Policy Rollout Action Selection in Continuous Domains for Sensor Path Planning

Folker Hoffmann[1], Alexander Charlish[1], Matthew Ritchie[2], Hugh Griffiths[2]
[1]Fraunhofer FKIE, [2]University College London
{folker.hoffmann,alexander.charlish}@fkie.fraunhofer.de
{m.ritchie,h.griffiths}@ucl.ac.uk

***Abstract*—Policy rollout is a method for the online computation of future costs in approximate dynamic programming, and has been utilized for various problems including sensor management. In previous work, it has predominately been applied to the selection of actions from discrete sets. In this paper we present methods for action selection from continuous sets and analyze their trade-offs. The methods are evaluated on the problem of sensor path planning, with the intent of minimizing the time to localize an emitter using bearing measurements.**

***Index Terms*—policy rollout, stochastic optimization, approximate dynamic programming, sensor management, sensor path planning, partially observable Markov decision process (POMDP), Markov decision process (MDP), emitter localization**

I. Introduction

Sensor management is the problem of controlling reconfigurable sensors for optimized performance. An example for sensor management is the problem of sensor path planning for emitter localization. Using a direction finding sensor, the localization performance is dependent on the sensor-to-target geometry. The sensor path planning problem is then to optimize the path of the sensor, based on the received measurements. Because the path planning algorithm does not yet know the position of the emitter, this can be seen as a problem of decision under uncertainty. Almost all sensor management problems deal with decisions under uncertainty, due to missing knowledge before the sensing process.

A framework to model decisions under uncertainty is stochastic dynamic programming. Problems can be modeled as Markov decision processes (MDPs) if the state is observable and only its transitions are stochastic. If the state is not fully observable, the problem can be modeled as a partially observable Markov decision process (POMDP). Due to this property, POMDPs are a topic of wide interest in the sensor management literature. Exact solutions for MDPs and POMDPs are only viable in small dimensional problems. Therefore, commonly approximate solution techniques are used.

One method to approximately solve MDPs and POMDPs is the policy rollout method [1]. This method examines the different actions available to the controller and simulates the future outcomes of choosing this action. The simulations are performed using realizations of random variables and a *base policy*, which is an already existing, often heuristic, policy for the system. The performance of this algorithm is typically better than the performance of the base policy. In previous works, this algorithm has been predominantly applied to discrete action sets, where the algorithm could exhaustively evaluate the set. In this paper, we analyze methods to extend the policy rollout algorithm to continuous sets.

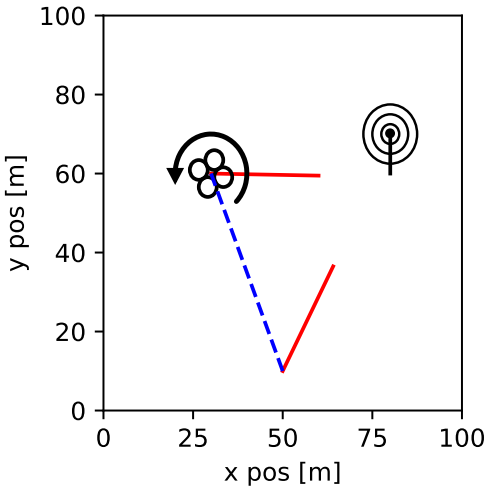


Fig. 1. A UAV with a directional antenna rotates to create a bearing (red) towards an emitter. Between measurements, it moves towards a new measurement position (blue).

As an application, we evaluate these methods on a sensor management problem, which falls into the domain of sensor path planning. We discuss a system where a directional antenna is mounted on an unmanned aerial vehicle (UAV) and bearing measurements can be derived by rotating the UAV around its vertical axis [2]–[5]. Existing solutions for this sensor management problem [2], [5] do not take into account the probabilistic outcome of future measurements. This uncertainty can easily be integrated into the policy rollout algorithm. A schematic visualization of this problem can be seen in Figure 1. An important aspect of the considered sensor management problem is that measurement generation and platform movement are mutually exclusive. Therefore, there exists a trade-off in using time for generating measurements and in moving to another location. The goal of the system is to localize a stationary, ground-based emitter as fast as possible.

A. Policy rollout

A POMDP formalizes the problem of decision under uncertainty: At time step k, the state of the system is denoted as $x_k \in \mathcal{X}$, where $\mathcal{X}$ is called the state space. A controller can act on this state via actions a_k, which are part of the action space $\mathcal{A}$. An action a_k leads to a transition between the states, which is described via a transition function $x_{k+1} = f(x_k, a_k)$. Knowledge over the state is only known by a measurement function $z_k = h(x_k, v_k)$, where z_k is a measurement belonging to the measurement space $\mathcal{Z}$, and v_k is the realization of a random variable representing the noise of the sensor measurements. Additionally, a real-valued cost function $c(x_k, a_k)$ is defined, which quantifies the costs of an action in a state. Commonly the measurements are summarized into a belief $b_k = p(x_k)$, which is a probability distribution over the state space. If the belief is a sufficient statistic of the state, i.e. summarizes all available information, this problem can be considered as a Markov decision process on the space of possible beliefs, which is called the belief space $\mathcal{B}$.

Given a POMDP formulation, a policy $\pi : \mathcal{B} \to \mathcal{A}$ is a mapping from belief state to the action space. The problem is then to find the optimal policy π^*, which minimizes the expected cost:

$$\pi^* = \underset{\pi}{\operatorname{argmin}}\ V^{\pi}(b_0)\ . \tag{1}$$

The function $V^{\pi} : \mathcal{B} \to \mathbb{R}$ is called the value function of π, and the value $V^{\pi}(b_k)$ is the expected total cost of following policy π in belief state b_k:

$$V^{\pi}(b_k) = \mathbb{E}\left[\sum_{i \geq k}^{\mathcal{T}(b_i)} c(x_i, \pi(b_i))\right]\ . \tag{2}$$

This sum is to be interpreted as summing up all $i \geq k$ until the termination criterion $\mathcal{T}(b_i)$ is true. We introduce this notation, as the total number of future steps varies for different realizations of the unknown true states x_i and future measurements z_i.

Given the value function V^* of the optimal policy π^*, an optimal action selection would be to select the action that minimizes the immediate cost and the future cost:

$$\pi^*(b_k) = \underset{a \in \mathcal{A}}{\operatorname{argmin}}\ \mathbb{E}\left[c(x_k, a) + V^*(b_{k+1})\right]\ . \tag{3}$$

The expected value represents the total expected cost of the action, and is therefore called the action value:

$$Q^*(b_k, a) = \mathbb{E}\left[c(x_k, a) + V^*(b_{k+1})\right]\ . \tag{4}$$

This framework for sequential decision processes is commonly used in the literature on dynamic programming [6], [7].

The policy rollout algorithm is an online computation method, which approximates the future cost. Its idea is to replace the optimal value function V^* by the cost function V^{π_B} of a base policy π_B.

$$\pi^R(b_k) = \underset{a \in \mathcal{A}}{\operatorname{argmin}}\ \mathbb{E}\left[c(x_k, a) + V^{\pi_B}(b_{k+1})\right] \tag{5}$$

$$= \underset{a \in \mathcal{A}}{\operatorname{argmin}}\ Q^{\pi_B}(b_k, a)\ . \tag{6}$$

Here Q^{π_B} represents the action value when following the base policy. Computation of this value is also called evaluation of action a. The policy rollout algorithm can be considered as a single step of policy iteration [6, p. 110]. The resulting policy is at least equal and typically better than the base policy. This property is called *rollout improvement property*.

In the present literature on policy rollout, the search for the optimal action, i.e. the argmin in (5), is commonly performed by evaluation of each action. For each action the expected value in (5) is computed, e.g. via a fixed number of Monte Carlo samples. This not only requires a discretization in a continuous action space, but also wastes computation time on evaluating suboptimal actions. In this paper, we discuss the problem of searching for the minimum in (5), which consists of two subproblems: Selecting which actions to evaluate and evaluating the expected value. This allows us to apply the rollout procedure in a continuous action space. Some of the methods also improve the action search in the classical setting with discrete actions.

Since it is a generic method, the policy rollout algorithm has been applied to a variety of problems. Early works consider the problem of playing Backgammon by Tesauro [8], combinatorial optimization by Bertsekas et al. [9] and stochastic scheduling by Bertsekas and Castañon [1].

It has been used in several works in sensor management [10]–[23], for example for the activation of nodes in a sensor network [12]–[15] and sensor to target association [17], [18]. Other problem settings encompass vehicle routing [24]–[28], inventory routing [29], revenue management [30] and scheduling [31].

B. Computation of the action value

A main component of the rollout algorithm is the evaluation of the expected value in (5). In some problem domains an analytical solution can be performed [25]–[29], typically via dynamic programming on the discrete state space. However, in sensor management, an exact evaluation is only possible in very specific cases [22], as the state space is commonly continuous.

Often the expected value is computed via Monte Carlo sampling, which is almost exclusively the case in sensor management applications. One problem with Monte Carlo methods to estimate an expected value is the variance of the estimate, which is why techniques for variance reduction exist [32]. One technique to reduce the variance of Monte Carlo estimates is the use of common random numbers (CRN), in which the evaluations of different actions use the same realizations of random variables. In the rollout literature, this is also known as sampling of the Q-factor differences [33]. Common random numbers have been used in several works on vehicle routing [27], [28], however, have not been explicitly reported in the sensor management literature in the context of policy rollout. The works in [13]–[16] can be considered as using a form of CRN, as they initialize their Monte Carlo rollouts using the same particles from a particle filter, which leads to the same initial underlying state in the rollouts for different actions.

A different method is the evaluation of the expected value via a set of representative fixed values for the random variables, also called scenarios [1]. One example for this approach is the work reported in [17]. Here samples of the uncertainty over the state space and the future measurements are deterministically created using a method similar to the unscented transform. The expected value is then computed using those samples. However, this only works for Gaussian uncertainties. In [34] a generic method to suboptimally discretize arbitrary probability densities [35] was used to create a set of representative samples. As a special case of deterministic samples, the expected values of the random variables can be taken [23].

In this work, we compare the effectiveness of random samples, common random numbers and the usage of deterministic samples. While random samples are often used, common random numbers are non-standard in sensor management, and the evaluation with deterministic samples is only used in a small number of works.

C. Search for the optimal action

Next to the computation of the expected value, another main component in evaluating (5) is the procedure to compute the `argmin`, i.e. to search for the action with minimal cost. In the vast majority of the literature, this was performed by evaluation of every action in a finite action set. When the action values were determined via Monte Carlo sampling, commonly an equal number of samples was used for each action.

If the action space contains a large number of discrete actions, evaluating each might not be feasible. An option to speed up the search is then to prune this space prior to the evaluation, which has been performed in [18], [19].

When each action can be evaluated and sampling is used to estimate the action values, the search can be improved by non-uniformly allocating the samples between the different possible actions. The idea is that it is more critical to estimate the value of actions that are candidates of being the optimal action, instead of improving the estimate of clearly inferior actions. As this uses the results of previous action evaluations to improve the allocation of future samples, it can be described as *adaptive action evaluation*. Tesauro proposed to stop evaluation of an action once it becomes unlikely that it is the optimal one [8]. Optimization of the sample allocation has been performed by [36], which uses the optimal computing budget allocation (OCBA) [37] algorithm. OCBA computes the sample mean and standard deviation of encountered action rewards. Then it uses those statistics to determine how often each action should be evaluated.

A method related to the online policy rollout method is classification-based policy iteration, which uses policy rollouts offline during training of a policy [38]. Here a multiple multi-armed bandit method has been used to allocate the resources between multiple decision problems of the type of (5).

Those approaches require a discrete action space and allocate a number of samples to each action. An alternative is to perform the action search directly on the continuous action space. This has been performed for Monte Carlo tree search (MCTS), which is a method related to policy rollout. MCTS uses rollouts to steer the exploration in a search tree. For this algorithm bandit algorithms for continuous action sets [39] have been used to perform continuous action selection with a discrete state space [40].

Except from the MCTS approach above, to the authors' knowledge a continuous action set has not been considered before in the context of policy rollout. In this work, we propose and evaluate several methods to search for the optimal action in a continuous action space, both by using a discretization, as well as a direct search on the continuous action space.

D. Structure of the Paper

The remainder of the paper is structured as follows. In Section II we formally describe the problem we want to solve. In Section III we describe the main components of our policy rollout based control algorithm, excluding the action evaluation and action selection step. Those steps are described in Sections IV and V. In Section VI we describe how we evaluate the different optimization methods, and present the results in Section VII. Section VIII evaluates how an improved action selection correlates with improved performance of the actual rollout algorithm. The results are discussed in Section IX. A discussion of the robustness of this approach and possible generalizations is done in Section X. Finally we conclude the paper in Section XI.

II. Problem Description

A. State space and transition

The state at the time step k is a 4-dimensional vector, consisting of the stationary target position $\mathbf{x}^t = (x^t, y^t)$, and the current position of the platform $\mathbf{x}_k^p = (x_k^p, y_k^p)$:

$$\mathbf{x}_k = (\mathbf{x}^t, \mathbf{x}_k^p)^T = (x^t, y^t, x_k^p, y_k^p)^T \in \mathbb{R}^4 \ . \tag{7}$$

At each time step k the measurement is generated based on the true bearing and additive Gaussian distributed measurement noise:

$$z_k = \text{atan2}(y^t - y_k^p, x^t - x_k^p) + \mathcal{N}(0, \sigma^2) \ , \tag{8}$$

with known standard deviation σ. Taking this measurement takes an amount of time, denoted t_M.

After each measurement a control algorithm chooses an action $\mathbf{a}_k \in \mathbb{R}^2$ to move the platform to a position, where the next measurement is performed. This leads to a distance cost in time of

$$t_D(\mathbf{x}_k^p, \mathbf{a}_k) = \frac{\|\mathbf{x}_k^p - \mathbf{a}_k\|_2}{v_p} \ , \tag{9}$$

where v_p is the speed of the platform. We assume here that the effect of acceleration is negligible and that the platform takes the direct path, however, this is not necessarily required. One could model the distance cost as being different in different directions to account for the influence of wind. It would also be possible to take into account some obstacle avoidance mechanism. However, to exclude those obstacles, this would necessitate a change of the action space.

Given the preceding definitions, the transition function of the state is the following:

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{a}_k) = (\mathbf{x}^t, \mathbf{a}_k)^T, \tag{10}$$

with an associated cost in time of

$$c(\mathbf{x}_k^p, \mathbf{a}_k) = t_D(\mathbf{x}_k^p, \mathbf{a}_k) + t_M \ . \tag{11}$$

This cost is received each time the platform moves and takes a measurement, thereby incentivizing it to localize the target as fast as possible.

B. Belief state

The state is not directly available to the control algorithm, but only indirectly observed via the received measurements. The measurements can be integrated into a belief, i.e. a probability distribution of the target position

$$b_k^t(\mathbf{x}^t) = p(\mathbf{x}^t \mid b_0^t, \mathbf{a}_0, z_1, \mathbf{a}_1, ..., \mathbf{a}_{k-1}, z_k) \ . \tag{12}$$

b_0^t denotes any prior knowledge about the target position or assumptions like a uniform prior distribution. The full knowledge of the state is the belief at time k, which includes the fully observable platform position:

$$b_k = (b_k^t, \mathbf{x}_k^p) \ . \tag{13}$$

C. Optimization objective

The objective of the control algorithm is to minimize the time until the target localization is sufficiently accurate. This requirement on the localization accuracy is formalized by the termination criterion $\mathcal{T}$:

$$\mathcal{T}(b_k) = \mathbb{1}_{\mu(b_k) \leq \mu_T}(b_k) \ . \tag{14}$$

Here $\mu(b_k)$ is the expected root-mean-squared error (RMSE) of the current estimate, μ_T is a threshold on the localization error, and $\mathbb{1}$ the indicator function.

Then the objective is to find a policy, which minimizes the expected cost in time

$$\underset{\pi}{\operatorname{argmin}} \ \mathbb{E}\left[\sum_{k \geq 0}^{\mathcal{T}(b_k)} c(\mathbf{x}_k, \pi(b_k))\right] \ , \tag{15}$$

where the sum goes over all $k \geq 0$, until $\mathcal{T}(b_k)$ is true. The expectation goes over all future measurements $z_{k \geq 1}$, as well as the unknown, but stationary, target position $\mathbf{x}^t$.

III. Stochastic Control Algorithm

The control algorithm is based on the policy rollout framework described in the introduction. Its basic components are identical to the algorithm in [34]. In this section, we describe in detail the used base policy, the localizer, as well as the way the estimates of the Q-values are computed.

A. Localizer

In a similar way to the localization methods in [41] and [2], we use a discrete grid-based Bayes filter to represent the probability distribution over the target position. Each grid cell represents the probability density of the target being at the corresponding position. This is more computationally demanding than e.g. a representation in form of a Gaussian distribution, but allows us to capture the non-linearity in the estimation process.

We denote the extension of the grid by $\underline{x}^B, \overline{x}^B, \underline{y}^B, \overline{y}^B$, which represent the minimum and maximum coordinate of interest in the x and y axes. The grid is discretized into 100 cells on the longer axis, and a proportionally smaller integer number on the other axis, which is chosen to make the grid cells closest to a square. If a prior distribution of the target's position is available (as in the scenario used later), the grid is initialized using this prior.

With successive measurements, the area of potential target positions shrinks as parts of the region become more and more unlikely. To focus the computation of the posterior on the regions of interest, we assume that the target is not outside $\pm 4\sigma$ of each measurement. Based on all received measurements we compute a convex hull, which encompasses the points that are inside $\pm 4\sigma$ of all encountered measurements. The belief grid is then resized to the minimal rectangle containing this convex hull. If due to this step the grid resolution in the larger dimension becomes smaller than 40 cells, the resolution in both dimensions is doubled. As the paper is focused on the control algorithm instead of localization methods, we did not implement an outlier detection step in the localization and instead discarded and resampled outliers $> 3\sigma$.

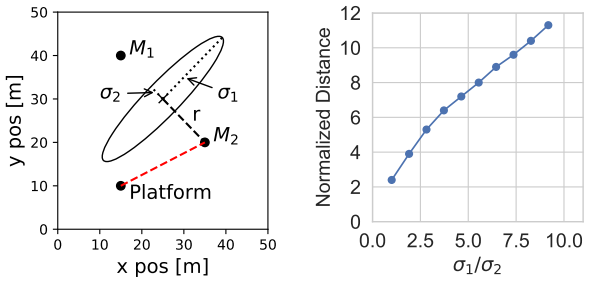
A point estimate $\tilde{\mathbf{x}}_k$ of the target position is derived from the center of the cell with maximum likelihood. Based on this point estimate and the grid estimate of the posterior, we computed the expected RMSE $\mu(b_k)$. Additionally, the covariance $\tilde{\mathbf{P}}_k$ is computed, which approximates the density of the grid.

B. Base policy

The base policy selects greedily the next measurement position with approximately the lowest expected root-mean-squared error. For this, it follows the intuition that, if the belief about the target position were Gaussian, the next measurement should be perpendicular to the major axis of the corresponding uncertainty ellipse.

Figure 2a visualizes this idea. Here the Gaussian approximation $(\tilde{\mathbf{x}}_k, \tilde{\mathbf{P}}_k)$ of the belief is shown as ellipse. Assuming one would only take a single measurement without considering movement costs, it should be perpendicular to the major axis. Then the next measurement position is determined by the distance r to the target position estimate $\tilde{\mathbf{x}}_k$. There are two possible measurement positions with distance r, which are denoted by M_1 and M_2 in the figure.

The optimal distance r between the measurement position and the target position estimate is dependent on the ratio of its axes. The greater the major axis is in relation to the minor axis, the further away a measurement should be taken to increase



(a) (b)

Fig. 2. Base policy for the policy rollout. (a) The next measurement point is chosen perpendicular to the major axis of the uncertainty ellipse. (b) The distance r is proportional to the ratio of the standard deviations. Note that $\sigma_1/\sigma_2 \geq 1$ by definition.

the chance of an approximate cross-bearing measurement. On the other hand, the measurement contains less information about the target position when taken from further away, as the likelihood of the measurement spreads over a larger region.

We computed the optimal distance r offline by setting the minor axis of a Gaussian shaped belief to a standard deviation of $\sigma_2 = 1$ and varying the major axis. For a given length of the major axis, we computed the expected posterior RMSE for each distance r, and selected the distance with the lowest RMSE. The expected posterior RMSE was computed by computing the posterior distribution for all possible measurements based on the grid cells and the measurement noise. For efficiency reasons the possible measurements were binned into 1° bins. Figure 2b shows the resulting optimal distance r, dependent on the standard deviation of the major axis. Values higher than precomputed are linearly interpolated from the last values.

At runtime the belief b_k of the target position is represented as a density on the belief grid. This density is approximated by a Gaussian distribution, centered on the estimated position $\tilde{\mathbf{x}}_k$. Based on the covariance $\tilde{\mathbf{P}}_k$ of the Gaussian approximation, the base policy computes the range r between the next measurement position and the estimated target position. For this, the standard deviations of the principal components are computed. Then, based on their ratio, a piecewise linear interpolation of the precomputed values is performed:

$$r = \sigma_2 \cdot \mathrm{PLI}\left[\frac{\sigma_1}{\sigma_2}\right] \ . \tag{16}$$

As can be seen in Figure 2a there are two possible candidate measurement positions M_1 and M_2 with range r. The base policy selects the one that is nearer to the platform, which is M_2 in the figure. Then it moves to this position (red line in the figure) and generates the next measurement there.

C. Rollout

Given the base policy, the policy rollout approximation of the action value is the sum of the myopic cost $c(x_k^p, a_k)$ and the expected value of $V^{\pi_B}(b_{k+1})$. We approximate this expectation by drawing samples of the state and the measurements during the rollout. The rollout is executed until the termination criterion $\mathcal{T}$ is true, which indicates that the target is sufficiently localized. Then the action value is

$$Q^{\pi_B}(b_k, \mathbf{a}) = c(\mathbf{x}_k^p, \mathbf{a}) + \sum_{j=0}^{N_a-1} w_{aj} \cdot \sum_{i>k}^{\mathcal{T}(b_{aji})} c(\mathbf{x}_{aji}, \pi_B(b_{aji})) \ , \tag{17}$$

where N_a is the number of rollouts performed for action $\mathbf{a}$ and w_{aj} is the weight of the j-th rollout. The rollout state

$$\mathbf{x}_{aji} = (\hat{\mathbf{x}}_{aj}^t, \mathbf{x}_{aji}^p)^T = (\hat{x}_{aj}^t, \hat{y}_{aj}^t, x_{aji}^p, y_{aji}^p)^T \tag{18}$$

consists of the sampled target position and the predicted platform position. $\hat{x}_{aj}^t, \hat{y}_{aj}^t$ is the target position sample for rollout j, evaluating action $\mathbf{a}$. We use the hat to denote that an element is sampled, and specify the sampling method in Section IV. The predicted platform position x_{aji}^p, y_{aji}^p varies during the rollout and is therefore also indexed by the future time step i. The belief b_{aji} is updated on measurements drawn according to

$$z_{aji} = \mathrm{atan2}(\hat{y}_{aj}^t - y_{aji}^p, \hat{x}_{aj}^t - x_{aji}^p) + \hat{v}_{aji} \ , \tag{19}$$

where $\hat{v}_{aji}$ represents the realization of the normal distributed noise in (8). With

$$C_j(b_k, \mathbf{a}) = \sum_{i>k}^{\mathcal{T}(b_{aji})} c(\mathbf{x}_{aji}, \pi_B(b_{aji})) \tag{20}$$

we denote the result of the j-th rollout for action $\mathbf{a}$ and b_k and its computation is called the j-th rollout. This is a single sample of the expected future cost.

D. Action search

The remaining steps of this algorithm are: First, the selection of the samples to evaluate the rollout result of an action (20). Second, to define how the search for the minimum proceeds, i.e. which actions $\mathbf{a}$ are to be evaluated, and with how many rollouts N_a. Methods for this are described in the following two sections.

An overview of the different combinations can be found in Table I. The three columns denote three sampling methods to evaluate the expected value: Plain Monte Carlo, common random numbers, and deterministic samples. Those methods are explained in the next section. The ways the search proceeds are listed in the rows. Those methods are explained in Section V. A combination of search method and sampling gives a specific implementation of the rollout. Combinations which are principally possible are denoted with a $+$. Those which are evaluated in this paper are denoted with a $\bigoplus$.

IV. Computation of the Expected Value

We use a sampling based method to compute the action value $Q^{\pi_B}(b, \mathbf{a})$. This is performed by executing multiple rollouts, where the result C_j of each rollout is determined by the chosen samples, i.e. the j-th *sample path*, which consists of $\hat{\mathbf{x}}_{aj}^t, \hat{v}_{aj(k+1)}, \hat{v}_{aj(k+2)}, \ldots$ For the creation of the sample paths, we compare three different approaches, plain Monte Carlo, common random numbers, and a deterministic sampling

TABLE I
COMBINATIONS OF THE CONSIDERED METHODS

	PMC	CRN	Det.
Uniform allocation	$\oplus$	$\oplus$	$\oplus$
Sequential halving	$\oplus$	$\oplus$	
Quadrant search	$+$	$\oplus$	$\oplus$
Stochastic gradient descent	$+$	$\oplus$	
BFGS		$+$	$\oplus$
Possible combination	$+$		
Evaluated combination	$\oplus$		

approach. The weights for each rollout result are identically assigned $w_{aj} = 1/N_a$ to average the rollout results, except in some cases of the deterministic sampling method.

A. Plain Monte Carlo

The plain Monte Carlo (PMC) approach is to create each sample path by using independent random numbers, i.e. $\hat{\mathbf{x}}^t_{aj}$ and $\hat{v}_{aji}$ are drawn independently for each action a, rollout j and future step i.

B. Common random numbers

Common random numbers (CRN) [32, p. 120] can be used to compare alternatives. With this method, the same random number sequences are used for each action, i.e. different actions are evaluated using the same sample paths. This induces a correlation between the rollouts, which reduces the variance of the relative error in the action value estimates. Note that the absolute error is not reduced, however, only the relative error is important to make the right decision. Here we implemented CRN, such that

$$\hat{\mathbf{x}}^t_{aj} = \hat{\mathbf{x}}^t_{a'j} \tag{21}$$

for the same rollout number j and different actions a and a'. Note that $\hat{\mathbf{x}}^t_{aj}$ is still randomly sampled. Similarly, we chose the measurement noise to be identical for different actions, but the same rollout number and future time step

$$\hat{v}_{aji} = \hat{v}_{a'ji} \ . \tag{22}$$

We refer to both CRN and PMC as Monte Carlo approaches.

C. Deterministic samples

Additionally, we evaluate a deterministic sampling approach, previously used in [34]. Here the uncertainty of the state is represented using a deterministic set of samples $\hat{\mathbf{x}}^t_{aj}$, which are computed with a density approximation algorithm [35] from the belief grid. For each rollout j a different sample is used, therefore N_a samples are required. These samples are used for each action, therefore (21) also describes this approach. As with CRN this introduces a correlation between the rollouts for different actions, but with the same rollout number j.

The method works by splitting prior samples of the density into two new samples each, both containing half of their parent's weight. Therefore, the weights of the samples are only uniformly $w_{aj} = 1/N_a$ when N_a is a power of two. In the evaluation below, we only use powers of two.

The measurements are approximated without measurement noise, i.e.

$$\hat{v}_{aji} = 0 \ . \tag{23}$$

Also discretizing the measurements would impose a greater computational demand, as measurement errors are assumed to be $i.i.d.$ and therefore the number of required discretizations would increase exponentially with the number of measurement steps. We therefore assume that the main source of uncertainty lies in the position of the target.

V. SEARCH FOR THE OPTIMAL ACTION

In this section methods are described to search for the optimal action $\mathbf{a}^* \in \mathcal{A}$, which minimizes (17):

$$\mathbf{a}^* = \operatorname*{argmin}_{a \in \mathcal{A}} Q^{\pi_B}(b_k, \mathbf{a}) \ . \tag{24}$$

For a given action $\mathbf{a}$, $Q^{\pi_B}(b_k, \mathbf{a})$ can be evaluated by performing a rollout based on a sample path as described above. However, it is not necessarily required that each $\mathbf{a}$ is evaluated using the same number of rollouts. In addition, with a continuous action space it is not possible to evaluate each action.

To compare the methods, it is useful to consider their performance based on the total amount of rollouts they require. We denote the total number of rollouts as *computational budget* N, with

$$N = \sum_{\mathbf{a} \in \mathcal{A}^E} N_a \ , \tag{25}$$

where $\mathcal{A}^E \subseteq \mathcal{A}$ is the finite set of evaluated actions.

None of the methods described below are novel, however, with the exception of the uniform allocation method their use in the policy rollout algorithm is novel.

A. Uniform allocation

The most straightforward method to search for the optimal action is to estimate each action value using the same number of samples and select the one whose estimate is best. In a continuous action space, this requires a discretization of the action set. We denote the discretized and finite action set by $\mathbb{A}$. Then the selected action is chosen by comparing the estimated value for each candidate action:

$$\pi^R(b) = \operatorname*{argmin}_{\mathbf{a} \in \mathbb{A}} Q^{\pi_B}(b, \mathbf{a}) \tag{26}$$

We use the same action discretization approach as previously in [34]. The assumption is that the optimal action is close to the target estimate, and therefore the considered region is centered on the belief grid. Additionally, we add the width and height of the belief grid to each border and crop by the scenario area $\underline{x}^S, \overline{x}^S, \underline{y}^S, \overline{y}^S$, resulting in the action space

$$\begin{aligned}
\underline{x}^A &= \max\left(\underline{x}^S, \underline{x}^B - (\overline{x}^B - \underline{x}^B)\right) \ , \\
\overline{x}^A &= \min\left(\overline{x}^S, \overline{x}^B + (\overline{x}^B - \underline{x}^B)\right) \ , \\
\underline{y}^A &= \max\left(\underline{y}^S, \underline{y}^B - (\overline{y}^B - \underline{y}^B)\right) \ , \\
\overline{y}^A &= \min\left(\overline{y}^S, \overline{y}^B + (\overline{y}^B - \underline{y}^B)\right) \ .
\end{aligned} \tag{27}$$

As described in Section III-A, $\underline{x}^B, \overline{x}^B, \underline{y}^B, \overline{y}^B$ denote the extension of the grid representation of the belief. This action space is then discretized equally in both dimensions, leading to an action grid.

The computation of the minimum is then performed by evaluating each action, either by plain Monte Carlo, common random numbers or using the deterministic samples.

B. Multi-armed bandits

Bandit algorithms follow the intuition of a gambler playing at a slot machine with multiple arms, where each arm has a different reward distribution. At each round, the gambler can pull one of the arms, which leads to a sample from the reward distribution. In its most common formulation, the goal is to maximize the cumulative reward over time, leading to a trade-off in using the arm with the currently highest reward expectation (exploitation) or trying other arms (exploration). A typical algorithm for this problem is the UCB algorithm [42].

In recent years another problem formulation has gained interest, the so called *best-arm-selection* problem or *pure exploration* [43]–[47]. In this case, the exploration stops after a certain number of rounds and the gambler commits to a single arm. This single arm is the only one of interest; any rewards previously obtained do not count. Therefore, there is no exploration-exploitation trade-off, but instead the goal is to explore the arms optimally to select the best one.

There is a clear connection between the best-arm-selection problem and the problem of searching for the optimal action in a sampling-based policy rollout algorithm. In both cases, there are multiple candidate actions, and the value of each action is only available by repeatedly performing a rollout or repeatedly pulling an arm. Each additional rollout performed gives a better estimate of the value of an action. Different to the classical bandit formulation in the best-arm-selection formulation, as well as in the policy rollout case, it is not important how much cost or reward is gained during the evaluation of the actions. Only the cost of the selected action is important.

Compared to the uniform allocation described in the previous section, with these methods different actions $\mathbf{a}_1, \mathbf{a}_2$ might get evaluated with a different number of samples $N_{a_1} \neq N_{a_2}$. Similar to the uniform allocation, a finite action set is required, for which we used the same grid discretization $\mathbb{A}$.

We implemented the *successive halving* method [47], which focuses its evaluations over time on the most promising actions. The basic idea is that the whole computational budget N is split into

$$N_r = \lceil \log_2 |\mathbb{A}| \rceil \tag{28}$$

rounds. For each round the same amount of samples N_s is used for each action:

$$N_s = \left\lfloor \frac{N}{|\mathbb{A}| N_r} \right\rfloor . \tag{29}$$

Note that due to the rounding to integers, in some cases the budget is not exhaustively used. After each round half of the actions, whose mean reward over all previous evaluations is worst, are removed and the computational budget is focused on the remaining actions. We use an implementation with plain Monte Carlo sampling, as well as one with common random numbers to compare the actions in each round. There is no straightforward use of the deterministic sampling approach with sequential halving. This is because the approach we used splits and therefore modifies the existing and evaluated samples, instead of simply adding new ones. In an adaptive approach, this means that samples, which are already used for the estimation of the action value, need to be thrown away later together with the corresponding rollout results, and replaced by two other samples. Therefore, the deterministic sampling approach would waste a lot of the computational budget and would therefore not be well suited for adaptive action evaluation.

C. Quadrant search

Quadrant search is the method of restricting the search iteratively into the most promising quadrant [48]. At the start of the algorithm, actions are evaluated on a 3×3 action grid. Then for each quadrant, the mean of the action values at the four corners is computed. Based on these values, the search focuses on the quadrant with the lowest mean action value. In this quadrant, additional five samples are evaluated, in the center and the middle of each border. Therefore, the quadrant now contains a smaller 3×3 grid. Then a quadrant from this smaller 3×3 grid is chosen. This step is repeated for a fixed number of iterations or until convergence.

Figure 3 visualizes this progress. In the first step, actions are evaluated at the positions A-I via rollouts. Then the quadrant averages were compared and it was determined that

$$\frac{1}{4} \sum_{\mathbf{a} \in \{\text{B,C,E,F}\}} Q^{\pi_B}(b_k, \mathbf{a}) \tag{30}$$

was less than the corresponding averages for the other quadrant. Therefore, in the second step the action values at J-N were evaluated. The next step would select a quadrant of (B,C,E,F), for example (L,M,N,F), and repeat the subdivision.

We implemented this method in a variant with common random numbers, as well as a variant based on deterministic samples. The initial area on which the 3×3 grid is placed is based on the same limits as the grid of the uniform allocation and sequential halving approaches (27).

D. Gradient-based methods

Gradient descent is a classic method for optimization problems. If a function is continuous, following the gradient leads to at least a local minimum. Figure 5 shows the true Q-values, computed using a sufficient number of samples (see Section VI), which appear sufficiently continuous under a visual inspection, and typically have two local minima.

If the gradient computation is not deterministic (e.g. because it is computed based on a random sample), the technique is called *stochastic gradient descent*. In this case, the method follows the sample of the gradient:

$$\mathbf{a}_{l+1} = \mathbf{a}_l - \eta_{l+1} \nabla \hat{Q}^{\pi_B}(b, \mathbf{a}_l) \ , \tag{31}$$

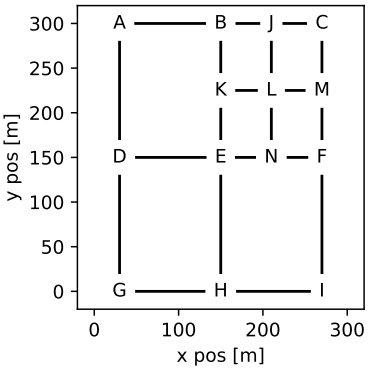


Fig. 3. Visualization of the quadrant search method.

where $\nabla\hat{Q}^{\pi_B}$ is a sample of the true gradient ∇Q^{π_B}, determined via execution of the rollout, l the iteration and η_{l+1} the - commonly decaying - step size. As the execution of the rollout is not differentiable, we use a finite difference approximation to the gradient, using two two-sided differences. Those four evaluation points use CRN to reduce the variance of the gradient estimate. However, the samples of the gradient at different iterations are not correlated. This algorithm is derived from stochastic approximation theory, especially the Kiefer-Wolfowitz algorithm [49], [50] and is commonly used to optimize machine learning models [51].

We also evaluate a version based on deterministic samples. As the evaluation of the action value is then a deterministic problem, standard optimization methods can be used, for example BFGS, which is an effective quasi-Newton method [52]. We used the implementation from JSAT [53], version 0.0.9.

VI. Experimental Methodology

We evaluated the different action search methods based on how well they can find the minimum of the action value Q^{π_B}. For this, it was required to determine the true minimum, which is not a-priori obvious. Therefore, a set of belief states was sampled, for which a close approximation to the minimum was computed offline. Then the optimization methods were tested on this set of beliefs.

To create the belief set, we ran the planner from [34] 20 times with different random seeds, and saved at each decision point the belief state. We used Scenario 2 from the prior work, which can be seen in Figure 4. Here the target position is drawn for each run from a Gaussian distributed prior, which is known to the localizer. The parameters correspond to the ones in the prior work, and are summarized in Table II. After exclusion of the beliefs where the target had been localized and only including the initial belief state once, this led to a set $\mathbb{B}$ of 25 distinct belief states.

For an approximation of the true optimal action value, the action space was discretized on the whole scenario area with an 150×150 action grid, i.e. with 2 m distance between the actions. For each action $\mathbf{a}$ and belief b we estimated the value of $Q^{\pi_B}(b,\mathbf{a})$, using sufficient random samples such

TABLE II
Parameters of the scenario

Symbol	Parameter	Value
v_p	Speed	5 m/s
t_M	Measurement time	10 s
σ	Measurement accuracy	4°
μ_T	Localization accuracy threshold	5 m

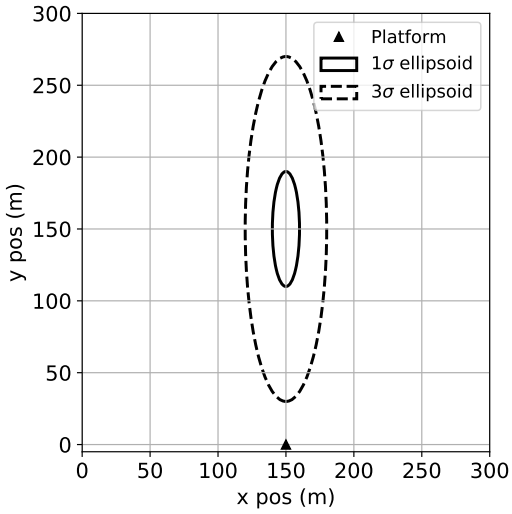


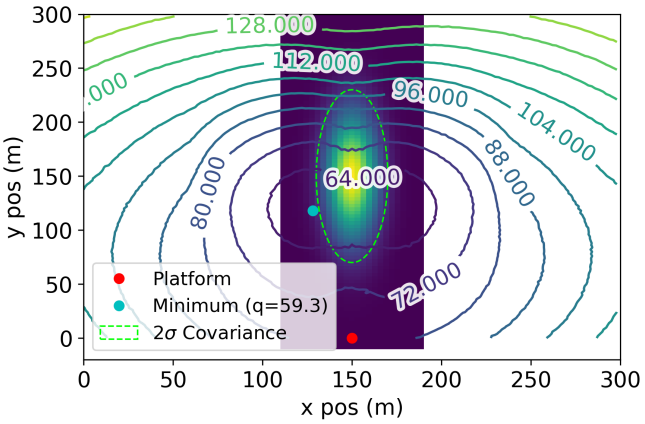
Fig. 4. Scenario used for evaluation.

that the standard error of the mean was below 0.1. This was around $2000-20\,000$ samples, depending on the action and belief. The results of these computations can be seen in Figure 5 for two exemplary beliefs. With q_b^{opt} we denote the minimal Q-value found in this step for belief b. While this is only an approximation, under a visual inspection the function $Q^{\pi_B}(b,\mathbf{a})$ appeared very continuous on the 150×150 action grid, and therefore it is not expected that the true optimum is significantly different.

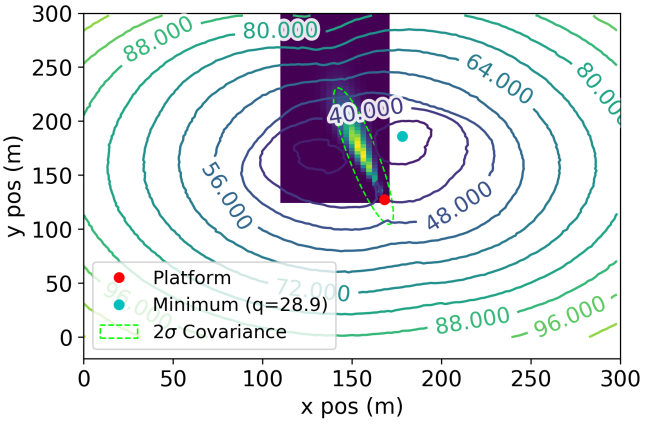
It can be seen that the function $Q^{\pi_B}(b,\mathbf{a})$ has typically two minima, which are approximately symmetric to the major axis of the Gaussian approximation $(\tilde{\mathbf{x}},\tilde{\mathbf{P}})$ of the belief. This is similar to the points M_1 and M_2 in the base policy (see Figure 2a). However, the optimal action was sometimes nearer to the target estimate $\tilde{\mathbf{x}}$ than the action chosen by the base policy. In addition, the optimal action was not always placed on a direct extension of the minor axis.

We then executed the control algorithm on each belief, using the combinations as indicated by Table I for action selection. The deterministic methods were executed once, the Monte Carlo based methods 100 times. For each selected action during evaluation run r and belief b, we computed the approximately true Q-value q_{br} of the selected action by linear interpolation of the 150×150 action grid.

Therefore, for each method we can compute the mean difference between the Q-value of the selected action to the



(a) Belief 0



(b) Belief 16

Fig. 5. Estimation of the Q-value using a sufficiently high number of samples. The heat map shows the grid-based belief representation, while the contours represent the Q-values. The ellipse shown is the Gaussian approximation $(\tilde{\mathbf{x}}, \tilde{\mathbf{P}})$ of the belief.

optimum q_b^{opt} of the corresponding belief:

$$\frac{1}{\|\mathbb{B}\|}\frac{1}{N^e}\sum_{b\in\mathbb{B}}\sum_{r=0}^{N^e-1} q_{br} - q_b^{\text{opt}}\ , \tag{32}$$

where N^e is the number of evaluations of the method, which was either 1 or 100, and q_{br} the Q-value for belief b and run r. This mean distance to the optimum can also be considered as the *optimization performance* of a method, taking the intuition that the action search is a stochastic optimization problem to find the action with minimal $Q^{\pi_B}(b, \mathbf{a})$.

VII. Results

For improved clarity, the presentation of the results is separated into three sections, where the first discusses the uniform allocation and sequential halving, and the other two the quadrant and gradient methods. The methods are split up in this way, as the first set of methods considers each action independently; therefore, these methods are also feasible for arbitrary action spaces. The second set of methods assume that the action value function is continuous, i.e. that evaluating an action also gives information about nearby actions.

The results are shown as the optimization performance (32) dependent on the required number of rollouts. A method is considered Pareto optimal, if no other method achieves a better optimization performance with less or an equal number of rollouts.

A. Uniform allocation and sequential halving

Figure 6 shows the results of uniform allocation and sequential halving. The uniform allocation uses an 10×10 and 20×20 action grid, and the number of samples is varied over powers of two. I.e. the first data point of each line corresponds to a single sample and either $100\ (= 10\cdot10\cdot1)$ or $400\ (= 20\cdot20\cdot1)$ rollouts. The following data points correspond to $2, 4, 8, \ldots$ samples.

The sequential halving algorithm is evaluated on the same action grids with a computational budget N, i.e. total number of rollouts, of $700, 1400$ and 2100 for the 10×10 grid and $3600, 7200, 10\,800$ for the 20×20 grid. These are chosen as multiples of $100\cdot\lceil\log_2(100)\rceil$ and $400\cdot\lceil\log_2(400)\rceil$. Due to the rounding, the actually used number of rollouts are $689, 1391, 2093$, and $3589, 7191, 10\,793$.

For the uniform allocation, it can be seen that using common random numbers is strictly better than plain Monte Carlo samples. It can also be seen that the approach with deterministic samples achieves better results than using common random numbers. However, while the Monte Carlo approaches improve monotonically for increasing number of samples, the same is not true for the deterministic approach. Sometimes it has a worse result for a higher number of used samples. This is likely because the deterministic samples are a suboptimal approximation method for the density. It should be noted that because of the deterministic nature of the algorithm it naturally also misses the averaging effect that multiple runs have on the Monte Carlo algorithms.

It can also be seen that for a small computational budget N it is better to have a smaller action space, where the actions are evaluated more often, than a bigger action space with potentially better actions, but which cannot be evaluated that often. However, for higher budgets the bigger action space shows better results.

The sequential halving algorithm shows consistently better results than uniform allocation, as it can focus on the most promising actions. Here each evaluated configuration is Pareto optimal. Uniform allocation with 64 deterministic samples on the 20×20 grid shows approximately the same optimization performance as sequential halving with common random numbers and a budget of $10\,800$. However, it required more than twice the number of rollouts ($25\,600$ vs $10\,793$).

These optimization methods are fundamentally limited, in that they will not be able to select an action not on their action set. The optimum is computed over an action grid with a very high resolution (see Section VI). This optimum does not necessarily have to be on the 10×10 or 20×20 action grid, therefore these methods likely would not reach zero even with a high amount of rollouts.

Finally, the base policy itself would score 2.98 on this chart, which shows that the rollout improvement property cannot

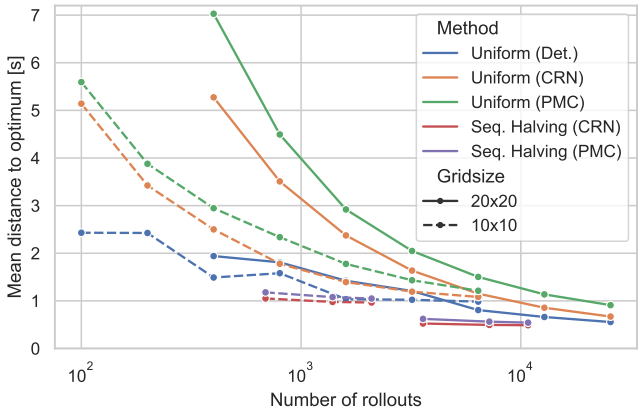


Fig. 6. Results of the uniform allocation and sequential halving.

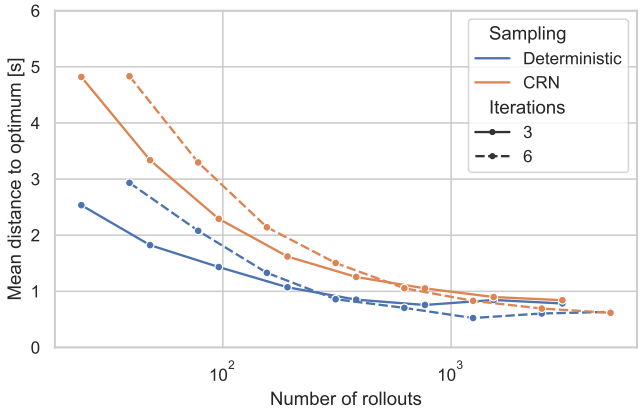


Fig. 7. Results of the quadrant search.

be achieved when the estimation of the action values is too inaccurate.

B. Quadrant search

Figure 7 shows the results of the quadrant search. For a given number of iterations, we vary the number of samples, again being powers of two. For example, the first point with a single sample and three iterations corresponds to $9 + 3 \cdot 5$ rollouts. The algorithm is at the risk of selecting the wrong quadrant, as the optimum is not guaranteed to be in the quadrant whose corners have on average the best values. Because the algorithm makes a hard decision on the quadrant, it would not be able to recover from such a choice. Therefore, even with perfect evaluation of the actions, the algorithm would not necessarily converge to zero. However, this method works surprisingly well, and achieves a good performance for a low number of rollouts.

Similar as in the case of uniform allocation, we can see that the approach with deterministic samples is better than the one with common random numbers.

C. Gradient-based methods

Figure 8 shows the results of the gradient-based methods. For the stochastic gradient descent (SGD) the number of iterations N^l is varied, with a fixed number of samples to estimate the gradient. As an example, SGD with one sample and 25 iterations would require a total number of 100 rollouts, as it uses two two-sided finite differences.

We used a finite difference size of ± 20 and an exponential decay of η_l with

$$\eta_l = 20 \cdot \exp\left(-4 \frac{l}{N^l - 1}\right) . \tag{33}$$

With iteration $l = 0, 1, \ldots N^l - 1$, the above step size schedule interpolates between $\eta_0 = 20$ and $\eta_{N^l-1} = 20 \cdot \exp(-4) \approx 0.366$. We did not perform an explicit hyperparameter optimization of the step size. The algorithm was initialized with the action selected by the base policy, which seems to be a reasonable choice in the policy rollout case.

For BFGS the same initialization and finite difference size was used, and the gradient was computed using the deterministic samples. We terminated the algorithm once a limit of evaluations of $Q^{\pi_B}(b, \mathbf{a})$ (including those to estimate the gradient) was reached, and varied this limit. The total number of rollouts then consists of this limit multiplied by the number of samples used to estimate the Q-value.

The stochastic gradient descent method shows very good results, being Pareto optimal to BFGS, except in a single case where the BFGS method is better. It also shows a monotonic improvement of the optimization performance with increasing number of iterations. For a high number of rollouts, it achieves a lower error as the previous methods. However, a convergence to the optimum is not guaranteed as those methods can become stuck in local optima.

Interestingly, BFGS shows almost no improvement when a higher number of rollouts is used. A likely explanation is that the estimate of the Q-values with deterministic samples is not as smooth as the true Q-value function. An example of computing the Q-values of the initial belief with two deterministic samples can be seen in Figure 9. One should note that the same holds true for estimating the Q-value with a small number of Monte Carlo samples. However, the stochastic gradient descent performs a new sampling in each iteration, therefore approximating gradient descent on the true value.

D. Comparison

Each of the considered methods in this section has several degrees of freedom in their parameters. The choice of search method, sampling method, number of samples, and other method specific parameters gives a specific rollout implementation. The number of rollouts we can execute in practice will be limited by computational power of the platform. Therefore, when we use this algorithm in an actual system, we likely will have a computational budget of possible rollouts and want to select the algorithm components, such that we can make the best use of this computational budget. Figure 10 shows a subset of the results in a single figure, for direct comparison. This figure gives the optimization performance we can achieve

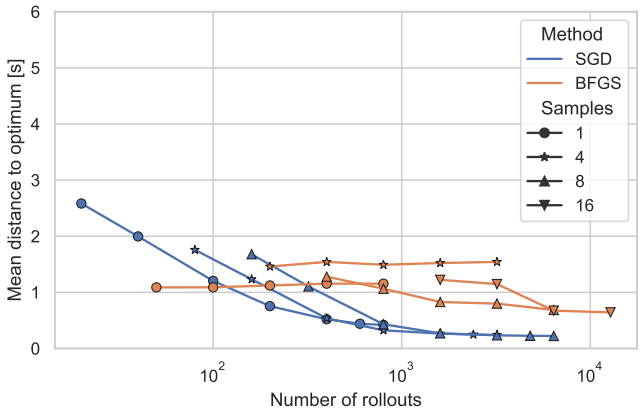


Fig. 8. Results of the gradient based methods.

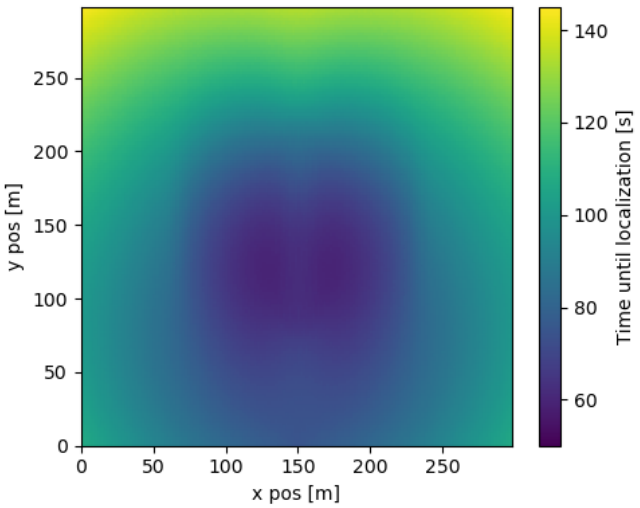
with a given number of rollouts and lets us compare the rollout implementations with a common measure. A method is Pareto optimal if no other method achieves a smaller mean distance to the optimum with lesser rollouts. It can be seen that the stochastic gradient descent approaches are almost everywhere Pareto optimal to the other approaches.

VIII. Correlation with Rollout Performance

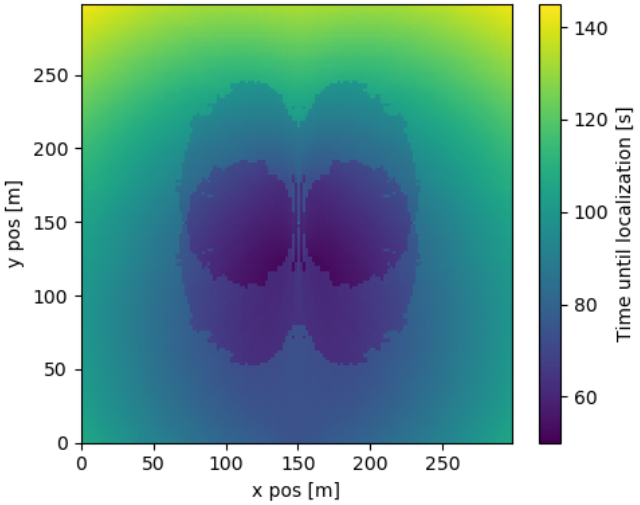
In the previous section, we evaluated the different methods to search for the optimal action on a sampled set of beliefs, where for each belief we had computed the approximately optimal action value. This gave us a way to quantify the optimization performance of a method, i.e. how close it comes to selecting the optimal action.

However, it is not self-evident whether this performance measure correlates with the performance of the resulting control algorithm. While one could expect that such a correlation exists, the base policy is only an approximation of future behavior. Therefore, non-minimum actions might be actually optimal. The rollout improvement property only guarantees that when the true minimizing action is chosen, the resulting control algorithm is at least as good as the base policy.

To evaluate how much the optimization performance correlates with the performance of the control algorithm, we evaluated the scenario with each action search configuration for 20 000 MC runs. Figure 11 shows the correlation between the optimization performance and the performance of the resulting control algorithm, given as mean time until localization. It can be clearly seen that a correlation exists (Pearson: r=0.775). The gradient-based algorithms show several outliers from this correlation, where the time until localization is worse than expected. The worse performance of the gradient-based methods is due to the initial belief. Here the base policy selects an action far from the optimal action. As this is used for the initialization for those methods, the optimal solution cannot be reached when the number of iterations is small. The grid-based methods and the quadrant-based method do not have this problem, as they do not require an initialization. We previously considered the initial belief as an independent belief. However, as it appears at the start of every simulation it appears more



(a) True value of $Q^{\pi_B}(b_0, \mathbf{a})$ for different $\mathbf{a}$



(b) Approximation with two deterministic samples

Fig. 9. Comparison of the approximately true Q-values of the initial belief and its estimate via two deterministic samples.

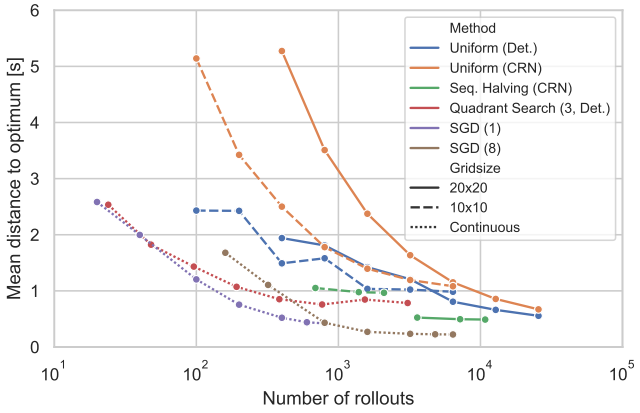


Fig. 10. A selected subset of the plots in Figure 6, 7, and 8.

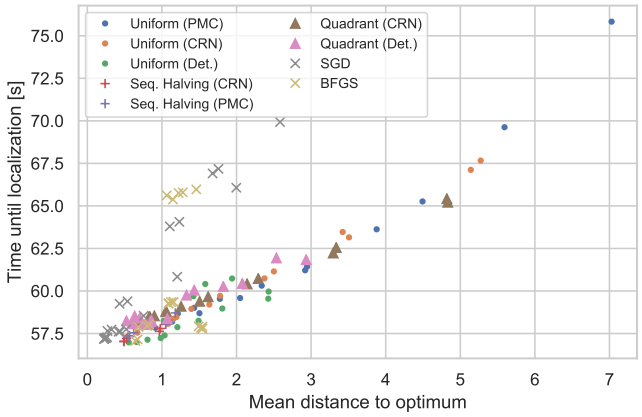


Fig. 11. Correlation between the optimizer performance and the rollout-algorithm performance.

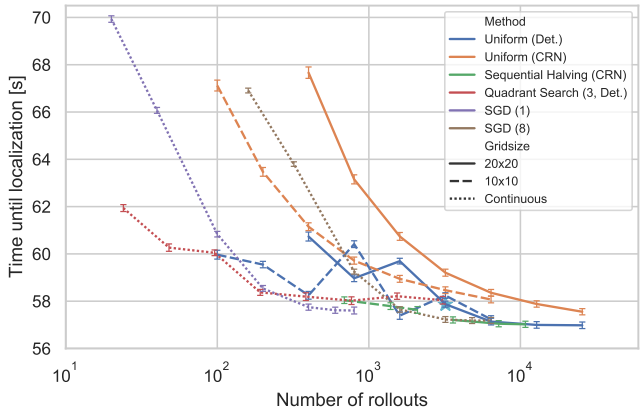


Fig. 12. Rollout performance for different number of rollouts, for a selected subset. Star denotes the method used in [34]. The error bars denote the 95% confidence interval.

frequently than the other beliefs. When correcting for this fact by weighting belief zero by the factor 20, the number of runs used to sample the beliefs, the correlation increases to r=0.965.

Figure 12 shows the actual performance of the same methods as shown in Figure 10. It can be seen that the optimization method in [34], denoted by a star, is not on the Pareto front. When compared to Figure 10 it can be seen that the stochastic gradient descent algorithms need a higher than expected number of samples to achieve reasonable performance, and the quadrant search algorithm dominates for a small computational budget time. This can be explained due to the initial belief, as described before. However, for a high computational budget, the best results are achieved by the stochastic gradient descent and sequential halving, as one would expect from Figure 10. Interestingly, the uniform allocation via deterministic samples achieves comparable results on the actual performance when using enough samples, even though it has a lower optimizing performance in Figure 10.

To compare these results with classic sensor management methods, we simulated the performance of a myopic entropy-based planner, similar to the one used in [2] for the same problem. This planner evaluates at each step the possible next positions by computing the expected entropy of the posterior. It does this by updating the current target estimate with a hypothetical noise-free measurement based on the mean of the target position estimate. Then the position with minimal entropy is chosen as the next measurement position. We chose a 60×60 action grid for the possible measurement positions. For the given scenario, this planner results in a time until localization of 67.07 ± 0.17 seconds. While some of the configurations are worse than this, e.g. stochastic gradient descent with a minimal number of steps, in most cases the rollout-based planner shows a clear improvement.

Figure 13 shows two exemplary paths found by the methods. This figure should give an intuition about how the resulting paths look like. The exact shape is dependent on the target position (which is here close to the expected value), the measurements, and the random components of the planner.

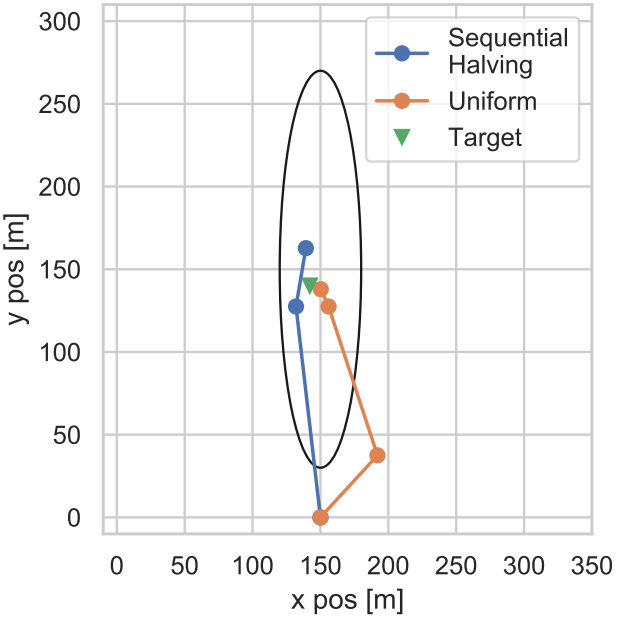


Fig. 13. Exemplary paths for sequential halving (using CRN with a 20×20 grid and a budget of 7200) and uniform allocation, on the same grid using 1 plain Monte Carlo sample per evaluation.

IX. Discussion

In this section, we discuss which conclusions we can take from our results. First, we note that the usage of common random numbers should be preferred to plain Monte Carlo when comparing action values.

A deterministic approximation to the uncertainty in the target estimate improves the performance of some methods, however, not necessarily. A likely explanation is that it improves the results in cases where the search is almost global (uniform allocation or quadrant search), however, leads to worse results in cases where the search is local (BFGS), as it might produce local minima. Different to the Monte Carlo methods, it did not produce a monotonic improvement in

optimization performance and performance of the resulting control algorithm.

For the given problem, there were two sources of uncertainty: uncertainty in the current state estimate and uncertainty in the future measurements. The deterministic samples only considered the first source of uncertainty and used the expected value for the second one. The implicit assumption is that it is more important to capture the uncertainty in the current state estimation than the uncertainty in the future measurements. This seems to work for the given problem, however, does not necessarily need to generalize to other problems where the future measurements might have a higher influence.

Sequential halving is an effective method to search for the optimal action with discrete or discretized action sets. This is likely to extend to other methods of adaptive action evaluation. As it uses Monte Carlo samples of the state and measurements, it does not need to make any assumptions about the uncertainty. An additional advantage is that given the action set, the only hyperparameter to be determined is the computational budget. A disadvantage is that it requires at least a computational budget of $N = |\mathbb{A}|\lceil \log_2 |\mathbb{A}| \rceil$.

Quadrant search works well and is worth considering when the computational budget is low. As it focuses on a single quadrant after evaluating only 9 actions, it has the possibility to focus the search on the wrong quadrant, however, this seems not to be an issue for the considered problem.

Stochastic gradient descent shows the strongest results when considering its optimization performance on the belief set. For a small computational budget, the resulting control algorithm is less effective than expected. However, with a high computational budget, the control also shows a strong performance, and its strong optimization performance makes it worth considering. In this work, we only evaluated a single step size schedule. This parameter has likely a significant and potentially problem dependent influence on the performance. Therefore, a higher performance might be possible, however, at the cost of more intensive hyperparameter tuning.

While we did not consider any additional constraints on the action space, this is an important consideration. It is easy to incorporate constraints into uniform allocation and sequential halving, as actions could simply be excluded from the action set. The gradient-based approaches could either use a regular reprojection into the action space if they move to a forbidden action or a method as L-BFGS-B [54] which allows for constraints could be used. The quadrant search could use the allowed action space to place its initial 3×3 action grid if the constraints are rectangular, however, it is not clear whether this would work for arbitrary constraints.

X. Robustness and Generalization

In this section, we perform a robustness analysis of the discussion in the last section. We vary parameters in the scenario and planner and determine whether our findings still hold.

In a first analysis, we analyze the algorithm performance for several choices of the action grid size. Figure 14 shows the time until localization for different grid sizes N_g and 20 000 MC runs. The computational budget is set to a fixed number of 16 samples per action for the uniform method. The sequential halving method receives an equivalent budget of $16 \cdot N_g \cdot N_g$. The performance strongly improves until a grid size of 10. After this, we see diminishing returns for higher grid sizes. For small grid sizes, we also see alternating behavior for even and odd sizes, as they contain different actions. For all action grid sizes, we can see that common random numbers are preferable to plain Monte Carlo sampling. We can also see that for the same budget sequential halving shows better results. Finally, we can also see that the deterministic sampling shows a non-monotonic behavior. This is similar as observed previously, e.g. in Figure 12.

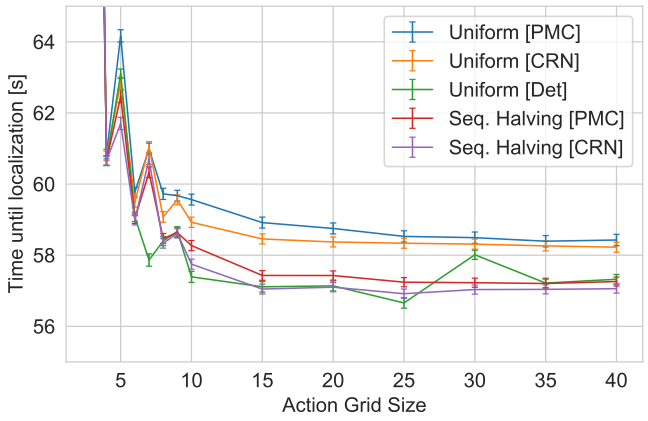


Fig. 14. Variation of the action grid size. A fixed budget of 16 samples per grid cell is chosen. The x-axis gives the number N_g of an $N_g \times N_g$ grid. The error bars denote the 95% confidence interval.

Figure 15 shows the time until localization for different required times for a single measurement with 20 000 MC runs. Different measurement times lead to different trade-offs between taking a measurement and moving to another measurement position.

We compare the stochastic gradient descent method with four steps and one sample per gradient computation (40 rollouts in total), the quadrant search with deterministic sampling, three iterations, and one sample per action (24 rollouts in total) and the sequential halving method with a sampling budget of 3600. For comparison, we also added the myopic entropy-based method. The figure shows that the advantage of quadrant search to stochastic gradient descent for small number of rollouts is robust over different scenario instantiations. We can also see that for a small number of rollouts, the entropy method performs approximately as well as the rollout method with stochastic gradient descent for action selection. However, the better performing quadrant search and the sequential halving with a higher budget outperform the entropy-based method for all variations of the measurement time.

Finally, we evaluate the methods on a different scenario than in the main part of the paper. Figure 16 shows the geometry of this scenario. The targets now appear uniformly in space on a ring around the platform, with a distance of $30-300$ m. In this scenario, the platform makes an initial measurement directly at the beginning from its starting position, after which the initial

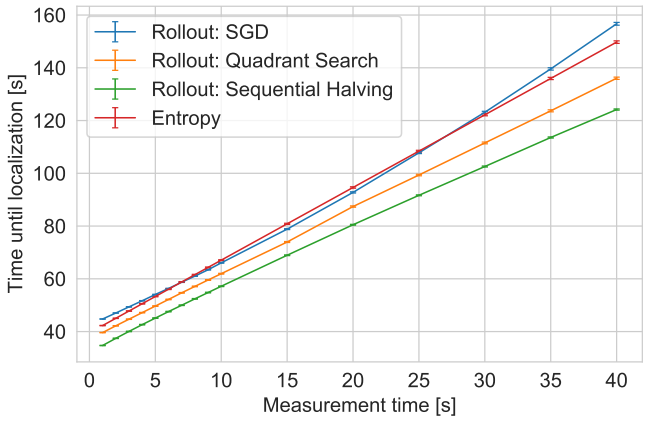


Fig. 15. Variation of the required time for a measurement. The stochastic gradient descent and quadrant search use a small computational budget with 40 and 24 rollouts. The sequential halving has budget of 3600 and a 20×20 grid. The error bars denote the 95% confidence interval.

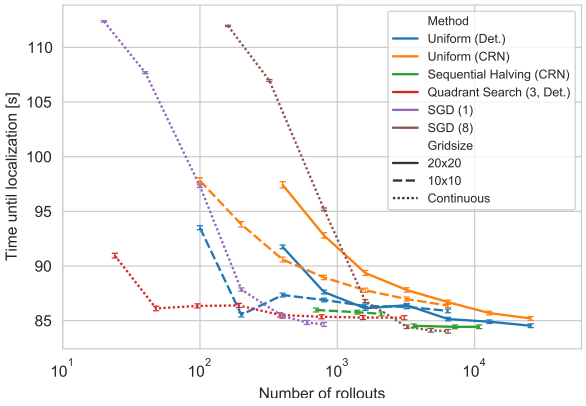


Fig. 17. Rollout performance for a different number of rollouts for the scenario with a uniform prior. The error bars denote the 95% confidence interval.

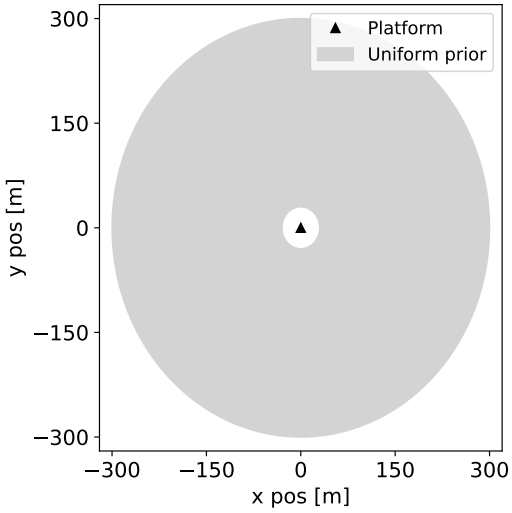


Fig. 16. Scenario with a uniform prior.

estimate is determined. Contrary to the previous localizer, the estimate is made using the expected value of the estimated probability density, instead of the maximal likelihood. The remaining parameters are kept at the same values.

Figure 17 shows the results of 20 000 MC runs on this scenario. It shows the same methods as in Figure 10. As in the previous scenario, we evaluated an entropy-based planner as comparison, resulting in an average localization time of 98.22 ± 0.2 seconds. While there are differences in the absolutes of the numbers, both figures show the same behaviors: The rollout-based approaches are in most cases better than the myopic entropy-based planner. The quadrant search is the best for a low number of rollouts. The methods based on deterministic samples show a non-monotonic improvement. Sequential halving is the best method for selecting from a finite set. We also see a fast improvement with additional samples in the stochastic gradient. This effect is even more pronounced in this scenario.

In this paper, the altitude of the UAV was not considered. The algorithms would extend straightforwardly to a three-dimensional case, where the sensor platform is able to change its altitude. As long as the target remains stationary and on the surface, the main adaption would be a search through a three-dimensional action space, instead of a two-dimensional. The methods considered in this paper would still be feasible, with the quadrant search then becoming an octant search. However, due to the higher dimension of the action space we would expect that methods that intelligently evaluate the actions (i.e. sequential halving, octant search, and stochastic gradient descent) become more useful with more dimensions than a uniform allocation.

Generalization to a moving emitter would also be feasible, however, would lead to additional challenges not considered in this paper. The first is the addition of a prediction model, which would expand the uncertainty in the target localization over time due to the target movement. The second would be the drawing of the samples. For a stationary emitter only a single position is sufficient, for a moving emitter its path need to be sampled. Especially, this sampling of the path needs to correspond to the true probabilities of target movement to approximate the expected future cost properly. An assumption that the target moves uniformly in its possible movement space, is likely not correct. Finally, the termination criterion of the rollout needs to be changed. In the presented algorithm, the rollout terminates once the emitter is localized. However, even if a moving emitter is localized, it will move away from this localization again. In this case, a rolling horizon approach would be more useful.

XI. Conclusion

In this paper, we propose and evaluate methods for the action selection step in a policy rollout algorithm, which have not been used for this purpose before. We compared the methods based on their ability to reach the minimal Q-value in comparison to the required number of rollouts, as well as by the performance of the resulting control algorithm. Our evaluation problem was to control a UAV, equipped with a

bearing only sensor, which should localize an emitter as fast as possible.

With this work, we made the following contributions:

- We have shown the application of policy rollout in a problem with a continuous action space. Prior work was mostly focused on discrete action spaces.
- We evaluated multiple methods to find the optimal action in a rollout. Prior work commonly chose Monte Carlo sampling with a fixed and equal number of rollouts for each action. We have shown that with successive halving also for a discrete action space methods that are more effective exist. To the authors' knowledge, this is the first time stochastic gradient descent was considered in the policy rollout context.
- We have shown that the optimization performance of the action selection algorithm correlates with the actual performance of the policy rollout. This seems to be intuitively clear, however, enforces our belief that optimizing this algorithmic component is important.
- Finally, we have described different variants with improved results of the UAV control algorithm previously presented in [34].

Due to space constraints, this analysis of optimization methods is only partial and some interesting methods have been omitted. It would be interesting for future work to evaluate other best-arm methods from the bandit literature, generalizations of bandits into continuous spaces [39], simulation optimization methods [37] or response surface modeling [55]. The strong optimizing performance of stochastic gradient descent also indicates that a further exploration of this method in the context of policy rollout would be worthwhile. Research in stochastic gradient descent led to several improved variants, mostly in the context of training machine learning models. It would be interesting to consider methods like e.g. Adam [56] or AdaGrad [57] in the context of policy rollout.

REFERENCES

[1] D. P. Bertsekas and D. A. Castañón, "Rollout Algorithms for Stochastic Scheduling Problems," *Journal of Heuristics*, vol. 5, no. 1, pp. 89–108, 1999.

[2] O. Cliff, R. Fitch, S. Sukkarieh, D. Saunders, and R. Heinsohn, "Online Localization of Radio-Tagged Wildlife with an Autonomous Aerial Robot System," in *Robotics: Science and Systems XI*. Rome, Italy: Robotics: Science and Systems Foundation, 2015.

[3] K. Vonehr, S. Hilaski, B. E. Dunne, and J. Ward, "Software Defined Radio for Direction-Finding in UAV Wildlife Tracking," in *IEEE International Conference on Electro Information Technology*. Grand Forks, ND, USA: IEEE, 2016, pp. 464–469.

[4] J. T. Isaacs, F. Quitin, L. R. García Carrillo, U. Madhow, and J. P. Hespanha, "Quadrotor control for RF source localization and tracking," in *International Conference on Unmanned Aircraft Systems (ICUAS)*. Orlando, FL, USA: IEEE, 2014, pp. 244–252.

[5] J. Vander Hook, P. Tokekar, and V. Isler, "Algorithms for Cooperative Active Localization of Static Targets With Mobile Bearing Sensors Under Communication Constraints," *IEEE Transactions on Robotics*, vol. 31, no. 4, pp. 864–876, 2015.

[6] D. P. Bertsekas, *Dynamic Programming and Optimal Control, Vol. II Approximate Dynamic Programming*, 4th ed. Belmont, Massachusetts, USA: Athena Scientific, 2012.

[7] W. B. Powell, *Approximate Dynamic Programming*, 2nd ed. Hoboken, New Jersey: John Wiley Sons, Inc., 2011.

[8] G. Tesauro and G. R. Galperin, "On-line Policy Improvement using Monte Carlo Search," in *Advances in Neural Information Processing Systems 9*, Denver, CO, USA, 1996, pp. 1068–1074.

[9] D. P. Bertsekas, J. N. Tsitsiklis, and C. Wu, "Rollout Algorithms for Combinatorial Optimization," *Journal of Heuristics*, vol. 3, no. 3, pp. 245–262, 1997.

[10] E. K. P. Chong, C. M. Kreucher, and A. O. Hero III, "Monte-Carlo-Based Partially Observable Markov Decision Process Approximations for Adaptive Sensing," in *Proceedings of the 9th International Workshop on Discrete Event Systems*. Göteborg, Sweden: IEEE, 2008, pp. 173–180.

[11] 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, no. 3, pp. 377–422, 2009.

[12] A. Saksena and I.-J. Wang, "Dynamic Ping Optimization for Surveillance in Multistatic Sonar Buoy Networks with Energy Constraints," in *47th IEEE Conference on Decision and Control*. Cancun, Mexico: IEEE, 2008, pp. 1109–1114.

[13] Y. He and E. K. P. Chong, "Sensor Scheduling for Target Tracking in Sensor Networks," in *43rd IEEE Conference on Decision and Control*. Atlantis, Paradise Island, Bahamas: IEEE, 2004, pp. 743–748.

[14] Y. He and E. K. Chong, "Sensor scheduling for target tracking: A Monte Carlo sampling approach," *Digital Signal Processing*, vol. 16, no. 5, pp. 533–545, 2006.

[15] Y. Li, L. Krakow, E. Chong, and K. Groom, "Approximate stochastic dynamic programming for sensor scheduling to track multiple targets," *Digital Signal Processing*, vol. 19, no. 6, pp. 978–989, dec 2009.

[16] L. W. Krakow, E. K. P. Chong, K. N. Groom, J. Harrington, Y. Li, and B. Rigdon, "Control of Perimeter Surveillance Wireless Sensor Networks Via Partially Observable Marcov Decision Process," in *Proceedings of the 40th Annual International Carnahan Conference on Security Technology*, Lexington, KY, USA, 2006, pp. 1–8.

[17] Z.-n. Zhang and G.-l. Shan, "Non-myopic sensor scheduling to track multiple reactive targets," *IET Signal Processing*, vol. 9, no. 1, pp. 37–47, 2015.

[18] M. K. Schneider and C. Chong, "A rollout algorithm to coordinate multiple sensor resources to track and discriminate targets," in *Proceedings Volume 6235, Signal Processing, Sensor Fusion, and Target Recognition XV*, I. Kadar, Ed., Orlando (Kissimmee), Florida, 2006, p. 62350E.

[19] R. Zahedi, L. W. Krakow, E. K. P. Chong, and A. Pezeshki, "Adaptive Estimation of Time-Varying Sparse Signals," *IEEE Access*, vol. 1, pp. 449–464, 2013.

[20] S. Ragi, H. D. Mittelmann, and E. K. P. Chong, "Directional Sensor Control: Heuristic Approaches," *IEEE Sensors Journal*, vol. 15, no. 1, pp. 374–381, jan 2015.

[21] S. Beyme and C. Leung, "Rollout Algorithms for Wireless Sensor Network-Assisted Target Search," *IEEE Sensors Journal*, vol. 15, no. 7, pp. 3835–3845, 2015.

[22] D. Jun and D. L. Jones, "The value of sleeping: A rollout algorithm for sensor scheduling in HMMs," in *2013 IEEE Global Conference on Signal and Information Processing*. Austin, TX, USA: IEEE, dec 2013, pp. 181–184.

[23] A. Charlish and F. Hoffmann, "Anticipation in cognitive radar using stochastic control," in *2015 IEEE Radar Conference (RadarCon)*, Arlington, VA, USA, May 2015, pp. 1692–1697.

[24] M. W. Ulmer, J. C. Goodson, D. C. Mattfeld, and M. Hennig, "Offline–Online Approximate Dynamic Programming for Dynamic Vehicle Routing with Stochastic Requests," *Transportation Science*, vol. 53, no. 1, pp. 185–202, feb 2019.

[25] N. Secomandi, "Analysis of a Rollout Approach to Sequencing Problems with Stochastic Routing Applications," *Journal of Heuristics*, vol. 9, no. 4, pp. 321–352, 2003.

[26] ——, "A Rollout Policy for the Vehicle Routing Problem with Stochastic Demands," *Operations Research*, vol. 49, no. 5, pp. 796–802, oct 2001.

[27] C. Novoa and R. Storer, "An approximate dynamic programming approach for the vehicle routing problem with stochastic demands," *European Journal of Operational Research*, vol. 196, no. 2, pp. 509–515, 2009.

[28] J. C. Goodson, J. W. Ohlmann, and B. W. Thomas, "Rollout Policies for Dynamic Solutions to the Multivehicle Routing Problem with Stochastic Demand and Duration Limits," *Operations Research*, vol. 61, no. 1, pp. 138–154, feb 2013.

[29] L. Bertazzi, A. Bosco, F. Guerriero, and D. Laganà, "A stochastic inventory routing problem with stock-out," *Transportation Research Part C: Emerging Technologies*, vol. 27, pp. 89–107, 2013.

[30] D. Bertsimas and I. Popescu, "Revenue management in a dynamic network environment," *Transportation Science*, vol. 37, no. 3, pp. 257–277, 2003.

[31] A. McGovern and E. Moss, "Scheduling straight-line code using reinforcement learning and rollouts," in *Advances in Neural Information Processing Systems*, Denver, CO, USA, 1999, pp. 903–909.

[32] R. Y. Rubinstein and D. P. Kroese, *Simulation and the Monte Carlo Method*. Wiley, 2016.

[33] D. Bertsekas, "Differential training of rollout policies," in *Proc of 35th Allerton Conference on Communication, Control and Computing*, Allerton Park, Illinois, USA, 1997, pp. 1–10.

[34] F. Hoffmann, H. Schily, A. Charlish, M. Ritchie, and H. Griffiths, "A Rollout Based Path Planner for Emitter Localization," in *Proceedings of the 22nd International Conference on Information Fusion (Fusion 2019)*, Ottawa, ON, 2019.

[35] V. Klumpp and U. D. Hanebeck, "Dirac Mixture Trees for Fast Suboptimal Multi-Dimensional Density Approximation," in *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*. Seoul, Korea: IEEE, 2008, pp. 593–600.

[36] Tao Sun, Qianchuan Zhao, P. Luh, and R. Tomastik, "Optimization of Joint Replacement Policies for Multipart Systems by a Rollout Framework," *IEEE Transactions on Automation Science and Engineering*, vol. 5, no. 4, pp. 609–619, oct 2008.

[37] C.-H. Chen and L. H. Lee, *Stochastic Simulation Optimization - An Optimal Computing Budget Allocation*. Singapore: World Scientific Publishing, 2011.

[38] V. Gabillon and A. Lazaric, "Rollout Allocation Strategies for Classification-based Policy Iteration," in *ICML 2010 Workshop on Reinforcement Learning and Search in Very Large Spaces*, Haifa, Israel, 2010, pp. 0–3.

[39] S. Bubeck, R. Munos, G. Stoltz, and C. Szepesvári, "X-armed bandits," *Journal of Machine Learning Research*, vol. 12, pp. 1655–1695, 2011.

[40] C. Mansley, A. Weinstein, and M. L. Littman, "Sample-based planning for continuous action Markov decision processes," in *ICAPS 2011 - Proceedings of the 21st International Conference on Automated Planning and Scheduling*, Freiburg, Germany, 2011, pp. 335–338.

[41] L. Dressel and M. J. Kochenderfer, "Pseudo-bearing Measurements for Improved Localization of Radio Sources with Multirotor UAVs," in *IEEE International Conference on Robotics and Automation (ICRA)*. Brisbane, Australia: IEEE, 2018, pp. 6560–6565.

[42] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time Analysis of the Multiarmed Bandit Problem," *Machine Learning*, vol. 47, no. 2-3, pp. 235–256, 2002.

[43] S. Bubeck, R. Munos, and G. Stoltz, "Pure exploration in multi-armed bandits problems," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 5809 LNAI, pp. 23–37, 2009.

[44] J.-Y. Audibert and S. Bubeck, "Best Arm Identification in Multi-Armed Bandits," in *COLT - 23th Conference on Learning Theory*, Haifa, Israel, 2010, p. 13 p.

[45] V. Gabillon, M. Ghavamzadeh, and A. Lazaric, "Best Arm Identification: A Unified Approach to Fixed Budget and Fixed Confidence," in *Advances in Neural Information Processing Systems 25*. Lake Tahoe, Nevada, USA: Curran Associates, Inc., 2012, pp. 3212–3220.

[46] A. Garivier and E. Kaufmann, "Optimal Best Arm Identification with Fixed Confidence," in *COLT - 29th Conference on Learning Theory*, New York, USA, 2016, pp. 1–30.

[47] Z. Karnin, T. Koren, and O. Somekh, "Almost optimal exploration in multi-armed bandits," in *Proceedings of the 30th International Conference on Machine Learning (ICML)*, vol. 28, 2013, pp. 1238–1246.

[48] M. L. Hernandez, "Optimal Sensor Trajectories in Bearings-Only Tracking," in *The 7th International Conference on Information Fusion (FUSION)*. Stockholm, Sweden: IEEE, 2004, pp. 1–8.

[49] J. Kiefer and J. Wolfowitz, "Stochastic Estimation of the Maximum of a Regression Function," *The Annals of Mathematical Statistics*, vol. 23, no. 3, pp. 462–466, 1952.

[50] H. J. Kushner and G. G. Yin, *Stochastic Approximation and Recursive Algorithms and Applications*, 2nd ed. New York, NY, USA: Springer, 2003.

[51] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[52] J. Nocedal and S. J. Wright, *Numerical Optimization*. Springer New York, 2006.

[53] E. Raff, "Jsat: Java statistical analysis tool, a library for machine learning," *Journal of Machine Learning Research*, vol. 18, no. 23, pp. 1–5, 2017.

[54] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu, "A Limited Memory Algorithm for Bound Constrained Optimization," *SIAM Journal on Scientific Computing*, vol. 16, no. 5, pp. 1190–1208, sep 1995.

[55] D. R. Jones, "A Taxonomy of Global Optimization Methods Based on Response Surfaces," *Journal of Global Optimization*, vol. 21, no. 4, pp. 345–383, 2001.

[56] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015*, San Diego, CA, USA, 2015.

[57] J. Duchi, E. Hazan, and Y. Singer, "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization," *Journal of Machine Learning Research*, vol. 12, pp. 1–40, 2011.

Folker Hoffmann Folker Hoffmann received his M.Sc. in computer science from the University of Bonn and is currently enrolled as a Ph.D. student at University College London. Since 2014 he works as a research associate in the Sensor Data and Information Fusion (SDF) Department at the Fraunhofer Institute for Communication, Information Processing and Ergonomics (FKIE). His research interests include the use of stochastic control methods applied to sensor management problems.

Alexander Charlish Alexander Charlish (SM'19) obtained his M.Eng. degree from the University of Nottingham and received his Ph.D. degree from University College London on the topic of multifunction radar resources management. In 2011, he joined the Sensor Data and Information Fusion (SDF) Department at the Fraunhofer Institute for Communication, Information Processing and Ergonomics (FKIE), where he now leads the Sensor and Resources Management Group. In this role, he leads a group of scientists conducting research on intelligent sensing with a focus on cognitive radar and resources management for sensor systems. Additionally, he is a visiting lecturer at RWTH Aachen University. He is currently serving as an Associate Editor for Radar Systems for IEEE Transactions on Aerospace and Electronic Systems, and as Subject Editor for Radar, Sonar and Navigation for IET Electronic Letters. He is currently an elected member of IEEE AESS Radar Systems Panel and the AESS Board of Governors. He is also active in the NATO community and received the NATO SET Panel Excellence Award in 2019 and the NATO SET Panel Early Career Award in 2020.

Matthew Ritchie Dr. Matthew Ritchie received an MSci degree in physics from The University of Nottingham, in 2008. Following this he completed an Eng.D degree at University College London (UCL), in association with Thales U.K., in 2013. He continued at UCL as a postdoctoral research associate focusing on machine learning applied to multi-static radar for micro-Doppler classification. In 2017 Dr. Ritchie took a Senior Radar Scientist position at the Defence Science and Technology Laboratories (Dstl) which also involved working as the Team Leader for the Radar Sensing group in the Cyber and Information Systems Division. During his time at Dstl he worked on a broad range of cutting edge RF sensing challenges collaborating with both industry and academia. As of 2018 he has now taken a lectureship role at UCL within the Radar Sensing group. Currently he serves as the Chair of the IEEE Aerospace and System Society (AESS) for the United Kingdom & Ireland, is a Subject Editor-in-Chief for the IET Electronics Letters journal and a Senior Member of the IEEE.

Hugh Griffiths Hugh Griffiths (Fellow, IEEE) received the M.A. degree in physics from Oxford University in l975, and the Ph.D. and D.Sc. (Eng.) degrees from University College London in 1986 and 2000, respectively. He holds the THALES/Royal Academy Chair of RF sensors with the Department of Electronic and Electrical Engineering, University College London, U.K. From 2006 to 2008, he was Principal of the Defence Academy of Management and Technology, Shrivenham. From 2001 to 2006, he served as Head of Department at University College London. He has published more than 500 papers and technical articles in the fields of radar, antennas, and sonar. His research interests include radar and sonar systems, signal processing (particularly synthetic aperture radar and bistatic radar), and antenna measurement techniques. He carried out some of the first experiments in passive radar.

Dr. Griffiths was elected to Fellowship of the Royal Academy of Engineering in 1997. He was a member of the IEEE AES Radar Systems Panel since 1989. He is a Fellow of the IET. He received the IEEE AESS Nathanson Award in 1996, the IET A F Harvey Research Prize in 2013, and the IEEE Picard Medal in 2017. He has also received the Brabazon Premium of the IERE and the Mountbatten and Maxwell Premium Awards of the IEE. He served as the President of the IEEE AES Society from 2012 to 2013. He chaired the working group which revised the IEEE Radar Definitions Standard P686 and reaffirmed the Radar Letter Band Standard in 2008. He was appointed as Officer of the Order of the British Empire (OBE) in the 2019 Queen's New Year's Honours List.