed as

$$c(T_k^n) = \begin{bmatrix} 1 & 1 & 0 & 0 \end{bmatrix} \boldsymbol{P}_{k+1|k}(T_k^n) \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}. \tag{21}$$

A long rollout of $H = 200$ is chosen in order to show the convergence of the AODB algorithm to values close to the optimal steady-state results from the OSB algorithm. For the comparison, five object tracking tasks are considered with the parameters shown in table II. Since the cost function is only depending on the measurement and maneuverability variances, the position of the objects is omitted here. The dwell time τ is constant while the revisit interval is discretized as $T \in [0.001 : 0.001 : 0.8]$. It is assumed that the budget values are recalculated every 5 seconds. In between, measurements of the objects are taken with the previously calculated revisit intervals T^n.

The simulation results are shown figure 2. It can be seen that the budget allocations $\theta_k^n = \tau / T_k^n$ converge to results that are very close to the values that have been determined with the OSB algorithm. The longer the rollout horizon, the closer the budgets approach those optimal results.

TABLE II: object tracking parameters for simulation 1.

Object	$\boldsymbol{\sigma_{x,n}[m^2]}$	$\boldsymbol{\sigma_{y,n}[m^2]}$	$\boldsymbol{\sigma_{w,n}[m^2]}$
1	25	45	25
2	25	30	250
3	300	55	25
4	150	15	50
5	60	25	30

C. Simulation 2: Influence of Number of Tasks on AODB.

Theoretically, the AODB algorithm should work with a larger number of tasks as well. In this section we present an

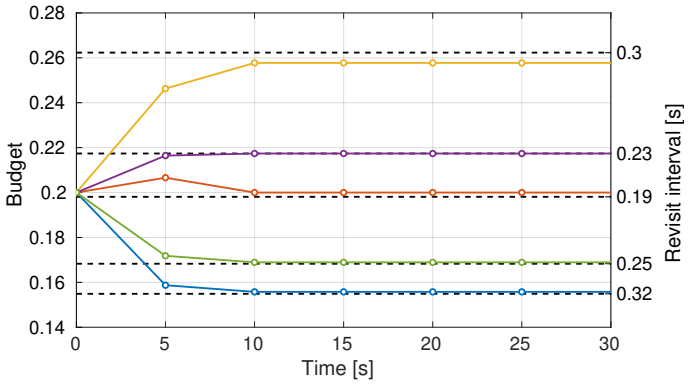

Fig. 2: Budget per task over time after initialization of AODB algorithm. Solid lines: results from AODB. Dashed lines: Optimal steady-state results from OSB.

example with more tasks and show how a larger number of objects leads to an increase in computational complexity.

The simulation of V-B has been repeated with 10 tasks. Figure 3 shows the approximately optimal budget distribution.

A second simulation shows the influence of an increasing number of tasks on the computational load and execution time of AODB. A horizon of $H = 20$ is applied in the policy rollout and the revisit interval is discretized as $T \in [0.01 : 0.01 : 2]$. The precision of the LR is set to 0.01. The implemented cost function in this simulation is the same as shown in (21).

The results of this simulation can be found in figure 4 and 5. It can be seen that the final cost (after convergence of LR) is increasing exponentially with rising number of object tracking tasks. Therefore, a single initial Lagrangian multiplier does not lead to a fast convergence for all numbers of tasks. In addition to that, for a growing number of objects track tasks, the time needed per LR iteration increases linearly. It is important to note that these convergence times only apply during the initialization phase. It should thus be seen as a worst case scenario. When the algorithm uses extra knowledge to choose a better Lagrange multiplier value, these times can be reduced tremendously.

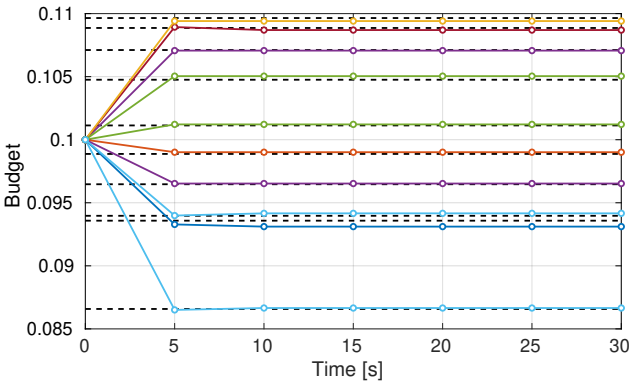

Fig. 3: Budget per task over time after initialization of AODB algorithm. Same simulation as in V-B with 10 tracking tasks.

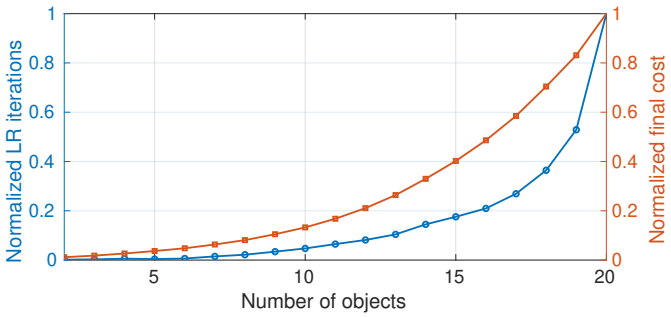

Fig. 4: Number of LR iterations needed for convergence (blue) and final optimal total cost (red) for different number of object tracking tasks. The data is normalized w.r.t. the resulting values for 20 simulated objects (maximum values). It is assumed that the AODB algorithm has just been initialized.

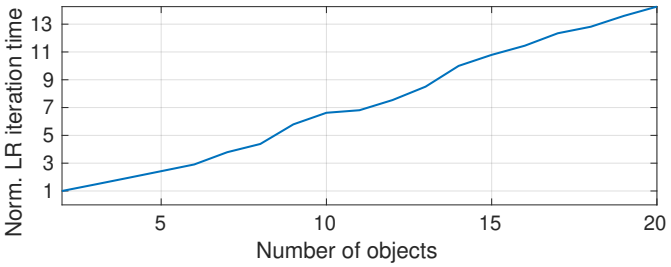

Fig. 5: Average length of an LR iteration in the AODB algorithm depending on the amount of object tracking tasks.

D. Simulation 3: Dynamic Tracking Scenario for AODB.

This simulation incorporates a dynamically changing scenario with changes in budget, number of objects and object tracking parameters. The same object parameters and cost function are used as in section V-B (see table II and (21)). The considered events are the following:

- **0s:** Objects 1 to 4 are being tracked ($\theta_{max} = 1$).
- **20s:** A track for object 5 is added.
- **45s:** Available budget decreases ($\theta_{max} = 0.9$).
- **105s:** Measurement variance in x direction of object 1 increases ($\sigma^2_{x,1} = 210m^2$).

It is assumed that the budget values are recalculated every 5 seconds. In between, measurements of the objects are taken with the previously calculated revisit intervals T^n. The general simulation parameters apply, as mentioned in table I. The final Lagrange multiplier from the previous LR balancing process is reused as initial value. In addition to that, the chosen policy rollout horizon is chosen as $H = 50$. The results of this simulation are shown in figures 6 and 7. It can be seen that the AODB algorithm manages to adapt to various situation changes. For example, the algorithm starts to adapt the budgets already about 20 seconds before the known variance change at $k = 105s$. It is also obvious that the LR algorithm finishes with a very low amount of iterations, if the situation does not change much. In case of bigger changes, a longer convergence

time is needed to converge to the new value.

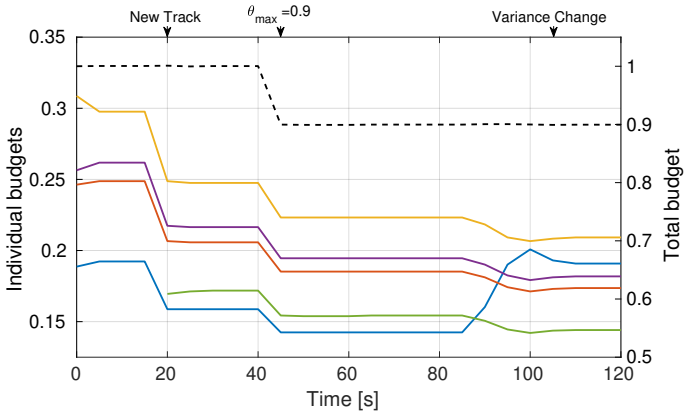

Fig. 6: Dynamic budget allocation with AODB algorithm. The algorithm adapts to three different changes over time. Track colors: blue (1), red (2), yellow (3), purple (4), green (5). The dashed line indicates the total available budget.

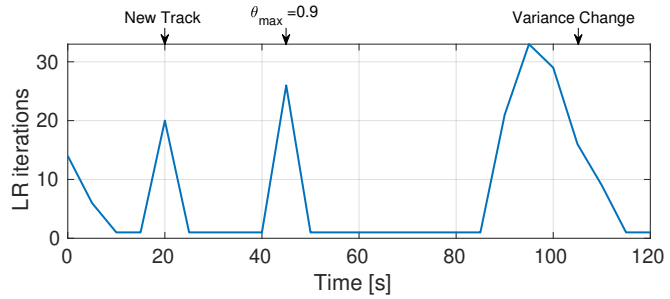

Fig. 7: Amount of LR iterations needed for the AODB algorithm to converge in a dynamic scenario with three situation changes.

VI. Conclusions

In this paper, we have introduced the AODB algorithm for approximately optimally solving a multi-object SRM problem. In [7] we had already introduced the OSB algorithm which is a resource balancing algorithm using LR. Here, it has been extended with a POMDP framework that has been solved non-myopically by using policy rollout. The OSB algorithm is limited to LTI problems, while the the new AODB algorithm can also be applied to dynamically changing scenarios.

The AODB algorithm was applied to a simplified tracking example in order to illustrate its performance. The revisit interval was optimized while using a cost function based on the predicted position error-covariance that can be computed using the Kalman filter.

The first scenario considered an LTI case like in [7]. We have shown that the resulting budget of the AODB algorithm approaches the result of the OSB algorithm already after a few iterations in which the error-covariance matrix converges. The AODB results converge to results that are very close to the optimal steady-state solution. The length of the policy rollout has an impact on how close the AODB algorithm gets to the optimal results.

The second scenario explored the influence of different numbers of object tracking tasks on the AODB algorithm. We illustrated that the optimization can also be done for a larger number of tasks. In addition to that, we showed that for fast convergence the choice of the initial Lagrange multiplier is crucial and depends on the amount of parallel tasks. We concluded that it is useful to incorporate existing knowledge into the choice of initial Lagrange multiplier value. The time to complete a step in the LR process increases with growing number of objects, which makes a fast convergence even more important.

In the third simulation scenario we illustrated the performance of the AODB algorithm in a dynamically changing situation. The final Lagrange multiplier value of the last budget update calculation is reused for the current one. For small situation changes, the amount of LR iterations is very small. Finally, we showed that the algorithm is capable of allocating the current budgets based on changes that are known in advance and changes that are not. It anticipated a known change in measurement variance of a tracking task and adjusted the object budgets early. Based on the chosen cost function, also other changes can be anticipated if they are known in advance or can be predicted.

In future work, we are planning to investigate the usage of the AODB algorithm with more operationally relevant cost functions. In addition to that, we will investigate how to improve and accelerate the LR algorithm. Finally, we will also investigate the usage of the AODB algorithm in a tracking and classification scenario.

Acknowledgments:
This research was partially funded by Nederlandse Organisatie voor Wetenschappelijk Onderzoek (NWO) through the "Integrated Cooperative Automated Vehicles" project (i-CAVE).

Appendix
POMDP definition

A POMDP is defined by the following parameters (see for example [17] and [18]):

State space $\boldsymbol{S}$: Consists of all possible states that can be reached within the process, see (1). At time k the state is defined as $\boldsymbol{s}_k$. Based on the underlying states and the received observations, the belief-state defines a probability distribution over the possible states. At time k it is defined as $\boldsymbol{b}_k$.

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

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

Transition probability $\boldsymbol{\Psi}(\boldsymbol{s}_k, \boldsymbol{s}_{k+1}, \boldsymbol{a}_k)$: The probability function $p(\boldsymbol{s}_{k+1}|\boldsymbol{s}_k, \boldsymbol{a}_k)$ that defines the probability of transitioning from state $\boldsymbol{s}_k$ to state $\boldsymbol{s}_{k+1}$ given action $\boldsymbol{a}_k$.

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

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

Discount factor γ: A discount factor that discounts future time steps with respect to the present.

POMDPs can be solved for finite or infinite horizons. In order to reduce the necessary computational power, a limited horizon H is assumed in this paper. The value of H represents the number of considered time steps with distance T. Every time a new budget allocation is calculated, the horizon H will be reapplied. This is therefore also called a receding horizon.

We would like to find the actions that minimize the total cost (value V_H over horizon H). This can be expressed as

$$V_H = E\left[\sum_{k=k_0}^{k_0+H} c(\boldsymbol{s}_k, \boldsymbol{a}_k)\right]. \tag{22}$$

Using belief states $\boldsymbol{b}$ and $C_B(\boldsymbol{b}_k, \boldsymbol{a}_k) = \sum_{\boldsymbol{s}\in\boldsymbol{S}} \boldsymbol{b}_k(\boldsymbol{s})c(\boldsymbol{s}, \boldsymbol{a}_k)$ being the expected cost given belief state $\boldsymbol{b}_k$, V_H can be written as a so-called value function of the belief state $\boldsymbol{b}_t$:

$$V_H(\boldsymbol{b}_t) = E\left[\sum_{k=k_0+t}^{k_0+t+H} C_B(\boldsymbol{s}_k, \boldsymbol{a}_k)|\boldsymbol{b}_t\right]. \tag{23}$$

For belief state $\boldsymbol{b}_0$ and taking action $\boldsymbol{a}$, the optimal value function is defined according to Bellman's equation [27] as

$$V_H^*(\boldsymbol{b}_0) = \min_{\boldsymbol{a}\in\boldsymbol{A}} \left(C_B(\boldsymbol{b}_0, \boldsymbol{a}) + \gamma \cdot E\left[V_{H-1}^*(\boldsymbol{b}_1)|\boldsymbol{b}_0, \boldsymbol{a}\right]\right). \tag{24}$$

Using this equation, the optimal policy can be expressed as

$$\pi_0^*(\boldsymbol{b}_0) = \arg\min_{\boldsymbol{a}\in\boldsymbol{A}} \left(C_B(\boldsymbol{b}_0, \boldsymbol{a}) + \gamma \cdot E\left[V_{H-1}^*(\boldsymbol{b}_1)|\boldsymbol{b}_0, \boldsymbol{a}\right]\right). \tag{25}$$

The optimal so-called Q-value is then defined as

$$Q_{H-t}(\boldsymbol{b}_t, \boldsymbol{a}) = C_B(\boldsymbol{b}_t, \boldsymbol{a}) + \gamma \cdot E\left[V_{H-t-1}^*(\boldsymbol{b}_{t+1})|\boldsymbol{b}_t, \boldsymbol{a}\right]. \tag{26}$$

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

$$\pi_t^*(\boldsymbol{b}_t) = \arg\min_{\boldsymbol{a}\in\boldsymbol{A}}(Q_{H-t}(\boldsymbol{b}_t, \boldsymbol{a})). \tag{27}$$

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

REFERENCES

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

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

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

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

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

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

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

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

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

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

[11] T. Brehard, P. A. Coquelin, E. Duflos, and P. Vanheeghe, "Optimal Policies Search for Sensor Management : Application to the ESA Radar," in *2008 11th International Conference on Information Fusion*, pp. 1–8, June 2008.

[12] Y. He and E. K. P. Chong, "Sensor scheduling for target tracking: A Monte Carlo sampling approach," *Digital Signal Processing*, vol. 16, no. 5, pp. 533 – 545, 2005. Special Issue on DASP 2005.

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

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

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

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

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

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

[19] F. Hoffmann, H. Schily, A. Charlish, M. Ritchie, and H. Griffiths, "A Rollout Based Path Planner for Emitter Localization," in *22nd International Conference on Information Fusion*, 2019.

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

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

[22] D. P. Bertsekas, J. N. Tsitsiklis, and C. Wu, "Rollout algorithms for combinatorial optimization," *Journal of Heuristics*, vol. 3, pp. 245–262, Dec. 1997.

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

[24] F. Katsilieris, H. Driessen, and A. Yarovoy, "Threat-based sensor management for target tracking," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 51, pp. 2772–2785, Oct 2015.

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

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

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