# Structured Synthesis for Probabilistic Systems

Nils Jansen[1], Laura Humphrey[2], Jana Tumova[3], and Ufuk Topcu[4⋆]

[1] Radboud University, Nijmegen, The Netherlands
[2] Air Force Research Laboratory, USA
[3] KTH Royal Institute of Technology, Sweden
[4] University of Texas at Austin, USA

**Abstract.** We introduce the concept of structured synthesis for Markov decision processes. A structure is induced from finitely many pre-specified options for a system configuration. We define the structured synthesis problem as a nonlinear programming problem (NLP) with integer variables. As solving NLPs is not feasible in general, we present an alternative approach. A transformation of models specified in the PRISM probabilistic programming language creates models that account for all possible system configurations by nondeterministic choices. Together with a control module that ensures consistent configurations throughout a run of the system, this transformation enables the use of optimized tools for model checking in a black-box fashion. While this transformation increases the size of a model, experiments with standard benchmarks show that the method provides a feasible approach for structured synthesis. We motivate and demonstrate the usefulness of the approach along a realistic case study involving surveillance by unmanned aerial vehicles in a shipping facility.

## 1 Introduction

The problem introduced in this paper is motivated by the following scenario stemming from the area of physical security. Consider a shipping facility equipped with a number of ground sensors to discover potential intruders. The facility operates unmanned aerial vehicles (UAVs) to perform surveillance and maintenance tasks. Intruders appear randomly, there are uncertainties in sensor performance, and the operation of the UAVs is driven by scheduling choices and the activation of sensors. Suitable models to capture such randomization, uncertainty, and scheduling choices are *Markov decision processes* (MDPs), where measures such as "the probability to encounter dangerous states of the system" or "the expected cost to achieve a certain goal" are directly assessable.

System designers may have to choose among a pre-specified family of possibly interdependent options for the system configuration, such as different sensors or the operating altitude of UAVs. Each of these options triggers different system behavior, such as different failure probabilities and acquisition cost. We call such possible design choices an underlying *structure of the system*; all

⋆ Partially supported by AFRL FA8650-15-C-2546 and Sandia National Lab 801KOB.

concrete instantiations of the system adhere to this structure. For instance, imagine a structure describing the option of installing one of two types of sensors. The cheaper sensor induces a smaller expected cost, while the more expensive sensor induces a higher probability of discovering intruders. The changes in the instantiations of the system are necessarily according to the structure, i. e., the replacement of one sensor type with the other. A question of interest is then which instantiation yields the lowest cost while it guarantees to adhere to a target specification regarding the desired probability.

We introduce *multiple–instance MDPs* (MIMDPs) as underlying semantic model for structured synthesis. Arbitrary expressions over system parameters capture a structure that describes dependencies between uncertain behavior and system cost. Each parameter has an associated finite set of values out of which it can be instantiated. Thereby, a MIMDP induces a finite family of MDPs. MIMDPs are inspired by *parametric MDPs* [3, 26, 35, 29] (pMDPs), whose transition probabilities are defined by functions over yet-to-be-specified parameters. However, the existing definitions of pMDPs only allow restrictions on parameter valuations in the form of continuous intervals. State-of-the-art methods as implemented in the tools `PARAM` [11], `PRISM` [12], or `PROPhESY` [21] do not support the definition of discrete sets of valuations. Consequently, to the best of our knowledge, these techniques cannot directly handle the scenarios we consider.

Another related approach to modeling structured systems is *feature-based modeling* [31], which allows to specify families of stochastic systems. Although feature-based modeling supports discrete parametrization, it does not directly offer parametrization of probabilities. Furthermore, it analyses a family in an all-in-one fashion as opposed to focusing on individual instantiations.

The formal problem considered in this paper is to compute an optimal instantiation of parameters and a control strategy for a given MIMDP subject to reachability specifications and cost minimization. We define this problem naturally as a *non–linear integer optimization problem* (NLIP). As a computationally tractable alternative, we present a transformation of an MIMDP to an MDP where all possible parameter instantiations are relayed to *nondeterministic choices*. The common language used for model specification in all available tools is the probabilistic programming language [19] originally developed for `PRISM`. We define the transformation of the MIMDP as a *program transformation*. By adding control variables to the transformed program, we keep track of all instantiations, ensuring system executions that are in accordance with the given structure. Computing a solution to the original problem is thereby reduced to *MDP model checking*, which is equivalent to solving a linear program (LP). From a practical viewpoint, the transformation enables the use of all capabilities of model checkers such as `PRISM` [12], `Storm` [30], or `IscasMC` [20].

We illustrate the feasibility of the proposed approach on several basic case studies from the `PRISM` benchmark suite [14]. We also report promising results for a more realistic case study based on the shipping facility example. In our experiments, we observe that the transformation from a MIMDP to an MDP

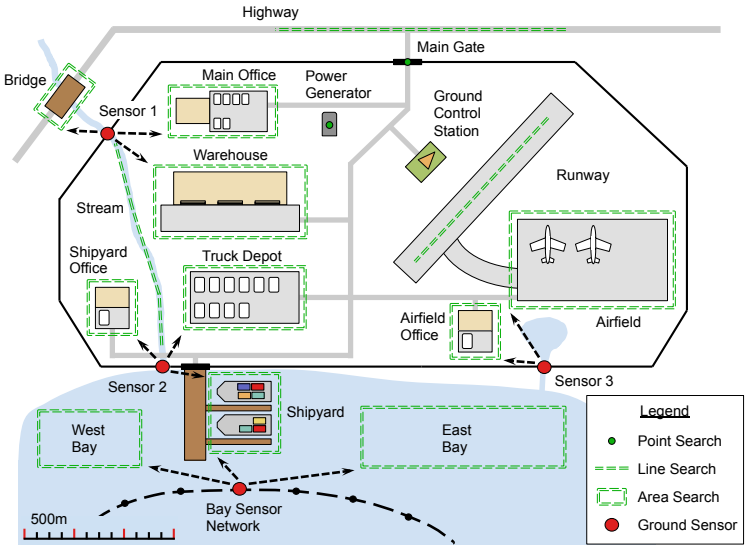



**Fig. 1.** A shipping facility that uses UAVs to perform surveillance tasks.

involves an increase in the number of states or transitions of up to two orders of magnitude. However, using an efficient model checker, we are able to demonstrate the applicability of our approach in examples with millions of states.

In summary, the contribution of this paper is threefold: i) We define a parametric model supporting discrete valuation sets and formalize the structured synthesis problem. ii) We develop a transformation of the parametric model to the `PRISM` language allowing us to practically address the structured synthesis problem. iii) We present a detailed, realistic case study of a shipping facility as well as experimental evaluation on standard benchmarks.

## 2 Case Study

In this section we introduce the case study that originally motivated the problem and the proposed approach. Several technical details are available in [33].

*Scenario.* Consider a shipping facility which uses one or more UAVs equipped with electro-optical (EO) sensors to perform surveillance tasks over various facility assets as shown in Fig. 1. These assets include an *Airfield* with a *Runway* and *Airfield Office*, a *Truck Depot*, a *Warehouse*, a *Main Office*, a *Shipyard* with a *Shipyard Office*, and a small bay that is partitioned into *West Bay*, *East Bay*, and *Shipyard* areas. An external *Highway* connects to the *Main Gate*, and a nearby *Bridge* crosses a *Stream* that cuts through the facility and empties into the bay. The facility is surrounded by a fence, but points where waterways run under the fence might allow intruders to enter. These points and the bay are monitored by

*Sensor 1*, *Sensor 2*, *Sensor 3*, and a *Bay Sensor Network*. All of these *ground sensors* can detect intruders with a certain false alarm rate.

UAVs take off and land from an area near the *Ground Control Station* (GCS). For each of the facility assets, a UAV can be tasked to perform a point, line, or area search as indicated in Fig. 1. Each of these search tasks requires the UAV to fly a certain distance to get to the task location, carry out the task, and fly back to the GCS. For point searches, simply flying to the point and back is enough. For line searches, the UAV must fly to one end, follow the line, and fly back from the other end. For area searches, the UAV must fly to one corner of the area, perform sequential parallel passes over it until the entire area has been covered by the UAV's sensor footprint, then fly back from the terminating corner. Note that the sensor footprint size increases with altitude, so more passes are needed to cover an area as a UAV's altitude decreases.

If a ground sensor reports that an intruder has been detected, a UAV may be tasked to fly to the respective area and perform a search; otherwise, the UAV can return to the GCS and continue on to another task. Probabilities for which areas intruders are likely to head toward might be estimated over time or assumed to be uniform if data are not available. We assume surveillance occurs frequently enough that at most one intruder will pass by a ground sensor before it is queried by a UAV. Similar problems involving UAVs searching for intruders based on ground sensor information are discussed in, e.g., [17] and [24].

*System Configuration, Safety, and Cost.* A configuration of the shipyard facility refers to the types of ground sensors and EO sensors installed onboard the UAVs. *Safety* of the shipping facility refers to the probability to successfully detect intruders, and *performance* describes the expected cost for the shipping facility.

Our goal is to find a configuration that ensures a certain safety probability on detecting intruders while minimizing cost. Different sensor types result in safety and performance tradeoffs for several reasons. First, each sensor type has a different one-time purchase cost. In turn, each sensor type has tunable parameters that result in a tradeoff between the probability of detecting an intruder and cost in terms of UAV flight time, with sensors that have a higher purchase cost providing a better tradeoff. The tradeoff between intruder detection and UAV flight time is also affected by the adjustable UAV operating altitude.

This tradeoff can be understood using two factors. The first is *ground sample distance* (GSD) [27], i. e., the number of meters per pixel of images sent back by a UAV, which depends on UAV altitude and EO sensor resolution. The second is the ground sensor *receiver operating characteristic* (ROC) [9], i. e., the tunable true positive versus false positive rate. Both true and false positives result in a UAV performing an area search for intruders. We now describe how probabilistic parameters relating to GSD and ROC can be adjusted by acquiring different types of sensors, tuning sensor parameters, or changing UAV operating altitude.

*Basic Task Costs.* For tasks that do not involve intruder detection, cost is driven mainly by manpower, logistics, and maintenance requirements, which roughly corresponds to cost per flight second $c_f$. Suppose the UAVs in this scenario all

fly at some standard operating ground speed $v_g$ measured in meters per second. The cost $c(t) = d(t)c_f/v_g$ for a task $t$ that does not involve intruder detection depends in a straightforward way on the distance $d(t)$ that a UAV must fly.

*Image GSD.* An important consideration for UAV surveillance tasks is the amount of visual detail a human operator needs to analyze, depending on the number of pixels comprising objects of interest in the images. GSD can be decreased by *decreasing altitude* or *increasing horizontal resolution* of the EO sensor. For this scenario, we consider three common EO sensor resolution options (480p, 720p, 1080p), with hypothetical purchase prices ($15k, $30k, $45k).

We use GSD in conjunction with the Johnson criteria [1] to estimate the probability that a human operator successfully analyzes an object. For each type of task and a corresponding digital image, a quantity $n_{50}$ defines the number of pairs of pixel lines across the "critical" or smaller dimension of an object needed for a 50% probability of task success. For instance, $n_{50} = 1$ for object detection.

Given $n_{50}$ and the number of pixels pairs $n$ across the critical dimension of an object (which depends on the size of the object and GSD), the probability for analysis success $p_d$ given sufficient time to analyze the image is estimated as

$$p_d = \frac{(n/n_{50})^{x_0}}{1 + (n/n_{50})^{x_0}} \quad \text{where} \quad x_0 = 2.7 + 0.7(n/n_{50}). \tag{1}$$

*Ground Sensor ROC.* The ROC curve of a sensor performing binary classification describes the tradeoff between the sensor's true positive rate/probability versus false positive rate/probability as the sensor's discrimination threshold is varied. Consider the three solid curves in Fig. 2. These represent hypothetical "low", "mid", and "high" cost ground sensors, with one-time purchase costs of $15k, $30k, and $45k, respectively. For each such ground sensor, the discrimination threshold can be varied to achieve an operating point on the corresponding curve. In order to reliably detect intruders, we need true positive rates to be fairly high. As the curves show, a high cost ground sensor provides the best tradeoff, since for each false positive rate, it provides a higher true positive rate. In our approach, we use quadratic approximations of the curves.

To help understand the effect ground sensors have on system costs and probabilistic parameters, suppose we choose to purchase a high cost sensor for Sensor 1. Clearly the purchase cost is higher than if we had chosen a mid or low cost sensor. However, each time the ground sensor generates a false positive, a UAV has to perform an unnecessary area search, which incurs additional system operational cost. To counteract this, we could decrease the ground sensor's false positive rate, but this would also decrease its true positive rate, resulting in a higher false negative rate. When a false negative occurs, the sensor fails to detect the presence of an intruder. A high cost sensor then mitigates operational cost by providing a lower false positive rate. Given these tradeoffs between purchase cost, operational cost, and probability of intruder detection, it is not clear which sensor minimizes cost while meeting safety specifications on intruder detection.

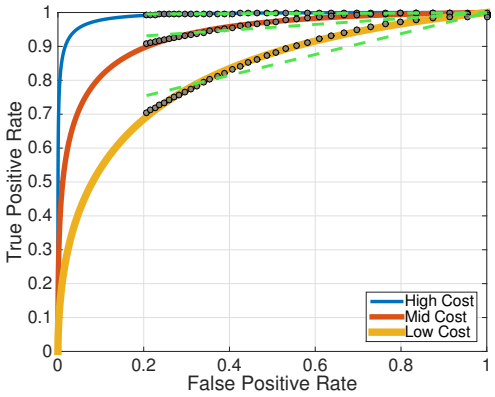



**Fig. 2.** ROC curves for different cost ground sensors. Linear and quadratic approximations are shown as green dashed lines and traces of black dots, respectively.

## 3 Preliminaries

A *probability distribution* over a finite set $X$ is a function $\mu\colon X \to [0,1] \subseteq \mathbb{R}$ with $\sum_{x\in X} \mu(x) = \mu(X) = 1$. The set of all distributions on $X$ is $\mathit{Distr}(X)$.

**Definition 1. (MDP)** *A* Markov decision process (MDP) $\mathcal{M} = (S, s_I, A, \mathcal{P})$ *consists of a finite set of states $S$, a unique initial state $s_I \in S$, a finite set $A$ of actions, and a probabilistic transition function $\mathcal{P}\colon S \times A \times S \to [0,1] \subseteq \mathbb{R}$ with $\sum_{s'\in S} \mathcal{P}(s,\alpha,s') = 1$ for all $s \in S, \alpha \in A$.*

The *enabled* actions at state $s \in S$ are $A(s) = \{\alpha \in A \mid \exists s' \in S.\, \mathcal{P}(s,\alpha,s') > 0\}$. A *cost function* $\mathcal{C}\colon S \to \mathbb{R}_{\geq 0}$ for an MDP $\mathcal{M}$ adds cost to a *state*. If $|A(s)| = 1$ for all $s \in S$, all actions can be disregarded and the MDP $\mathcal{M}$ reduces to a *discrete-time Markov chain (MC)*, also denoted by $\mathcal{D}$. To define a probability measure and expected cost on MDPs, the nondeterministic choices of actions are resolved by *strategies*. We restrict ourselves to *memoryless* strategies, see [36] for details.

**Definition 2. (Strategy)** *A* randomized strategy[5] *for an MDP $\mathcal{M}$ is a function $\sigma\colon S \to \mathit{Distr}(A)$ such that $\sigma(s)(a) > 0$ implies $a \in A(s)$. A strategy with $\sigma(s)(a) = 1$ for $a \in A$ and $\sigma(b) = 0$ for all $b \in A \setminus \{a\}$ is called* deterministic. *The set of all strategies over $\mathcal{M}$ is denoted by $\mathit{Str}^{\mathcal{M}}$.*

Applying strategy $\sigma \in \mathit{Str}^{\mathcal{M}}$ to MDP $\mathcal{M}$ yields an *induced Markov chain* $\mathcal{M}^\sigma$.

**Definition 3. (Induced MC)** *Let MDP $\mathcal{M} = (S, s_I, A, \mathcal{P})$ and strategy $\sigma \in \mathit{Str}^{\mathcal{M}}$. The* MC induced by $\mathcal{M}$ and $\sigma$ *is $\mathcal{M}^\sigma = (S, s_I, \mathcal{P}^\sigma)$ where*

$$\mathcal{P}^\sigma(s,s') = \sum_{a\in A(s)} \sigma(s)(a) \cdot \mathcal{P}(s,a,s') \quad \text{for all } s,s' \in S\ .$$

[5] If needed, we extend the state space of the original MDP to account for memory.

*PRISM's Guarded Command Language.* We briefly introduce the probabilistic programming language used to specify probabilistic models in PRISM. For a finite set Var of integer variables, let $\mathcal{V}(\text{Var})$ denote the set of all variable valuations.

**Definition 4 (Probabilistic program).** *A* probabilistic program $(\text{Var}, s_I, \mathfrak{M})$ *consists of* Var*, an initial variable valuation* $s_I \in \mathcal{V}(\text{Var})$*, and a finite set of modules* $\mathfrak{M} = \{M_1, \ldots, M_k\}$*. A module* $M_i = (\text{Var}_i, A_i, C_i)$ *consists of* $\text{Var}_i \subseteq \text{Var}$ *such that* $\text{Var}_i \cap \text{Var}_j = \emptyset$ *for* $i \neq j$*, a finite set* $A_i$ *of (synchronizing) actions, and a finite set* $C_i$ *of commands.*

*A command has the form* $[\alpha]\ g \rightarrow p_1 : f_1 + \ldots + p_n : f_n$ *with* $\alpha \in A_i$*,* $g$ *a Boolean guard over the variables in* Var*,* $p_j \in [0,1] \subseteq \mathbb{R}$ *with* $\sum_{j=1}^{n} p_j = 1$*, and* $f_j : \mathcal{V}(\text{Var}) \rightarrow \mathcal{V}(\text{Var}_i)$ *a variable update function.*

A model with several modules is equivalent to a single module, obtained by the *parallel composition* using synchronizing actions. For details we refer to [23].

*Specifications.* For threshold $\lambda \in [0,1]$ and MC $\mathcal{D}$, a *reachability specification* $\varphi = \mathbb{P}_{\leq\lambda}(\Diamond T)$ asserts that a set of target states $T \subseteq S$ is to be reached with probability at most $\lambda$. The expected cost of reaching a set of goal states $G$ is denoted by $\text{EC}^{\mathcal{D}}(\Diamond G)$. Using recent results from [32], we also consider the probability $\Pr^{\mathcal{D}}(\Diamond T \wedge C < n)$, where the total cost $C$, i. e., the sum of the costs of all paths satisfying $\Diamond T$, is bounded by $n$. For MDPs, one needs to compute minimizing/maximizing strategies. Formal definitions are given in e.g., [36].

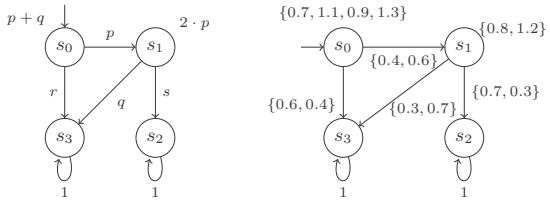
## 4 Structured Synthesis

We first introduce *multiple–instance Markov decision process* (MIMDP) over a finite set $V = \{p_1, \ldots, p_n\}$ of parameters. Each parameter $p \in V$ has a finite range of values $\mathit{Val}(p) = \{v_1, \ldots, v_m\} \subseteq \mathbb{R}$. A *valuation* is a function $u : V \rightarrow \bigcup_{p \in V} \mathit{Val}(p)$ that respects the parameter ranges, meaning that for a parameter $p$, $u(p) = v \in \mathit{Val}(p)$. Let $U(V)$ denote the (finite) set of valuations on $V$.

Let $\mathit{Expr}(V)$ denote the set of *expressions* over $V$ and $p \in l$ state that parameter $p$ occurs in expression $l \in \mathit{Expr}(V)$. $\mathit{Val}(l)$ denotes the (finite) set of possible values for $l \in \mathit{Expr}(V)$ according to the parameters $p \in l$ and their value ranges $\mathit{Val}(p)$. With a slight abuse of notation, we lift valuation functions from $U(V)$ to expressions: $u : \mathit{Expr}(V) \rightarrow \bigcup_{l \in \mathit{Expr}(V)} \mathit{Val}(l)$. In particular, $u(l) = v \in \mathit{Val}(l)$ is the valuation of $l$ obtained by the instantiation of each $p \in l$ with $u(p)$.

*Remark 1.* A valuation of two expressions $l, l'$ in $\mathit{Expr}(V)$ does not guarantee consistent parameter valuations. A parameter valuation $u(p)$ that results in expression valuation $u(l)$ might be different than a $u'(p)$ that results in $u(l')$.

*Example 1.* For $V = \{p, q\}$, $\mathit{Val}(p) = \{0.1, 0.2\}$, $\mathit{Val}(q) = \{0.3, 0.4\}$, and $\mathit{Expr}(V) = \{p + q, p + 2 \cdot q\}$, the ranges of values for the expressions are $\mathit{Val}(p + q) = \{0.4, 0.5, 0.6\}$, and $\mathit{Val}(p + 2 \cdot q) = \{0.7, 0.8, 0.9, 1\}$. For parameter valuation $u(p) = 0.1, u(q) = 0.3$, the associated valuation on expressions is $u(p + q) =$




(a) MIMDP $\mathcal{M}$ (b) Values for $\mathcal{M}$

**Fig. 3.** An example MIMDP and its possible valuations.

$0.4, u(p + 2 \cdot q) = 0.7$. For a valuation on the expressions $v(p + q) = 0.4$, $v(p + 2 \cdot q) = 1$, there is no consistent parameter valuation: the first is consistent with $u(p) = 0.1$ and $u(q) = 0.3$, the second with $u(p) = 0.2$ and $u(q) = 0.4$.

**Definition 5. (MIMDP)** *A* multiple-instance MDP $\mathcal{M} = (S, V, s_I, A, \mathcal{P})$ *has a finite set $S$ of states, a finite set $V$ of parameters with associated finite sets of valuations from $Val(V)$, a unique initial state $s_I \in S$, a finite set $A$ of actions, and a transition function $\mathcal{P}\colon S \times A \times S \to Expr(V)$.*

A *cost function* $\mathcal{C}\colon S \to Expr(V)$ associates (parametric) cost to states. For each valuation $u \in Val(V)$ of parameters, the *instantiated MIMDP* is $\mathcal{M}[u]$. We denote the set of all expressions occurring in the MIMDP by $\mathcal{L}_{\mathcal{M}}$.

*Remark 2.* An MIMDP is a special kind of parametric MDP (pMDP) [11, 28] in the sense that each parametric cost and probabilities can only take a finite number of values, i. e., there are multiple but finite instantiations of a MIMDP. The state–of–the–art tools such as `PARAM` [11], `PRISM` [12], or `PROPhESY` [21], however, only allow for defining continuous intervals to restrict parameter valuations.

*Example 2.* Fig. 3(a) shows a MIMDP $\mathcal{M}$ with parametric transition probabilities $p, q, r$ and $s$. Costs are $p + q$ and $2 \cdot p$. The valuations of parameters are $Val(p) = \{0.4, 0.6\}$, $Val(q) = \{0.3, 0.7\}$, $Val(s) = \{0.7, 0.3\}$, $Val(r) = \{0.6, 0.4\}$, $Val(p + q) = \{0.7, 1.1, 0.9, 1.3\}$, and $Val(2 \cdot p) = \{0.8, 1.2\}$. The sets of valuations are depicted in Fig. 3(b) to show all instantiations of the MIMDP. E.g., a valuation $u$ with $u(p) = 0.4$ and $u(r) = 0.4$ is *not well–defined* as it induces no probability distribution, thereby not yielding an MDP.

*Formal problem statement.* For a MIMDP $\mathcal{M}$, a specification $\varphi = \mathbb{P}_{\leq \lambda}(\Diamond T)$, and a set of goal states $G$, the structured synthesis problem is to determine a valuation $u \in U(V)$ and a strategy $\sigma$ for the MDP $\mathcal{M}[u]$ such that $\mathcal{M}[u]^\sigma \models \varphi$, and the expected cost $\mathrm{EC}^{\mathcal{M}[u]^\sigma}(\Diamond G)$ is minimal.

## 5 An Integer Programming Approach

We first observe that the synthesis problem is in fact a multi-objective verification problem that requires randomized strategies as in Def. 2 [13, 10, 16]. We formulate the corresponding optimization problem using the following variables:

– $c_s \in \mathbb{R}_{\geq 0}$ for each $s \in S$ represents the expected cost to reach $G \subseteq S$ from $s$.
– $p_s \in [0,1]$ for each $s \in S$ represents the probability to reach $T \subseteq S$ from $s$.
– $\sigma_s^\alpha \in [0,1]$ for each $s \in S$ and $\alpha \in A$ represents the probability to choose action $\alpha \in A(s)$ at state $s$.

We also introduce a characteristic variable $x_u \in \{0,1\}$ for each valuation $u \in U(V)$. If $x_u$ is set to 1, all parameters and expressions are evaluated using $u$.

$$\text{minimize } c_{s_I} \tag{2}$$
$$\text{subject to}$$
$$p_{s_I} \leq \lambda \tag{3}$$
$$\forall s \in T.\ p_s = 1 \tag{4}$$
$$\forall s \in G.\ c_s = 0 \tag{5}$$
$$\forall s \in S.\ \sum_{\alpha \in A} \sigma_s^\alpha = 1 \tag{6}$$
$$\sum_{u \in U(V)} x_u = 1 \tag{7}$$
$$\forall s \in S.\ p_s = \sum_{\alpha \in A(s)} \sigma_s^\alpha \cdot \Big( \sum_{s' \in S} \sum_{u \in U(V)} x_u \cdot u(\mathcal{P}(s,\alpha,s')) \cdot p_{s'} \Big) \tag{8}$$
$$\forall s \in S.\ c_s = \sum_{\alpha \in A(s)} \sigma_s^\alpha \cdot \Big( \sum_{s' \in S} \sum_{u \in U(V)} x_u \cdot (u(\mathcal{C}(s)) + u(\mathcal{P}(s,\alpha,s')) \cdot c_{s'}) \Big) \tag{9}$$
$$\forall s \in S, \alpha \in A(s).\ \sum_{s' \in S} \sum_{u \in U(V)} x_u \cdot u(\mathcal{P}(s,\alpha,s')) = 1 \tag{10}$$

**Theorem 1 (Soundness and completeness).** *The optimization problem* (2) – (10) *is* sound *in the sense that each minimizing assignment induces a solution to the synthesis problem. It is* complete *in the sense that for each solution to the problem there is a minimizing assignment for* (2) – (10).

*Proof Sketch.* The first two equations induce satisfaction of the specifications: (2) minimizes the expected cost to reach goal states $G \subseteq S$ at $s_I$; (3) ensures that the probability to reach the target states $T \subseteq S$ from $s_I$ is not higher than the threshold. (4) and (5) set the probability and the expected cost at target and goal states to 1 and 0, respectively. (6) ensures well-defined strategies, and (7) ensures for all possibles values $u \in U(V)$ that exactly one characteristic variable $x_u$ is set to 1. In (8), $p_s$ is assigned the probability of reaching $T$ from $s$ by multiplying the probability to reach successor $s'$ with the probability of reaching $T$ from $s'$, depending on the scheduler variables $\sigma_s^\alpha$. Variables $c_s$ are analogously assigned the expected cost in (9). (10) ensures that the concrete instantiations chosen at each transition form well–defined probability distributions.

Any satisfying assignment yields a well-defined randomized strategy and a well-defined assignment of parameters. Moreover, such an assignment necessarily

satisfies the safety specification $\varphi = \mathbb{P}_{\leq\lambda}(\lozenge T)$, as the probability to reach $T$ is ensured to be smaller than or equal to $\lambda$. Likewise, the expected cost to reach $G$ from the initial state is minimized while at each state the $c_s$ variables are assigned the exact expected cost. We need to assume that the probability to reach $G$ is one under all strategies. If this assumption is not true, additional constraints can enforce that property for each solution of the optimization problem. Thus, a satisfying assignment induces a solution to the synthesis problem. *Completeness* is given by construction, as the optimization problem encodes each instantiation of the problem.

*Complexity of the Optimization Problem.* Consider constraint (8), where an integer variable $x_v$ is multiplied with the real–valued variable $p_{s'}$ and the strategy variable $\sigma_s^\alpha$. Such constraints render this program a *non–linear integer optimization problem*. The number of constraints is governed by the number of state and action pairs and the number of possible instantiations of expressions i. e., the *size of the problem* is in $\mathcal{O}(|S_r|\cdot|A|\cdot|\mathit{Val}(\mathcal{L}_\mathcal{M})|^2)$. The problem is, that already solving nonlinear problems without integer variables is NP-hard [37, 6]. Summarized, despite the compact problem representation in form of a MIMDP, the problem is hard.

## 6 Transformation of `PRISM` Programs

As a feasible—yet not optimal—solution to the synthesis problem, we present a transformation of MIMDPs specified as probabilistic programs in the `PRISM` language as in Def. 4. Similar to [28], we see the possible choices of parameter values as nondeterminism. Say, a parameter $p \in V$ has valuations $\mathit{Val}(p) = \{v_1, v_2\}$ and state $s$ has cost $\mathcal{C}(s) = 2 \cdot p$. First, the MIMDP is transformed in the following way. From state $s$, a nondeterministic choice between actions $\alpha_{v_1}$ and $\alpha_{v_1}$ replaces the original transitions. Each action leads with probability one to a fresh state having cost $2 \cdot v_1$ or $2 \cdot v_2$, respectively. From these states, the original transitions of state $s$ emanate. Minimal or maximal expected cost in this transformed MDP correspond to upper and lower bounds to the optimal solution of the synthesis problem. Intuitively, we *relax dependencies between parameters*. That is, if at one place $p$ is assigned its value $v_1$, it is not necessarily assigned the same everywhere in the MIMDP, leading to inconsistent valuations. To tighten these bounds, a further program transformation ensures parameter dependencies for each execution of the model. Intuitively, in the resulting MDP each nondeterministic choice corresponding to a parameter value leads to a (sub-)MDP where the assignment of that value is fixed.

*Remark 3 (Nondeterminism and continuous parameters).* If the original MIMDP has nondeterministic choices, we introduce a new level of nondeterminism. We then assume that both types of nondeterminism minimize the expected cost for our problem. Alternatively, one can generate and evaluate a stochastic game [15].

If the original problem has continuous parameters in addition, we gain a parametric MIMDP. In that case, mature tools for pMDPS like `PARAM` [11] or `PROPhESY` [21] may be employed following the program transformation.

*Program Transformation 1—Parametric Cost.* Intuitively, for each state satisfying a certain guard, transitions with the same guard are added. Each of the transitions leads to new states with an instantiated rewards. From these states, the transitions of the original system emanate. Assume a `PRISM` program $M = (\text{Var}, A, C)$ as in Def. 4, and a parametric reward structure of the form:

---

**rewards**
$g_1 : l$
**end rewards**

---

with $g_1$ the guard and $l \in \mathit{Expr}(\text{Var})$. Let $\mathit{Val}(l) = \{\overline{v}_1, \ldots, \overline{v}_m\}$ be the finite set of instantiations of $l$ with $\overline{v}_i \in \mathbb{R}$ for $1 \leq i \leq m$. We introduce a fresh (characteristic) variable $x_l$ with $\text{dom}(x_l) = \{0, \ldots, m\}$. Intuitively, there is a unique variable value for $x_l$ for each valuation from $\mathit{Val}(l)$. Consider now all commands $c \in C$ of the form

---

$[\alpha]\ g \to p_1 : f_1 + \ldots + p_n : f_n;$

---

with $g \models g_1$, i. e., the guard of the command satisfies the guard of the reward structure. Replace each such commands $c$ by the following set of commands:

---

$[]\ g \to 1 : x_l' = 1;$
$\vdots$
$[]\ g \to 1 : x_l' = m;$
$[\alpha] \bigvee_{1 \leq i \leq m} x_l = i \to p_1 : f_1 + \ldots + p_n : f_n;$

---

and replace the reward structure for each command $c$ by

---

**rewards**
$x_l = 1 : v_1;$
$\vdots$
$x_l = m : v_m;$
**end rewards**

---

This transformation corresponds to a nondeterministic choice between the concrete reward values.

*Program Transformation 2—Parametric Transitions.* For all parameters values, transitions with concrete probabilities are introduced. As all transitions satisfy the same guard, we have again a nondeterministic choice between these transitions. For program $M = (\text{Var}, A, C)$, consider a command $c \in C$ of the form

---

$[\alpha]\ g \to p_1 : f_1 + \ldots + p_n : f_n$

---

with $p_1, \ldots, p_n \in \mathit{Expr}(\mathrm{Var})$. Let $\mathit{Val}(p_1, \ldots, p_n) = \{v_1^n, \ldots, v_m^n\}$ with $v_i^n = (v_{i1}, \ldots, v_{in})$ for $1 \leq i \leq n$ and $v_{ij} \in \mathbb{R}$ for $1 \leq j \leq m$. Replace each such command $c$ by the following set of commands:

---

$[\alpha]\ g \to v_{11}\colon f_1 + \ldots + v_{1n}\colon f_n;$
$\vdots$
$[\alpha]\ g \to v_{m1}\colon f_1 + \ldots + v_{mn}\colon f_n;$

---

For a program $M$, we denote the program after Transformation 1 and Transformation 2 by $M'$. The induced MIMDP of $M$ is denoted by $\mathcal{M}_M$ and the induced MDP of $M'$ by $\mathcal{M}_{M'}$.

*Program Transformation 3—Parameter Dependencies.* We finally propose a transformation of the transformed MDP $\mathcal{M}_{M'}$ which enforces that once a parameter is assigned a specific value, this assignment is always used throughout a system execution. Therefore, we add a *control module* to the PRISM formulation.

In the transformed MDP, taking actions $\alpha_{v_1}$ or $\alpha_{v_2}$ induces that parameter $p$ is assigned its value $v_1$ or $v_2$. In the PRISM encoding, the corresponding commands are of the form

---

$[\alpha_{v_1}]\ g_1 \to \ldots;$
$[\alpha_{v_2}]\ g_2 \to \ldots;$

---

where $g_1$ and $g_2$ are arbitrary guards. Now, for each of these actions $\alpha_{v_1}$ and $\alpha_{v_2}$, we use control variables $q_{v_1}$ and $q_{v_2}$ and build a control module of the form:

---

**module** *control*
  $q_{v_1}$: **bool init** 0;
  $q_{v_2}$: **bool init** 0;
  $[\alpha_{v_1}]\ \neg q_{v_1} \to (q'_{v_1} = \texttt{true});$
  $[\alpha_{v_1}]\ q_{v_1} \to (q'_{v_1} = \texttt{true});$
  $[\alpha_{v_2}]\ \neg q_{v_2} \to (q'_{v_2} = \texttt{true});$
  $[\alpha_{v_2}]\ q_{v_2} \to (q'_{v_2} = \texttt{true});$
**endmodule**

---

If this module is included in the parallel composition, a control variable is set to `true` once the corresponding action is taken in the MDP. We can now guard the commands such that only non–conflicting assignments are possible. The original commands are transformed in the following way:

---

$[\alpha_{v_1}]\ g_1 \wedge \neg q_{v_2} \to \ldots;$
$[\alpha_{v_2}]\ g_2 \wedge \neg q_{v_1} \to \ldots;$

---

With the control module, consistent choices are enforced for any execution of the system, while there may be inconsistencies in parameter valuations at states or transitions that are not visited by the same run of the system. In these (rare) cases, the resulting strategy does not offer any guarantees. Specifically, an optimal strategy does not necessarily induce consistent parameter assignments, which is an NP-hard problem and handled in [34].

Technically, we transform an integer nonlinear optimization problem to a linear program at the cost of increasing the size of the underlying MDP. We exploit highly optimized model checking tools. Aggressive state space reduction techniques together with a preprocessing that removes inconsistent combinations of parameter values beforehand render the MIMDP synthesis problem feasible.

## 7 Experiments

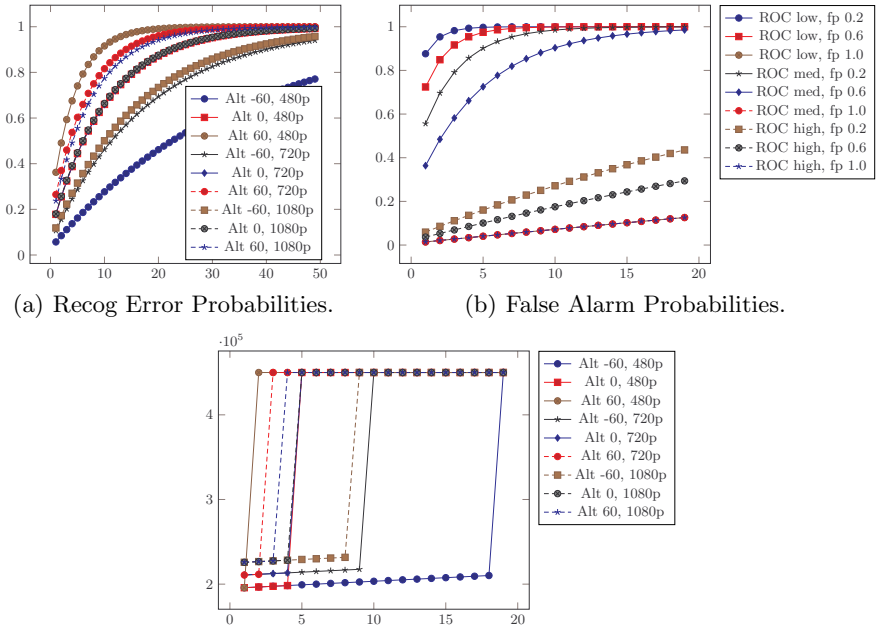
We first report on results for the case study from Sec. 2. We created a `PRISM` program for the MIMDP underlying all aspects of the shipyard example with 430 lines of code. We use the program to generate an explicit Markov chain (MC) model where the parameter instantiations are fixed. That explicit model has 1 728 states and 5 145 transitions. From the MIMDP model, we generate the MDP according to the transformations in Sec. 6. The underlying (extended) `PRISM` program has 720 lines of code. The explicit MDP generated from the transformed program has 2 912 states and 64 048 transitions. For our case study, the size of the transformed MDP is reasonable: From the MIMDP to the MDP, states increase by a factor of 1.8, transitions by a factor of 12.5. We performed all experiments on a MacBook Pro with a 2.3 GHz Intel Core i5 CPU and 8GB of RAM with the standard configuration of the `Storm` model checker.

*Results Case Study.* The experiments show several (partially unforeseeable) intricacies of the case study. We have the following structure for the MIMDP defined by parameters and their valuation sets. For details see Sec. 2.

- EO sensor for the UAV: $V_{\mathrm{EO}} = \{480p, 720p, 1080p\}$.
- Deviation from the UAV operational altitude: $V_{\mathrm{Alt}} = \{-60, -30, 0, 30, 60\}$.
- ROC ground sensors (Sensor 1, Sensor 2, Sensor 3, Bay Area Network): $V_{\mathrm{ROC}} = \{\mathrm{low}, \mathrm{med}, \mathrm{high}\}$.
- False positive rates for each ROC ground sensor: $V_{\mathrm{fp}} = \{0.2, 0.3, \ldots, 1.0\}$.

This structure induces 1 440 possible system configurations. However, as we restrict ground sensors to have the same quality, we have only 360 possibilities. We implemented benchmark scripts using the `Python` interface of `Storm`. For the results presented below, we iteratively try all possible combinations for measures of interest, and compare the time to compute an optimal value obtained from the transformed MDP to these cumulated model checking times.

*Probability of Recognition Error.* First, we investigate the probability of not recognizing an intruder—after a ground sensor was triggered—in dependence of the number of missions an UAV flies. The curves shown in Fig. 4(a) depend on the deviation from the standard UAV operational altitude and on the type of EO sensor. The cheap 480p EO sensor has the lowest probability for a recognition error at a low altitude. For all other sensors and altitudes, the probability quickly approaches one. The cumulated model checking time was 78.13 seconds, computing the optimal result on the transformed MDP took 2.3 seconds.




(a) Recog Error Probabilities.

(b) False Alarm Probabilities.

(c) Probability and Cost Tradeoffs.

**Fig. 4.** Safety and Performance Measures per Number of Missions.

*Probability of False Alarms.* For the ROC sensors, we measured the probability for a false alarm, depending again on the number of missions. The curves in Fig. 4(b) depend on the quality of the ROC sensor and on the false positive rates. The high-quality sensor is the only one with relatively low false alarm probabilities. The cumulated model checking time was 52.13 seconds, computing the optimal result on the transformed MDP took less than one second.

*Probability and Cost Tradeoffs.* Finally, we show the expected cost in dependence of the number of missions in Fig. 4(c). Additionally, for each data point, the probability for a recognition error needs to be below 50% . The violation of this property is indicated by the maximum value $4.5 \cdot 10^{-5}$. The results show that for this kind of property indeed the low-resolution (480p) sensor at lowest altitude is the best choice, as it has relatively low initial cost. While the task cost at low altitude is slightly larger than at higher altitudes, with this sensor, the UAV is able to maintain the probability threshold of safely recognizing an intruder. The cumulated model checking time was 59.1 seconds, computing the optimal result on the transformed MDP took 1.2 seconds.

*Further Benchmarks.* We additionally assessed our approach on well-known parametric Markov chain examples from the `PARAM`-website [22], that originally stem from the `PRISM` benchmark suite [14]. We tested a parametric version of the Knuth-Yao Die (Die), several instances of the Zeroconf protocol [7], and the

**Table 1.** Parametric Benchmarks

| Model | Type | States | Transitions | MC (s) | SE (s) |
|---|---|---|---|---|---|
| Die | parametric | 13 | 20 | — | 0.04 |
| | transformed | 13 | 48 | 0.04 | — |
| | controlled | 37 | 60 | 0.02 | — |
| Zeroconf1 | parametric | 1 004 | 2 005 | — | 60.3 |
| | transformed | 1 004 | 6 009 | 0.18 | — |
| | controlled | 9 046 | 18 075 | 0.19 | — |
| Zeroconf2 | parametric | 100 004 | 200 005 | — | TO |
| | transformed | 100 004 | 600 009 | 0.90 | — |
| | controlled | 900 046 | 1 800 075 | 7.36 | — |
| Crowds1 | parametric | 7 421 | 12 881 | — | 5.80 |
| | transformed | 7 421 | 20 161 | 0.08 | — |
| | controlled | 66 826 | 116 148 | 0.42 | — |
| Crowds2 | parametric | 572 153 | 1 698 233 | — | 18.81 |
| | transformed | 572 153 | 2 261 273 | 4.39 | — |
| | controlled | 5 149 474 | 15 284 736 | 33.70 | — |
| Crowds3 | parametric | 2 018 094 | 7 224 834 | — | TO |
| | transformed | 2 018 094 | 9 208 354 | 17.15 | — |
| | controlled | 18 162 973 | 65 024 355 | 137.71 | — |

Crowds protocol [5]. For all benchmarks, we introduced three discrete values as domain for each parameter. Table 1 shows the results, where we list the number of states and transitions as well as the model checking times (MC): "transformed" refers to the MDP after Transformation 1 and 2, "controlled" refers to the MDP after Transformations 1 – 3. For the original parametric Markov chains, we tested the time to perform state elimination (SE), which is the standard model checking method for such models [11, 8]. We used a timeout (TO) of 600 seconds.

We draw the following conclusions: (1) The first two program transformations only increase the number of transitions with respect to the MIMDP model, the third transformation increases the states and transitions by up to one order of magnitude. (2) Except for the Crowds2 instance, model checking the transformed and controlled MDP is superior to performing state elimination. (3) For these benchmarks, we are able to handle instances with millions of states.

## 8 Conclusion and Future Work

We introduced structured synthesis for MDPs. Driven by a concrete case study from the area of physical security, we defined MIMDPs and demonstrated the hardness of the corresponding synthesis problem. As a feasible solution, we presented a transformation to an MDP where nondeterministic choices represented the underlying structure of the MIMDP. Our experiments showed that we are able to analyze meaningful measures for such problems. In the future, we will investigate further intricacies of the case study regarding continuous state spaces. Moreover, we will extend our approaches to account for so-called high-level counterexamples, that provide insight on errors in the system on the level of the `PRISM` language [18, 25].

*Acknowledgements.* We want to thank Sebastian Junges for providing us with valuable insights on the correctness of our approaches.

## References


1. J. Johnson. Analysis of image forming systems. In *Image Intensifer Symposium*, pages 249 – 273, 1958.
2. P. M. Moser. Mathematical model of FLIR performance. Technical Report Technical Memorandum NADC-20203, Naval Air Development Center, Warminster, PA, 1972.
3. Jay K. Satia and Roy E. Lave Jr. Markovian decision processes with uncertain transition probabilities. *Operations Research*, 21(3):728–740, 1973.
4. R. Driggers, M. Kelley, and Paul Cox. National imagery interpretation rating system and the probabilities of detection, recognition, and identification. *Optical Engineering*, 36(7):1952–1959, 1997.
5. Michael K. Reiter and Aviel D. Rubin. Crowds: Anonymity for web transactions. *ACM Trans. on Information and System Security*, 1(1):66–92, 1998.
6. Jean B. Lasserre. Global optimization with polynomials and the problem of moments. *SIAM Journal on Optimization*, 11(3):796–817, 2001.
7. Henrik Bohnenkamp, Peter Van Der Stok, Holger Hermanns, and Frits Vaandrager. Cost-optimization of the IPv4 zeroconf protocol. In *DSN*, pages 531–540. IEEE CS, 2003.
8. Conrado Daws. Symbolic and parametric model checking of discrete-time Markov chains. In *ICTAC*, volume 3407 of *LNCS*, pages 280–294. Springer, 2004.
9. Tom Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8):861–874, 2006.
10. Kousha Etessami, Marta Z. Kwiatkowska, Moshe Y. Vardi, and Mihalis Yannakakis. Multi-objective model checking of Markov decision processes. *Logical Methods in Computer Science*, 4(4), 2008.
11. Ernst Moritz Hahn, Holger Hermanns, and Lijun Zhang. Probabilistic reachability for parametric Markov models. *Software Tools for Technology Transfer*, 13(1):3–19, 2010.
12. Marta Kwiatkowska, Gethin Norman, and David Parker. PRISM 4.0: Verification of probabilistic real-time systems. In *CAV*, volume 6806 of *LNCS*, pages 585–591. Springer, 2011.
13. Vojtech Forejt, Marta Z. Kwiatkowska, and David Parker. Pareto curves for probabilistic model checking. In *ATVA*, volume 7561 of *LNCS*, pages 317–332. Springer, 2012.
14. Marta Kwiatkowska, Gethin Norman, and David Parker. The PRISM benchmark suite. In *QEST*, pages 203–204. IEEE CS, 2012.
15. Taolue Chen, Vojtech Forejt, Marta Z. Kwiatkowska, David Parker, and Aistis Simaitis. Prism-games: A model checker for stochastic multi-player games. In *TACAS*, volume 7795 of *LNCS*, pages 185–191. Springer, 2013.
16. Christel Baier, Clemens Dubslaff, and Sascha Klüppelholz. Trade-off analysis meets probabilistic model checking. In *CSL-LICS*, pages 1:1–1:10. ACM, 2014.
17. Hua Chen, Krishna Kalyanam, Wei Zhang, and David Casbeer. Continuous-time intruder isolation using Unattended Ground Sensors on graphsround sensors on graphs. In *ACC*, 2014.
18. Christian Dehnert, Nils Jansen, Ralf Wimmer, Erika Ábrahám, and Joost-Pieter Katoen. Fast debugging of PRISM models. In *ATVA*, volume 8837 of *Lecture Notes in Computer Science*, pages 146–162. Springer, 2014.
19. Andrew D. Gordon, Thomas A. Henzinger, Aditya V. Nori, and Sriram K. Rajamani. Probabilistic programming. pages 167–181. ACM Press, 2014.

20. Ernst Moritz Hahn, Yi Li, Sven Schewe, Andrea Turrini, and Lijun Zhang. iscasMc: A web-based probabilistic model checker. In *FM*, volume 8442 of *LNCS*, pages 312–317. Springer, 2014.
21. Christian Dehnert, Sebastian Junges, Nils Jansen, Florian Corzilius, Matthias Volk, Harold Bruintjes, Joost-Pieter Katoen, and Erika Ábrahám. Prophesy: A probabilistic parameter synthesis tool. In *CAV (1)*, volume 9206 of *LNCS*, pages 214–231. Springer, 2015.
22. `PARAM` Website, 2015. `http://depend.cs.uni-sb.de/tools/param/`.
23. `PRISM` Website, 2015. `http://prismmodelchecker.org`.
24. Steven Rasmussen and Derek Kingston. Development and flight test of an area monitoring system using Unmanned Aerial Vehicles and Unattended Ground Sensors. In *International Conference on Unmanned Aircraft Systems*, pages 1215–1224, 2015.
25. Ralf Wimmer, Nils Jansen, Andreas Vorpahl, Erika Ábrahám, Joost-Pieter Katoen, and Bernd Becker. High-level counterexamples for probabilistic automata. *Logical Methods in Computer Science*, 11(1), 2015.
26. Karina Valdivia Delgado, Leliane N. de Barros, Daniel B. Dias, and Scott Sanner. Real-time dynamic programming for markov decision processes with imprecise probabilities. *Artif. Intell.*, 230:192–223, 2016.
27. Derek Kingston, Steven Rasmussen, and Laura Humphrey. Automated UAV tasks for search and surveillance. In *CCA*, pages 1–8. IEEE, 2016.
28. Tim Quatmann, Christian Dehnert, Nils Jansen, Sebastian Junges, and Joost-Pieter Katoen. Parameter synthesis for Markov models: Faster than ever. In *ATVA*, volume 9938 of *LNCS*, pages 50–67, 2016.
29. Murat Cubuktepe, Nils Jansen, Sebastian Junges, Joost-Pieter Katoen, Ivan Papusha, Hasan A. Poonawala, and Ufuk Topcu. Sequential convex programming for the efficient verification of parametric MDPs. In *TACAS (2)*, volume 10206 of *Lecture Notes in Computer Science*, pages 133–150, 2017.
30. Christian Dehnert, Sebastian Junges, Joost-Pieter Katoen, and Matthias Volk. A storm is coming: A modern probabilistic model checker. In *CAV*, volume 10427 of *LNCS*, pages 592–600. Springer, 2017.
31. Philipp Chrszon, Clemens Dubslaff, Sascha Klüppelholz, and Christel Baier. Profeat: feature-oriented engineering for family-based probabilistic model checking. *Formal Asp. Comput.*, 30(1):45–75, 2018.
32. Arnd Hartmanns, Sebastian Junges, Joost-Pieter Katoen, and Tim Quatmann. Multi-cost bounded reachability in MDPs. In *TACAS*, LNCS, April 2018.
33. Nils Jansen, Laura R. Humphrey, Jana Tumova, and Ufuk Topcu. Structured synthesis for probabilistic systems. *CoRR*, abs/1807.06106, 2018.
34. Milan Ceska, Nils Jansen, Sebastian Junges, and Joost-Pieter Katoen. Shepherding hordes of Markov chains. *CoRR*, abs/1902.05727, 2019.
35. Sebastian Junges, Erika Abraham, Christian Hensel, Nils Jansen, Joost-Pieter Katoen, Tim Quatmann, and Matthias Volk. Parameter synthesis for Markov models. *arXiv preprint arXiv:1903.07993*, 2019.
36. Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. The MIT Press, 2008.
37. Dimitri P. Bertsekas. *Nonlinear Programming*. Athena Scientific Belmont, 1999.
38. N. S. Kopeika. *A System Engineering Approach to Imaging*. SPIE Press, 1998.

# A Case Study–Details

## A.1 UAV Task Distances

**Table 2.** Task distances in meters for point search tasks.

| **Task** | $d_g$ | $d'_g$ |
|---|---|---|
| Main Gate | 1000 | 1000 |
| Power Generator | 1100 | 1100 |
| Sensor 1 | 3250 | 3250 |
| Sensor 2 | 3375 | 3375 |
| Sensor 3 | 2375 | 2375 |
| Bay Sensor Network | 3750 | 3750 |

For a point search task $t_p \in \{$Main Gate, Power Generator, Sensor 1, Sensor 2, Sensor 3, Bay Sensor Network$\}$, let $d_g(t_p)$ be the distance from the Ground Control Station to the task location and $d'_g(t_p)$ be the distance back, as given in Table 2. Then the total distance $d(t_p)$ traveled to perform task $t_p$ is

$$d(t_p) = d_g(t_p) + d'_g(t_p). \tag{11}$$

In this case distances are symmetric, with $d_g(t_p) = d'_g(t_p)$ for all $t_p$.

**Table 3.** Task distances in meters for line search tasks.

| **Task** | $d_g$ | $d'_g$ | $d_l$ |
|---|---|---|---|
| Highway | 2000 | 2000 | 2875 |
| Runway | 1000 | 1250 | 2000 |
| Stream | 3250 | 3250 | 2050 |

For a line search task $t_l \in \{$Highway, Runway, Stream$\}$, let $d_g(t_l)$ be the distance from the Ground Control Station to the start of the line, $d'_g(t_l)$ the distance back to the Ground Control Station from the end of the line, and $d_l(t_l)$ the length of the line to be searched, as given in Table 3. Then the total distance $d(t_l)$ traveled to perform task $t_l$ is

$$d(t_l) = d_g(t_l) + d_l(t_l) + d'_g(t_l). \tag{12}$$

For an area search task $t_a \in \{$Bridge, Main Office, Warehouse, Truck Depot, Shipyard Office, Shipyard, West Bay, East Bay, Airfield Office, Airfield$\}$, let $d_g(t_a)$ be the distance from the Ground Control Station to the starting corner of the search, $d'_g(t_a)$ be the distance back to the Ground Control Station from the terminal corner of the search, and let $d_w(t_a)$ and $d_h(t_a)$ be the width and height of the area to be searched, as given in Table 3. In order to sweep an EO

**Table 4.** Base task distances in meters for area search tasks.

| **Task** $t_a$ | $d_g(t_a)$ | $d'_g(t_a)$ | $d_s(t_a)$ | $d'_s(t_a)$ | $d_h(t_a)$ | $d_w(t_a)$ |
|---|---|---|---|---|---|---|
| Bridge | 3750 | 3500 | $d_g$(Sensor 1) = 3250 | 500 | 375 | 600 |
| Main Office | 2600 | 1525 | $d_g$(Sensor 1) = 3250 | 625 | 500 | 1100 |
| Warehouse | 2750 | 1250 | $d_g$(Sensor 1) = 3250 | 575 | 700 | 1500 |
| Truck Depot | 2950 | 2100 | $d_g$(Sensor 2) = 3375 | 450 | 625 | 1200 |
| Shipyard Office | 3350 | 3150 | $d_g$(Sensor 2) = 3375 | 425 | 550 | 1000 |
| Shipyard | 3625 | 3050 | $\{d_g$(Sensor 2), $d_g$(Bay Sensor)$\}$ | 500 | 800 | 700 |
| West Bay | 4250 | 4900 | $d_g$(Bay Sensor) = 3750 | 1000 | 575 | 1000 |
| East Bay | 3125 | 3350 | $d_g$(Bay Sensor) = 3750 | 1500 | 575 | 2000 |
| Airfield Office | 2100 | 2000 | $d_g$(Sensor 3) = 2375 | 450 | 550 | 500 |
| Airfield | 1750 | 2875 | $d_g$(Sensor 3) = 2375 | 700 | 950 | 1625 |

sensor footprint across the entire search area, a UAV has to make several parallel passes over the area along its height. Let $\bar{e}_w$ be the width of a UAV's EO sensor footprint at the UAV's standard operating altitude. Then the total distance $d(t_a)$ traveled to perform task $t_a$ is approximately

$$d(t_a) = d_g(t_a) + \frac{d_w(t_a)}{\bar{e}_w} d_h(t_a) + d'_g(t_a). \tag{13}$$

When an area search is prompted by the detection of an intruder by a ground sensor, the distance calculation is modified to account for travel time to the sensor as well as a potentially different sensor footprint width $e_w$. Let the total distance in response to a ground sensor be $d_s(t_a)$. Then

$$d(t_a) = d_s(t_a) + d'_s(t_a) + \frac{d_w(t_a)}{e_w} d_h(t_a) + d'_g(t_a). \tag{14}$$

### A.2 Image GSD

We measure GSD $G$ as the number of pixels across the width of an EO sensor footprint at its center as shown in Fig. 5, resulting in the expression

$$G = \frac{e_w}{r_h} = \frac{2h_{alt} \tan(\eta/2)/\sin(\theta_g)}{r_h}, \tag{15}$$

where $e_w$ is the width of the EO sensor footprint and $r_h$ the horizontal resolution of the EO sensor. Note that $e_w$ depends on UAV altitude $h_{alt}$, EO sensor field of view $\eta$, and EO sensor gimbal angle $\theta_g$.

In the case study, most objects are large enough that GSD will not be a concern. However, intruders on foot near Sensor 1, Sensor 2, or Sensor 3, or intruders on small personal watercraft near the Bay Sensor Network will require a low GSD. For detecting intruders indicated by these ground sensors, we assume a minimum of $\pi/12$ for $\eta$, since smaller values tend to result in a shaky stream of video images due to noise in UAV flight trajectories, and we assume $\theta_g$ is set to

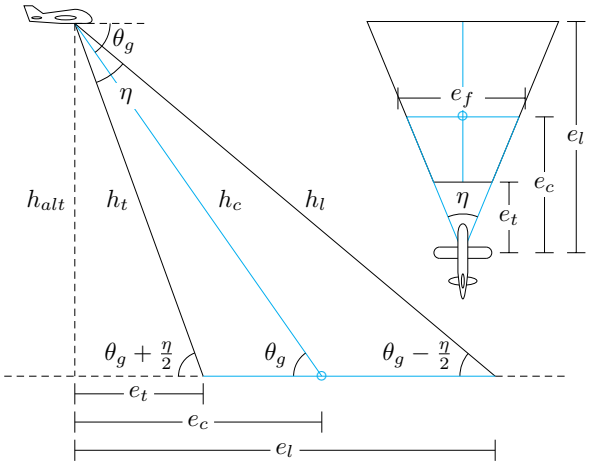



**Fig. 5.** The footprint of an EO sensor, which depends on gimbal elevation angle, field of view, and UAV altitude.

$\pi/2$ so that the camera is pointing straight down. With $\eta = \pi/12$ and $\theta_g = \pi/2$, the expression for $G$ becomes

$$G = \frac{2h_{alt}\tan(\pi/24)}{r_h}. \tag{16}$$

### A.3 Object Recognition versus Altitude, Speed, and Sensor Quality

The UAVs in the case study are essentially tools for carrying out various surveillance tasks in collaboration with human operators. For each surveillance task, a UAV flies over the associated region and sends imagery back to the ground control station, where an operator analyzes it. The level of analysis the operator is able to perform depends on many factors, including the size of objects of interest relative to the resolution of the imagery and how long the operator is able to view objects in the imagery.

Models of human performance for basic visual analysis tasks are often traced to research performed by Johnson in the context of night vision systems [1]. Johnson classified visual analysis tasks into several different categories, including *detection*, *recognition*, and *identification*. Detection is roughly defined as the ability to discern whether an object of significance is present in an image; recognition as determining the class of an object relative to similarly sized objects, e.g., a truck versus a car; and identification as distinguishing between specific members of the same class, e.g., a man versus a woman. Johnson characterized the probability of successfully performing these tasks as a function of the number of line pairs across the critical dimension of an object in an image, where a line pair corresponds to two lines of pixels in digital imagery and critical dimension refers to the smaller dimension of an object. Much of this research was originally performed on targets

with aspect ratios less than 2:1. Later work by Moser found that for objects with large aspect ratios of about 10:1, the number of line pairs or pixels across the area of the object is a better predictor of task performance [2]. As given in [4], Table 5 lists $n_{50}$ values, i.e., numbers of line pairs across an object needed for a 50% probability of successfully performing a visual analysis task on the object. Note that the 1D criterion applies to the critical dimension of an object with a "small" aspect ratio, while the 2D criterion applies to the geometric mean of the height and width of an object with a "large" aspect ratio.

**Table 5.** Johnson chart – the number of line pairs needed for a 50% probability of performing a visual analysis task on an object.

| **Task type** | **1D** $n_{50}$ | **2D** $n_{50}$ | **Description** |
|---|---|---|---|
| Detection | 1.0 | 0.75 | Object is present |
| Recognition | 4.0 | 3.0 | Class of object |
| Identification | 8.0 | 6.0 | Class member of object |

Suppose there are $n$ line pairs across an object in an image. Then given an unlimited amount of time to view the image, the probability of an operator being able to successfully perform a visual analysis task can be estimated as

$$P_\infty = \frac{(n/n_{50})^{x_0}}{1 + (n/n_{50})^{x_0}} \quad \text{where} \quad x_0 = 2.7 + 0.7(n/n_{50}). \tag{17}$$

Further details on (17) can be found in [38].

Suppose UAVs in the case study are equipped with standard electro-optical (EO) sensors. Most EO sensors feature adjustable magnification, which allows the sensor's field of view to vary. An EO sensor may also be mounted on a gimbal, so that it can be pointed in different directions relative to the UAV. As shown in Fig. 5, the size of the EO sensor's footprint on the ground depends on its gimbal elevation angle $\theta_g$, its vertical field of view $\eta_v$, its horizontal field of view $\eta_h$, and the altitude of the UAV $h_{alt}$. Suppose the vertical and horizontal fields of view are equal so that $\eta_h = \eta_v$. Suppose also that the sensor is pointed straight down so that $\theta_g = 90°$. Then the height and width $d_w$ of the sensor footprint will be the same, with

$$d_w = 2h_{alt} \tan(\eta_h/2). \tag{18}$$

The region that lies within the sensor footprint is projected onto a sensor array that converts light into electrical signals. The resolution of the resulting image depends on the resolution of this sensor array. As shown in Table 6, EO sensors with different resolutions can be purchased for different prices, where $r_v$ and $r_h$ are vertical and horizontal resolution in pixels. Given the sensor resolution, current field of view, UAV altitude, and assuming $r_h > r_v$, we can calculate a conservative upper bound on the ground sample distance $gsd$ of the image in

m/pixel as

$$gsd = \frac{d_w}{r_v}. \tag{19}$$

The number of line pairs $n$ across an object of size $d_o$ in the image is then

$$n = \frac{d_o}{2gsd} = \frac{d_o r_v}{2d_w} = \frac{d_o r_v}{4h_{alt}\tan(\eta_h/2)}. \tag{20}$$

**Table 6.** Hypothetical costs for EO sensors with different image resolutions.

| | **Resolution** $r_h \times r_v$ | | |
|---|---|---|---|
| | 640×480 (480p) | 1280×720 (720p) | 1920×1080 (1080p) |
| **Cost** | $15k | $30k | $45k |

### A.4 UAV Altitude

Since the probability that a human operator can analyze an object in an image defined in (1) varies significantly with altitude $h_{alt}$, we define a different operating altitude for each type of EO sensor. Let $r_h^l$, $r_h^m$, and $r_h^h$ be the horizontal resolution of the low (480p), mid (720p), and high (1080p) resolution EO sensor options, respectively. Assume $d_o = 0.5$ m for intruders, both humans on foot and small watercraft. Then let us define operating altitudes $h_0^l$, $h_0^m$, and $h_0^h$ such that the probability of detection $p_d^l$, $p_d^m$, and $p_d^h$ for each EO sensor option is approximately 0.95 at the corresponding operating altitude. In this case, $h_0^l = 296$ m, $h_0^m = 593$ m, and $h_0^h = 889$ m. Suppose we allow altitude to vary by an amount $\Delta h^l, \Delta h^m, \Delta h^h \in [-60, 60]$ m around each operating altitude. Then Fig. 6 shows the probability of detecting an intruder as $\Delta h^l, \Delta h^m$, and $\Delta h^h$ vary for the three EO sensor options around operating altitudes $h_0^l$, $h_0^m$, and $h_0^h$. Linear and quadratic approximations were computed using MATLAB's constrained linear least squares solver *lsqlin* with the constraint that the approximated function have an upper bound of 1. Linear approximations, shown as dashed lines in Fig. 2, are

$$p_d^l = -0.0008\Delta h^l + 0.9461 \quad \text{for } \Delta h^l \in [-60, 60] \tag{21}$$
$$p_d^m = -0.0004\Delta h^m + 0.9498 \quad \text{for } \Delta h^m \in [-60, 60] \tag{22}$$
$$p_d^h = -0.0003\Delta h^h + 0.9505 \quad \text{for } \Delta h^h \in [-60, 60], \tag{23}$$

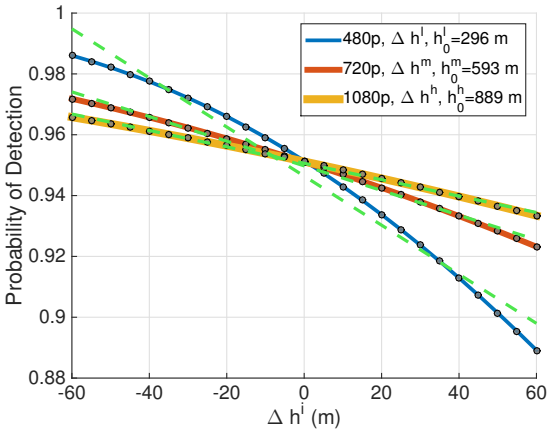



**Fig. 6.** Probability of detecting an intruder for $\Delta h \in [-60, 60]$ for three EO sensor resolution options. Linear approximations are shown as dashed lines, and quadratic approximations are shown as traces of black dots.

and quadratic approximations, plotted with dots in Fig. 2, are

$$p_d^l = -0.000004\Delta h^{l\,2} - 0.000810\Delta h^l + 0.951075 \quad \text{for } \Delta h^l \in [-60, 60] \tag{24}$$

$$p_d^m = -0.000001\Delta h^{m\,2} - 0.0004051\Delta h^m + 0.9511169 \quad \text{for } \Delta h^m \in [-60, 60] \tag{25}$$

$$p_d^h = -0.000000\Delta h^{h\,2} - 0.000300\Delta h^h + 0.950500 \quad \text{for } \Delta h^h \in [-60, 60], \tag{26}$$

with linear and quadratic approximations for $p_d^h$ being equivalent.

The altitude at which a UAV performs an area search will also change the amount of time needed to perform the search. This is because the width of the EO sensor footprint $e_w$ changes with altitude, and in order to sweep the EO sensor footprint over the entire area, a UAV must make approximately $d_w/e_w$ passes of length $d_h$ over the area, where $d_w$ is the task area width and $d_h$ task area height, as given in Table 4. Let us consider how this changes the operational cost of an area search when searching for intruders. Suppose cost per flight hour is approximately 1,000 dollars per hour or $c_f = 5/18$ dollars per second. Suppose a UAV's standard operating altitude $h_0$ for surveillance tasks other than intruder area searches is $h_0 = h_0^h$, its standard ground speed $v_g$ is 15 m/s, and its speed $v_a$ to ascend or descend during altitude changes is 5 m/s. Suppose for intruder area searches, a UAV changes altitude from $h_0$ to $h_0^i + \Delta h^i$ for $\Delta h^i \in [-60, 60]$ m, $i \in \{l, m, h\}$ before a search, then changes back to $h_0$ after a search. Then

the modified cost for an area search task $t_a$ for intruders is approximately

$$c(t_a) = \frac{5}{18}\Big(d_s(t_a) + d'_s(t_a) + \frac{1}{e_w}\frac{d_w(t_a)d_h(t_a)}{v_g} + \frac{2(h_0 - h_0^i - \Delta h^i)}{v_a} + d'_g(t_a)\Big), \tag{27}$$

which replaces the expression for cost given in Section 2. Fig. 7 shows how $1/e_w$

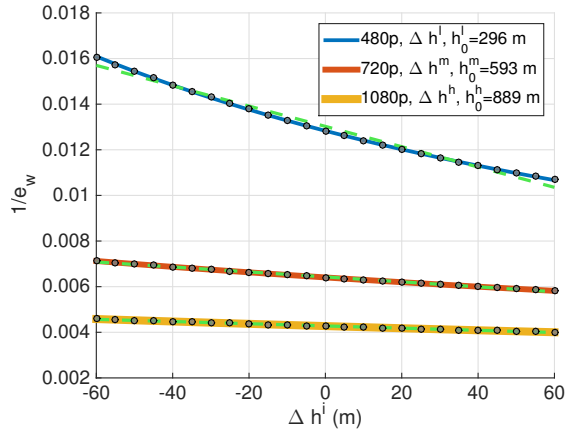



**Fig. 7.** Plots of $1/e_w$ for $\Delta h \in [-60, 60]$ for three sensor resolution options. Linear approximations are shown as dashed lines, and quadratic approximations are shown as traces of black dots.

changes for the low, mid, and high cost EO sensor options given their operating points. In our approach, we obtain the following linear approximations to use in (27), shown as dashed lines in Fig. 7,

$$1/e_w^l = -0.000045\Delta h^l + 0.013026 \quad \text{for } \Delta h^l \in [-60, 60] \tag{28}$$

$$1/e_w^m = -0.000011\Delta h^m + 0.006428 \quad \text{for } \Delta h^m \in [-60, 60] \tag{29}$$

$$1/e_w^h = -0.000005\Delta h^h + 0.004279 \quad \text{for } \Delta h^h \in [-60, 60], \tag{30}$$

and the following quadratic approximations to use in (27), plotted as black dots in Fig. 7,

$$1/e_w^l = 0.0000002{\Delta h^l}^2 - 0.0000445h^l + 0.0128284 \quad \text{for } \Delta h^l \in [-60, 60] \tag{31}$$

$$1/e_w^m = -0.0000000{\Delta h^m}^2 - 0.0000110h^m + 0.0064280 \quad \text{for } \Delta h^m \in [-60, 60] \tag{32}$$

$$1/e_w^h = -0.0000000{\Delta h^h}^2 - 0.0000050h^h + 0.0042790 \quad \text{for } \Delta h^h \in [-60, 60], \tag{33}$$

with $1/e_w^m$ and $1/e_w^h$ each having equivalent linear and quadratic approximations.

### A.5 ROC Curve Approximations

Approximations of the ROC curves were computed using MATLAB's constrained linear least squares solver *lsqlin* with the constraint that the approximated function have an upper bound of 1. Let $p_f^l$, $p_f^m$, and $p_f^h$ be the false positive rate for low, mid, and high cost sensors respectively, and let $p_t^l$, $p_t^m$, and $p_t^h$ be the true positive rate for low, mid, and high cost sensors respectively. Then we have the following linear approximations, shown as dashed lines in Fig. 2,

$$p_t^l = 0.0041 p_f^l + 0.9949 \quad \text{for } p_f^l \in [0.2, 1.0] \tag{34}$$

$$p_t^m = 0.0853 p_f^m + 0.9137 \quad \text{for } p_f^m \in [0.2, 1.0] \tag{35}$$

$$p_t^h = \quad \text{for } p_f^h \in [0.2, 1.0], \tag{36}$$

and the following quadratic approximations, plotted with dots in Fig. 2,

$$p_t^l = -0.0183 {p_f^l}^2 + 0.0273 p_f^l + 0.9888 \quad \text{for } p_f^l \in [0.2, 1.0] \tag{37}$$

$$p_t^m = -0.2243 {p_f^m}^2 + 0.3779 p_f^m + 0.8391 \quad \text{for } p_f^m \in [0.2, 1.0] \tag{38}$$

$$p_t^h = -0.4676 {p_f^h}^2 + 0.9213 p_f^h + 0.5346 \quad \text{for } p_f^h \in [0.2, 1.0]. \tag{39}$$

Let $q_f$ be the false negative rate and $q_t$ be the true negative rate. Then note that given a true positive rate $p_t$ and a false positive rate $p_f$, $q_f = 1 - p_t$ and $q_t = 1 - p_f$.