Statistical Model Checking

Axel Legay[1], Anna Lukina[2(✉)], Louis Marie Traonouez[1], Junxing Yang[3], Scott A. Smolka[3], and Radu Grosu[2]

[1] Inria Rennes – Bretagne Atlantique, Rennes, France
anna.lukina@tuwien.ac.at
[2] Cyber-Physical Systems Group, Technische Universität Wien, Vienna, Austria
[3] Department of Computer Science, Stony Brook University, Stony Brook, USA

Abstract. We highlight the contributions made in the field of *Statistical Model Checking* (SMC) since its inception in 2002. As the formal setting, we use a very general model of *Stochastic Systems* (an SS is simply a family of time-indexed random variables), and *Bounded LTL* (BLTL) as the temporal logic. Let S be an SS and φ a BLTL formula. Our survey of the area is centered around the following five main contributions.

Qualitative approach to SMC: Is the probability that S satisfies φ greater or equal to a certain threshold?

Quantitative approach to SMC: What is the probability that S satisfies φ? Typically this results in a confidence interval being computed for this probability.

Rare Events: What happens when the probability that S satisfies φ is extremely small, i.e. it is a rare event? To make the SMC approach viable in this setting, rare-event estimation techniques *Importance Sampling* and *Importance Splitting* are deployed to great advantage.

Optimal Planning: Motivated by the success of Importance Sampling and Importance Splitting in rare-event SMC, we explore the use of these techniques in the context of optimal planning. In particular, we consider ARES, an optimal-planning approach based on a notion of adaptive receding-horizon planning. We illustrate the utility of ARES on the planning problem of bringing a flock of birds (autonomous agents) from a random initial configuration to a *V-formation*, an energy-conservation formation deployed by migrating geese. Somewhat ironically, the performance of ARES can be evaluated using (quantitative) SMC, as the problem to be solved is of the form $F(J \leq \theta)$; i.e. does an ARES-generated plan eventually bring the flock to a configuration where the flock-wide cost function J is below a given threshold θ?

Optimal Control: We show that the techniques we presented for optimal planning in the form of ARES carry over to the control setting in the form of *Adaptive-Horizon Model-Predictive Control* (AMPC). We again use the V-formation problem for evaluation purposes. We also introduce the concept of *V-formation games*, and show how the power of AMPC can be used to ward off cyber-physical attacks.

B. Steffen and G. Woeginger (Eds.): Computing and Software Science, LNCS 10000, pp. 478–504, 2019.
https://doi.org/10.1007/978-3-319-91908-9_23

1 Introduction

Quantitative models of computer systems include *stochastic systems*, whose state transitions are equipped with a probability distribution. Stochastic systems in turn include both discrete- and continuous-time Markov Chains. Our main interest will be in computing the probability by which a stochastic system S satisfies a given temporal-logic property φ. In contrast to the Boolean version of the model checking problem, this *quantitative model checking* (QMC) problem allows one to precisely determine how well S satisfies φ. When φ is a linear temporal logic (LTL) formula, QMC serves as a way to measure the number of paths that satisfy the formula.

The QMC problem is typically solved by a numerical approach that, like state-space exploration, iteratively computes (or approximates) the exact measure of paths satisfying relevant subformulas. The algorithm for computing such measures depends on the class of stochastic systems being considered as well as the logics used for specifying the correctness properties. QMC algorithms for a variety of such contexts have been discovered [1,8,9] and there are mature tools (see e.g. [7,28]) that have been used to analyze a variety of systems in practice.

Despite the great strides made by numerical QMC algorithms, there are many challenges. Numerical algorithms work only for systems that have certain structural properties. Further, these algorithms have significant time and space requirements, and thus scaling to large systems is a challenge. Also, the temporal logics supported by these QMC algorithms are extensions of classical temporal logics that are not particularly popular among engineers. Finally, numerical techniques do not easily scale to extended stochastic models whose semantics also depends on other quantities such as real-time or energy.

Another approach to QMC is to *simulate* the system for finitely many runs, and use techniques from the area of statistics to infer whether the samples provide a *statistical* evidence for the satisfaction or violation of the specification [41]. The crux of this approach is that since sample runs of a stochastic system are drawn according to the distribution defined by the system, they can be used to obtain estimates of the probability measure on executions. These techniques are known under the name of *Statistical Model Checking* (SMC).

The SMC approach enjoys many advantages. First, these algorithms only require the system to be simulatable (or rather, sample executions be drawn according to the measure space defined by the system). Thus, it can be applied to a larger class of systems than numerical QMC algorithms, including black-box systems and infinite-state systems. Second, the approach can be generalized to a larger class of properties, i.e., Fourier transform-based logics [2,3]. Finally, the algorithm is easily parallelizable, which can help scale to large systems. In cases where the model-checking problem is undecidable or too complex, SMC is often the only viable solution. As we shall see, SMC has been the subject of intensive research. SMC algorithms have been implemented in a series of tools, including Ymer [39], Prism [29], and UPPAAL [10]. Recently, we have implemented a series of SMC techniques in a flexible and modular toolset called Plasma Lab [4].

Despite the successes SMC has enjoyed, a serious obstacle in its application is its poor performance in predicting the satisfaction of properties holding with very low probability, so-called *rare events* (REs). In such cases, the number of samples required to attain a high confidence ratio and a low error margin explodes [16,42]. Two sequential Monte-Carlo techniques, *importance sampling (ISam)* [12] and *importance splitting (ISpl)* [15], originally developed for statistical physics [25], promise to overcome this obstacle. We discuss the important role these techniques have come to play in the SMC arena and beyond, in particular, for the purposes of optimal planning and control of controllable MDPs, a popular strategy-based probabilistic modeling formalism [14].

With this background in place, the following discussion summarizes the main contributions of this chapter. It also serves as a guide to how the chapter is organized.

- Section 2 provides definitions of the two basic ingredients of the SMC problem. It presents a very general definition of stochastic system as a family of time-indexed random variables, and it also introduces the temporal logic Bounded LTL (BLTL), which is often used in the SMC setting.
- Section 3 describes SMC-based approaches to both the qualitative and quantitative stochastic verification problems [33,39]. The qualitative version is of the form "How is the probability of property satisfaction related to a given threshold?", whereas the quantitative version asks the question "What is a confidence interval for this probability?"
- Section 4 considers the impact of *rare events* on the performance of SMC. As discussed above, we show how *importance sampling* and *importance splitting* can be successfully used for the statistical model checking of rare-event properties. In particular, we consider *command-based importance sampling* which is intended to reduce the computational burden imposed by ISam. Instead of reiterating the process of choosing a good distribution, the command-based approach considers parametrization over the syntax of stochastic guarded commands. We thereafter investigate the application of importance splitting with fixed and adaptive levels for rare-event probability estimation. These approaches are implemented in the Plasma toolset, thereby allowing for a thorough evaluation of their performance.
- Section 5 shows how the rare-event approach to SMC can be exploited for the purpose of optimal plan synthesis for controllable MDPs. Specifically, we present ARES, an efficient approximation algorithm for generating optimal plans (action sequences) that take an initial state of an MDP to a state whose cost is below a specified (convergence) threshold [30]. ARES uses Particle Swarm Optimization (PSO), with *adaptive sizing* for both the receding horizon and the particle swarm. Inspired by Importance Splitting, the length of the horizon and the number of particles are chosen such that at least one particle reaches a *next-level* state, that is, a state where the cost decreases by a required delta from the previous-level state.
- Section 6 demonstrates the utility of importance splitting and PSO in the context of control, where we present a new formulation of model-predictive

control called *Adaptive-Horizon MPC* (AMPC). We show that under certain controllability conditions, an AMPC controller can bring a system to an optimal state (WRT a given cost function) with probability 1. Somewhat ironically, we provide statistical guarantees of the performance of AMPC and ARES using SMC, the same approach that inspired our use of rare-event techniques in the first place.
– Section 7 draws our conclusions and discusses future work in the area.

2 Formal Definitions

In this section, we introduce several formal definitions that will be used in the rest of the chapter. We consider a set of states S and a time domain $T \subseteq \mathbb{R}$. We first introduce the general definition of a stochastic system.

Definition 1 (Stochastic system). *A* stochastic system *over S and T is a family of random variables $\mathcal{X} = \{X_t \mid t \in T\}$, each random variable X_t having range S.*

The definition of a stochastic system as a family of random variables is quite general and includes systems with both continuous and discrete dynamics. In this work, we will focus our attention on a limited, but important, class of stochastic system: stochastic discrete event systems, which we note $\mathcal{S} = (S, T)$. This class includes any stochastic system that can be thought of as occupying a single state for a duration of time before an event causes an instantaneous state transition to occur. An *execution* for a stochastic system is any sequence of observations $\{x_t \in S \mid t \in T\}$ of the random variables $X_t \in \mathcal{X}$. It can be represented as a sequence $\omega = (s_0, t_0), (s_1, t_1), \ldots, (s_n, t_n) \ldots$, such that $s_i \in S$ and $t_i \in T$, with time stamps monotonically increasing, e.g. $t_i < t_{i+1}$. Let $0 \leq i \leq n$, we denote $\omega^i = (s_i, t_i), \ldots, (s_n, t_n)$ the suffix of ω starting at position i. Let $\overline{s} \in S$, we denote $Path(\overline{s})$ the set of executions of $\mathcal{X}$ that starts in state $(\overline{s}, 0)$ (also called initial state) and $Path^n(\overline{s})$ the set of executions of length n.

In [39], Younes showed that the set of executions of a stochastic system is a measurable space, which defines a probability measure μ over $Path(\overline{s})$. The precise definition of μ depends on the specific probability structure of the stochastic system being studied.

Properties over traces of $\mathcal{S}ys$ are defined via the so-called Bounded Linear Temporal Logic (BLTL). BLTL restricts Linear Temporal Logic by bounding the scope of the temporal operators. The syntax of BLTL is defined as follows:

$$\phi = \phi \vee \phi \mid \phi \wedge \phi \mid \neg\phi \mid \mathrm{F}^{\leq t}\phi \mid \mathrm{G}^{\leq t}\phi \mid \phi\,\mathrm{U}^{\leq t}\phi \mid \mathrm{X}\phi \mid \alpha$$

$\vee, \wedge$ and $\neg$ are the standard logical connectives and α is a Boolean constant or an atomic proposition constructed from numerical constants, state variables and relational operators. X is the *next* temporal operator: $\mathrm{X}\phi$ means that ϕ will be true on the next step. F, G and U are temporal operators bounded by time interval $[0, t]$, relative to the time interval of any enclosing formula. We refer to this as a *relative interval*. F is the *finally* or *eventually* operator: $\mathrm{F}^{\leq t}\phi$

means that ϕ will be true at least once in the relative interval $[0, t]$. G is the *globally* or *always* operator: $G^{\leq t}\phi$ means that ϕ will be true at all times in the relative interval $[0, t]$. U is the *until* operator: $\psi \mathrm{U}_{\leq t}\phi$ means that in the relative interval $[0, t]$, either ϕ is initially true or ψ will be true until ϕ is true. Combining these temporal operators creates complex properties with interleaved notions of *eventually* (F), *always* (G) and *one thing after another* (U).

3 On Verifying Requirements: The SMC Approach

Consider a stochastic system (S, T) and a property ϕ. *Statistical model checking* refers to a series of simulation-based techniques that can be used to answer two questions: (1) **Qualitative:** Is the probability that (S, T) satisfies ϕ greater or equal to a certain threshold? and (2) **Quantitative:** What is the probability that (S, T) satisfies ϕ? Contrary to numerical approaches, the answer is given up to some correctness precision. As we shall see later, SMC solves those problems with two different approaches, while classical numerical approaches only solve the second problem, which implies the first one, but is harder.

In the rest of the section, we overview the two first statistical model checking techniques that were proposed in the literature. Let B_i be a discrete random variable with a Bernoulli distribution of parameter p. Such a variable can only take 2 values 0 and 1 with $Pr[B_i = 1] = p$ and $Pr[B_i = 0] = 1 - p$. In our context, each variable B_i is associated with one simulation of the system. The outcome for B_i, denoted b_i, is 1 if the simulation satisfies ϕ and 0 otherwise. The latter is decided with the help of a monitoring procedure [18]. The objective of an SMC algorithm is to generate simulations and exploit the Bernoulli outcomes to extract the global confidence on the system.

In the next subsections, we present three algorithms used in the early works on SMC to solve both the quantitative and the qualitative problems. Extension of those algorithms to unbounded temporal operators [17,34] and to nested probabilistic operators exist [39]. As shown in [19] those extensions are debatable and often slower. Consequently, we will not discuss them.

3.1 Qualitative Analysis Using Statistical Model Checking

The main approaches [33,39] proposed to answer the qualitative question are based on *hypothesis testing*. Let $p = Pr(\phi)$, to determine whether $p \geqslant \theta$, we can test $H : p \geqslant \theta$ against $K : p < \theta$. A test-based solution does not guarantee a correct result but it is possible to bound the probability of making an error. The *strength* (α, β) of a test is determined by two parameters, α and β, such that the probability of accepting K (respectively, H) when H (respectively, K) holds, called a Type-I error (respectively, a Type-II error), is less or equal to α (respectively, β).

A test has *ideal performance* if the probability of the Type-I error (respectively, Type-II error) is exactly α (respectively, β). However, these requirements

make it impossible to ensure a low probability for both types of errors simultaneously (see [39] for details). A solution to this problem is to relax the test by working with an *indifference region* (p_1, p_0) with $p_0 \geqslant p_1$ ($p_0 - p_1$ is the *size of the region*). In this context, we test the hypothesis $H_0 : p \geqslant p_0$ against $H_1 : p \leqslant p_1$ instead of H against K. If the value of p is between p_1 and p_0 (the indifference region), then we say that the probability is sufficiently close to θ so that we are indifferent with respect to which of the two hypotheses K or H is accepted. The thresholds p_0 and p_1 are generally defined in terms of the single threshold δ, e.g., $p_1 = \theta - \delta$ and $p_0 = \theta + \delta$. We now need to provide a test procedure that satisfies the requirements above. In the next two subsections, we recall two solutions proposed by Younes in [39,40].

Single Sampling Plan. This algorithm plays more a historical role rather than to be used directly. However, it is still exploited in subsequent algorithms. To test H_0 against H_1, we specify a constant c. If $\sum_{i=1}^{n} b_i$ is larger than c, then H_0 is accepted, else H_1 is accepted. The difficult part in this approach is to find values for the pair (n, c), called a *single sampling plan (SSP in short)*, such that the two error bounds α and β are respected. In practice, one tries to work with the smallest value of n possible so as to minimize the number of simulations performed. Clearly, this number has to be greater if α and β are smaller but also if the size of the indifference region is smaller. This results in an optimization problem, which generally does not have a closed-form solution except for a few special cases [39]. In [39], Younes proposes a binary search based algorithm that, given p_0, p_1, α, β, computes an approximation of the minimal value for c and n.

Sequential Probability Ratio Test (SPRT). The sample size for a single sampling plan is fixed in advance and independent of the observations that are made. However, taking those observations into account can increase the performance of the test. As an example, if we use a single plan (n, c) and the $m > c$ first simulations satisfy the property, then we could (depending on the error bounds) accept H_0 without observing the $n-m$ other simulations. To overcome this problem, one can use the *sequential probability ratio test (SPRT in short)* proposed by Wald [36]. The approach is briefly described below.

In SPRT, one has to choose two values A and B ($A > B$) that ensure that the strength of the test is respected. Let m be the number of observations that have been made so far. The test is based on the following quotient:

$$\frac{p_{1m}}{p_{0m}} = \prod_{i=1}^{m} \frac{Pr(B_i = b_i | p = p_1)}{Pr(B_i = b_i | p = p_0)} = \frac{p_1^{d_m}(1-p_1)^{m-d_m}}{p_0^{d_m}(1-p_0)^{m-d_m}}, \tag{1}$$

where $d_m = \sum_{i=1}^{m} b_i$. The idea behind the test is to accept H_0 if $\frac{p_{1m}}{p_{0m}} \geq A$, and H_1 if $\frac{p_{1m}}{p_{0m}} \leq B$. The SPRT algorithm computes $\frac{p_{1m}}{p_{0m}}$ for successive values of m until either H_0 or H_1 is satisfied; the algorithm terminates with probability 1 [36]. This has the advantage of minimizing the number of simulations. In [39], Younes proposed a logarithmic based algorithm SPRT that given p_0, p_1, α and β implements the sequential ratio testing procedure.

SPRT has been largely used in the formal methods area. In this paper, we shall show that the approach extends to a much larger class of problems than the one originally foreseen.

3.2 Quantitative Analysis Using Statistical Model Checking and Estimation

In the case of estimation, existing SMC algorithms rely on classical Monte Carlo estimation. More precisely, they calculate a priori the required number of simulations according to a Chernoff bound [31] that allows the user to specify an error ε and a probability δ that the estimate $\hat{p}$ will not lie outside the true value $\pm\varepsilon$. Given that a system has true probability p of satisfying a property, the Chernoff bound ensures $\mathrm{P}(\mid \hat{p} - p \mid \geqslant \varepsilon) \leqslant \delta$. Parameter δ is related to the number of simulations N by $\delta = 2e^{-2N\varepsilon^2}$ [31], giving

$$N = \left\lceil (\ln 2 - \ln \delta)/(2\varepsilon^2) \right\rceil . \tag{2}$$

4 Rare Events

SMC is a Monte Carlo method that takes advantage of robust statistical techniques to bound the error of the estimated result (e.g., [31,36]). To quantify a property, it is necessary to observe the property, where increasing the number of observations generally increases the confidence of the estimate. Rare properties are often highly relevant to system performance (e.g., bugs and system failures are required to be rare) but pose a problem for statistical model checking because they are difficult to observe. Fortunately, rare-event techniques such as *importance sampling* [24,26] and *importance splitting* [25,26,32] may be successfully applied to statistical model checking.

Importance sampling and importance splitting have been widely applied to specific simulation problems in science and engineering. Importance sampling works by estimating a result using weighted simulations and then compensating for the weights. Importance splitting works by reformulating the rare probability as a product of less rare probabilities, conditioned on the levels that must be achieved.

In this section, we summarize our contributions in terms of applying importance sampling and importance splitting to the SMC problem. We then discuss their implementation within the Plasma toolset.

4.1 Command-Based Importance Sampling

Importance sampling works by simulating a probabilistic system under a weighted (*importance sampling*) measure that makes a rare property more likely to be seen [23]. It then compensates the results by the weights, to estimate the probability under the original measure. When simulating Markov Chains, this compensation is typically performed on the fly, with almost no additional overhead.

Given a set of finite traces $\omega \in \Omega$ and a function $z : \Omega \to \{0, 1\}$ that returns 1 iff a trace satisfies some property, the importance sampling estimator is given by

$$\sum_{i=1}^{N} z(\omega_i) \frac{\mathrm{d}f(\omega_i)}{\mathrm{d}f'(\omega_i)}.$$

N is the number of simulation traces ω_i generated under the importance sampling measure f', while f is the original measure. $\frac{\mathrm{d}f}{\mathrm{d}f'}$ is the *likelihood ratio*.

For importance sampling to be effective it is necessary to define a "good" importance sampling distribution: (i) the property of interest must be seen frequently in simulations and (ii) the distribution of the simulation traces that satisfy the property in the importance sampling distribution must be as close as possible to the normalized distribution of the same traces in the original distribution. Failure to consider both (i) and (ii) can result in underestimated probability with overestimated confidence.

Since the main motivation of importance, sampling is to reduce the computational burden, the process of finding a good importance sampling distribution must maintain the scaling advantage of SMC and, in particular, should not iterate over all the states or transitions of the system. We, therefore, consider parameterized importance sampling distributions, where our parametrization is over the syntax of stochastic *guarded commands*, a common low-level modeling language of probabilistic systems[1].

Each command has the form (*guard*, *rate*, *action*). The *guard* enables the command and is a predicate over the state variables of the model. The *rate* is a function from the state variables to $\mathbb{R}_{>0}$, defining the rate of an exponential distribution. The *action* is an update function that modifies the state variables. In general, each command defines a set of semantically linked transitions in the resulting Markov chain.

The semantics of a stochastic guarded command is a Markov jump process (has discrete movements with random arrival times, i.e., a Poisson process). The semantics of a parallel composition of commands is a system of concurrent Markov jump processes. Sample execution traces can be generated by discrete-event simulation. In any state, zero or more commands may be enabled. If no commands are enabled the system is in a halting state. In all other cases, the enabled commands "compete" to execute their actions: sample times are drawn from the exponential distributions defined by their rates and the shortest time "wins". As showed in [20], this optimization can be performed, e.g., with the cross-entropy method. The techniques also extend to real-time stochastic systems (see [22]).

4.2 Importance Splitting

The earliest application of importance splitting is perhaps that of [24,25], where it is used to calculate the probability that neutrons pass through certain shielding

[1] http://www.prismmodelchecker.org/manual/ThePRISMLanguage/.

materials. This physical example provides a convenient analogy for the more general case. The system comprises a source of neutrons aimed at one side of a shield of thickness T. The distance traveled by a neutron in the shield defines a monotonic sequence of levels $\ell_0 = 0 < \ell_1 < \ell_2 < \cdots < \ell_n = T$, such that reaching a given level implies having reached all the lower levels. While the overall probability of passing through the shield is small, the probability of passing from one level to another can be made arbitrarily close to 1 by reducing the distance between levels. Denoting the abstract level of a neutron as ℓ, the probability of a neutron reaching level ℓ_i can be expressed as $\mathrm{P}(\ell \geqslant \ell_i) = \mathrm{P}(\ell \geqslant \ell_i \mid \ell \geqslant \ell_{i-1})\mathrm{P}(\ell \geqslant \ell_{i-1})$. Defining $\gamma = \mathrm{P}(\ell \geqslant \ell_m)$ and $\mathrm{P}(\ell \geqslant \ell_0) = 1$, we obtain

$$\gamma = \prod_{i=1}^{m} \mathrm{P}(\ell \geqslant \ell_i \mid \ell \geqslant \ell_{i-1}). \tag{3}$$

Each term of (3) is necessarily greater than or equal to γ, making their estimation easier.

The general procedure is as follows. At each level, a number of simulations are generated, starting from a distribution of initial states that corresponds to reaching the current level. It starts by estimating $\mathrm{P}(\ell \geqslant \ell_1 | \ell \geqslant \ell_0)$, where the distribution of initial states for ℓ_0 is usually given (often a single state). Simulations are stopped as soon as they reach the next level; the final states becoming the empirical distribution of initial states for the next level. Simulations that do not reach the next level (or reach some other stopping criterion) are discarded. In general, $\mathrm{P}(\ell \geqslant \ell_i | \ell \geqslant \ell_{i-1})$ is estimated by the number of simulation traces that reach ℓ_i, divided by the total number of traces started from ℓ_{i-1}. Simulations that reached the next level are continued from where they stopped. To avoid a progressive reduction of the number of simulations, the generated distribution of initial states is sampled to provide additional initial states for new simulations, thus replacing those that were discarded.

Score Function. The concept of levels can be generalized to arbitrary systems and properties in the context of SMC, treating ℓ and ℓ_i in (3) as values of a score function over the model-property product automaton. Intuitively, a score function discriminates good paths from bad, assigning higher scores to paths that more nearly satisfy the overall property. Since the choice of levels is crucial to the effectiveness of importance splitting, various ways to construct score functions from a temporal logic property are proposed in [20].

Formally, given a set of finite trace prefixes $\omega \in \Omega$, an ideal score function $S : \Omega \to \mathbb{R}$ has the characteristics $S(\omega) > S(\omega') \iff \mathrm{P}(\models \varphi \mid \omega) > \mathrm{P}(\models \varphi \mid \omega')$, where $\mathrm{P}(\models \varphi \mid \omega)$ is the probability of eventually satisfying φ given prefix ω. Intuitively, ω has a higher score than ω' iff there is more chance of satisfying φ by continuing ω than by continuing ω'. The minimum requirement of a score function is $S(\omega) \geq s_\varphi \iff \omega \models \varphi$, where s_φ is an arbitrary value denoting that φ is satisfied. Any trace that satisfies φ must have a score of at least s_φ and any trace that does not satisfy φ must have a score less than s_φ. In what follows we assume that (3) refers to scores.

The Fixed-Levels Algorithm. The fixed-levels algorithm follows the general procedure previously presented. Its advantages are that it is simple, it has low computational overhead, and the resulting estimate is unbiased. Its disadvantage is that the levels must often be guessed by trial and error, adding to the overall computational cost.

In Algorithm 1, $\tilde{\gamma}$ is an unbiased estimate (see, e.g., [11]). Furthermore, from Proposition 3 in [5], we can deduce the following $(1-\alpha)$-confidence interval:

$$CI = \left[\hat{\gamma}/\left(1+\frac{z_\alpha \sigma}{\sqrt{n}}\right), \hat{\gamma}/\left(1-\frac{z_\alpha \sigma}{\sqrt{n}}\right)\right] \qquad \text{with} \qquad \sigma^2 \geqslant \sum_{i=1}^{m} \frac{1-\gamma_i}{\gamma_i}. \tag{4}$$

Confidence is specified via z_α, the $1-\alpha/2$ quantile of the standard normal distribution, while n is the per-level simulation budget. We infer from (4) that for a given γ the confidence is maximized by making both the number of levels m and the simulation budget n large, with all γ_i equal.

Algorithm 1. Fixed levels

Let $(\tau_k)_{1\leq k\leq m}$ be the sequence of thresholds with $\tau_m = \tau_\varphi$
Let *stop* be a termination condition
$\forall j \in \{1,\ldots,n\}$, set prefix $\tilde{\omega}_j^1 = \epsilon$ (empty path)
for $1 \leq k \leq m$ **do**
  $\forall j \in \{1,\ldots,n\}$, using prefix $\tilde{\omega}_j^k$, generate path ω_j^k until $(S(\omega_j^k) \geq \tau_k) \vee stop$
  $I_k = \{\forall j \in \{1,\ldots,n\} : S(\omega_j^k) \geq \tau_k\}$
  $\tilde{\gamma}_k = \frac{|I_k|}{n}$
  $\forall j \in I_k$, $\tilde{\omega}_j^{k+1} = \omega_j^k$
  $\forall j \notin I_k$, let $\tilde{\omega}_j^{k+1}$ be a copy of ω_i^k with $i \in I_k$ chosen uniformly randomly
$\tilde{\gamma} = \prod_{k=1}^{m} \tilde{\gamma}_k$

In general, however, score functions will not equally divide the conditional probabilities of the levels, as required by (4) to minimize variance. In the worst case, one or more of the conditional probabilities will be too low for the algorithm to pass between levels. Finding good or even reasonable levels by trial and error may be computationally expensive and has prompted the development of adaptive algorithms that discover optimal levels on the fly [6,20,21]. Instead of pre-defining levels, the user specifies the proportion of simulations to retain after each iteration. This proportion generally defines all but the final conditional probability in (3).

Adaptive importance splitting algorithms first perform a number of simulations until the overall property is decided, storing the resulting traces of the model-property automaton. Each trace induces a sequence of scores and a corresponding maximum score. The algorithm finds a level that is less than or equal to the maximum score of the desired proportion of simulations to retain. The simulations whose maximum score is below this current level are discarded. New

simulations to replace the discarded ones are initialized with states corresponding to the current level, chosen at random from the retained simulations. The new simulations are continued until the overall property is decided and the procedure is repeated until a sufficient proportion of simulations satisfy the overall property.

4.3 Rare Events: Comparison of Methods

In this section, we compare the two rare-event approaches with the Monte Carlo approach.

Model. We consider a chemically reacting system that consists of a set of three chemical reactions between five molecular species (A, B, C, D, E). These reactions are the following:

$$A + B \rightarrow C \tag{5}$$

$$C \rightarrow D \tag{6}$$

$$D \rightarrow E \tag{7}$$

Initially, the system only contains species A and B. Then reactions start according to a rate that depends on the number of reactants. We model this system as a continuous time Markov chain (CTMC) using the Reactive Module Language (RML), the input language of the tools Prism and Plasma Lab. The code of the model, presented in Fig. 1, contains a single module component with three transitions to model the three reactions. The quantities of each element are modeled as an integer variable from 0 to 1000. A and B start with 1000 elements, while C, D, and E start at zero. A transition that models a chemical reaction is composed of a guard, that is always true, a rate, e.g., $\mathsf{a} * \mathsf{b}$, and a set of updates, e.g., $(\mathsf{a}' = \mathsf{a} - 1)$. The rate of the transition defines the speed of the reaction as the rate of an exponential distribution: the higher it is the faster will be the reaction. An example of a simulation of this system is presented in Fig. 2. It shows how the quantities of each species may evolve over time, where time is presented as the number of steps (chemical reaction) performed by the system.

Property. We consider a bounded linear temporal logic formula as the property of this system to be verified:

$$\varphi := F \leqslant 3000(d > 470)$$

It checks if a level of 471 for species D can be reached within 3000 steps. We would like to estimate the probability of satisfying this formula. As we can see in a typical simulation run of this system in Fig. 2, species D tends to reach a maximum of 400 before being transformed into species E.

Model Checking. The first approach to compute the probability of φ would be to use a probabilistic model checker like PRISM to compute its exact value. However, this model checking problem is intractable due to the size of its state space (10^{15}).

```
ctmc

module chem
  a : [0..1000] init 1000;
  b : [0..1000] init 1000;
  c : [0..1000] init 0;
  d : [0..1000] init 0;
  e : [0..1000] init 0;
  [] true -> a*b :
    (a'=a-1) & (b'=b-1) & (c'=c+1)
       ;
  [] true -> c :
    (c'=c-1) & (d'=d+1);
  [] true -> d :
    (d'=d-1) & (e'=e+1);
endmodule
```

Fig. 1. CTMC model of chemical reactions written in the Reactive Module Language

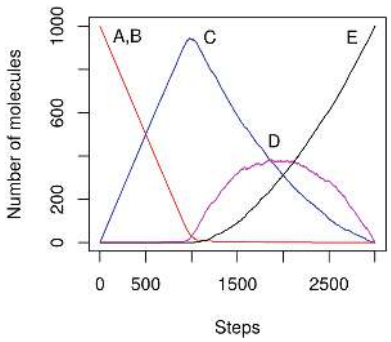

Fig. 2. Simulation of the evolution of chemical species through time (in number of steps)

Monte Carlo. Consider next the Monte Carlo statistical model checking approach. We used Plasma Lab to run 1,000,000 simulations on an 8-core 2.6 GHz computer. It took 839 s, but we were not able to find even one trace that satisfies the formula. To have an idea of the evolution of the probability according to maximum value of species D checked by the property, we plot in Fig. 3 the results of Monte Carlo analyses with 100,000 simulations for several values of the maximum value from 350 to 450. As one can see, the probability to reach a maximum greater than 400 of species D is very low. To estimate the probability of reaching 471 we need to use statistical techniques for rare events.

Importance Splitting. Importance splitting works by splitting the verification of a rare property into a sequence of less rare properties. For instance, in our problem, any trace that eventually satisfies the formula φ, satisfies the formulas $F <= 3000(d > 460)$, $F <= 3000(d > 450)$, $F <= 3000(d > 440)$, etc. Therefore, the maximum value of d reached by a trace defines a natural notion of a level that can be used to split the rare property into a sequence of less-rare properties. We implement this decomposition in Plasma Lab by writing an observer that computes the score of a trace, i.e., the maximum value of d along the trace.

We can then use the adaptive importance splitting of Plasma Lab to estimate the probability of φ. The results are summarized in Table 1. We performed three experiments of the algorithm with a different number of simulations (100, 200, 500). Each experiment is repeated 20 times and we report the average value of the estimated probability, number of levels, and computation time. We also report the standard deviation of the probability and the relative standard deviation (quotient of standard deviation and average probability).

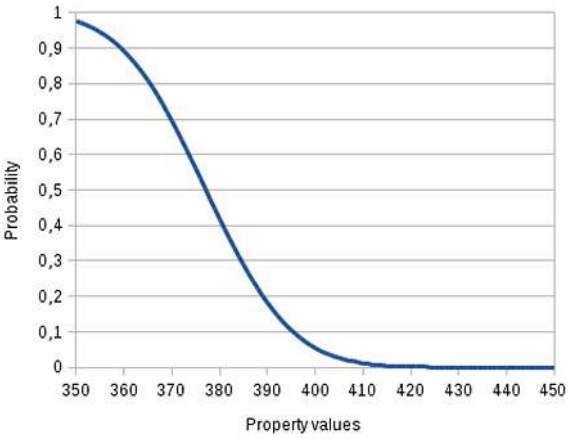

Fig. 3. Probability estimation with Monte Carlo for a maximum value on D that ranges from 350 to 450

Using this technique we are able to find traces that satisfy the property in 5 s with 1 core of a 2.6 GHz computer, using only a budget of 100 simulations. However, we were not able to run the adaptive algorithm with a budget of 1000 as we ran out of memory. Indeed this algorithm is more memory-intensive than classical Monte Carlo as it needs to keep in memory all the traces.

The relative standard deviation is a good measure of the performance of the estimator. A reliable estimator should have a relative standard deviation lower than 0.3. As can be seen, the relative standard deviation of our adaptive estimator is much higher. It tends to decrease when increasing the number of simulations, but using this algorithm we are limited by the memory.

Table 1. Results of the adaptive importance splitting algorithm

Nb. simulations	100	200	500
Nb. levels	116.6	121.2	127.9
Probability	6.02×10^{-11}	9.46×10^{-11}	2.07×10^{-10}
Std. deviation	1.44×10^{-10}	1.76×10^{-10}	4.51×10^{-10}
Relative std. deviation	2.39	1.86	2.18
Time (s)	5	14.6	55

The original importance-splitting algorithm uses a fixed number of levels. This algorithm is less memory intensive because it only keeps in memory the final states of the simulations. However, we must specify by hand the intermediate levels that we want to reach. To minimize the variance of the estimator, we should select as much as possible a set of levels whose conditional probabilities

are equal. We selected 32 levels of d between $[380, 471]$ and ran the importance-splitting algorithm for a different number of simulations. We report the results in Table 2. They show that when increasing the number of simulations, we improve the relative deviation of the results.

Table 2. Results of the fixed levels importance splitting algorithm

Nb. simulations	1000	2000	5000
Probability	1.35×10^{-10}	1.28×10^{-10}	9.83×10^{-11}
Std. deviation	1.74×10^{-10}	9.07×10^{-11}	6.44×10^{-11}
Relative std. deviation	1.28	0.706	0.655
Time (s)	30.7	47.3	114

Importance Sampling. In Plasma Lab, we implemented the importance sampling algorithm for the Reactive Module Language. It requires adding sampling parameters to the model in order to modify the rate of some transitions. To produce a good result, the sampling parameters should have optimal values. This is determined using the minimum cross-entropy algorithm. This algorithm iteratively determines the values of the sampling parameters by running Monte Carlo experiments and counting the number of times each transition is used.

To use importance sampling on our model, we replace it by the one in Fig. 4. The three sampling parameters are named lambda. They are each associated with a counter variable nb_lambda. Parameters are initialized such that every simulation satisfied the rare property φ.

We then use the minimum cross-entropy algorithm to determine the optimal values of the parameters. In a run of the algorithm, we use 50 iterations to determine the final values of the parameters. Fig. 5 illustrates the evolution of the three parameters during a run of the algorithm. We ran the algorithm 10 times with 50 iterations and 1000 at each iteration. We report the results in Fig. 6. In this problem, the algorithm provides the best results, with a relative standard deviation lower than 0.3.

5 Importance Splitting/Sampling for Optimal Planning

In this section, we demonstrate how the incorporation of importance splitting and importance sampling into SMC for the treatment of rare-event properties has inspired planning algorithms for Markov decision processes (MDPs), a popular modeling formalism for policy-based stochastic systems. The goal of Sect. 6 is similar, but in this case for control algorithms. Planning and control often go hand-in-hand, with planning focused on long-term system objectives (e.g., how can an autonomous system get from point A to point B by following a sequence of so-called waypoints), and with control focused on the sub-second decisions the system must make in order to realize the planning objectives in question.

```
ctmc sampling

const double lambda1 = 2;
const double lambda2 = 1;
const double lambda3 = 0.1;
global nb_lambda1 : int init 0;
global nb_lambda2 : int init 0;
global nb_lambda3 : int init 0;

module test
a : [0..1000] init 1000;
b : [0..1000] init 1000;
c : [0..1000] init 0;
d : [0..1000] init 0;
e : [0..1000] init 0;
[] true -> {lambda1} a*b : (a'=a-1)&(b'=b-1)&(c'=c+1)
                         & (nb_lambda1'=nb_lambda1+1);
[] true -> {lambda2} c : (c'=c-1)&(d'=d+1)
                       & (nb_lambda2'=nb_lambda2+1);
[] true -> {lambda3} d : (d'=d-1)&(e'=e+1)
                       & (nb_lambda3'=nb_lambda3+1);
endmodule

label "rate_lambda1" = a*b;
label "rate_lambda2" = c;
label "rate_lambda3" = d;
```

Fig. 4. CTMC model of chemical reactions with sampling parameters

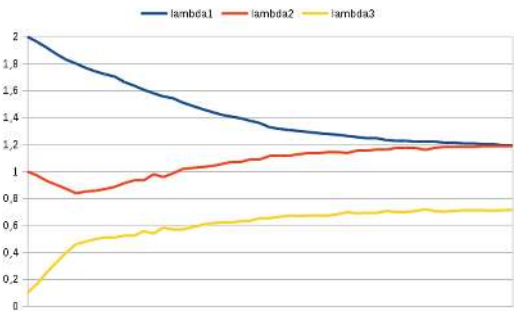

Fig. 5. Evolution of the three sampling parameters during a run of the minimum cross-entropy algorithm

Nb. sims.	1000
Prob.	1.50×10^{-10}
Std. dev.	$4,03 \times 10^{-11}$
Rel. std. dev.	$0,269$
Time(s)	118

Fig. 6. Results of the importance sampling algorithm with minimum cross-entropy

Definition 2. *A **Markov decision process (MDP)** $\mathcal{M}$ is a sequential decision problem that consists of a set of states S (with an initial state s_0), a set of actions A, a transition model T, and a cost function J. An MDP is **deterministic** if for each state and action, $T : S \times A \rightarrow S$ specifies a unique state.*

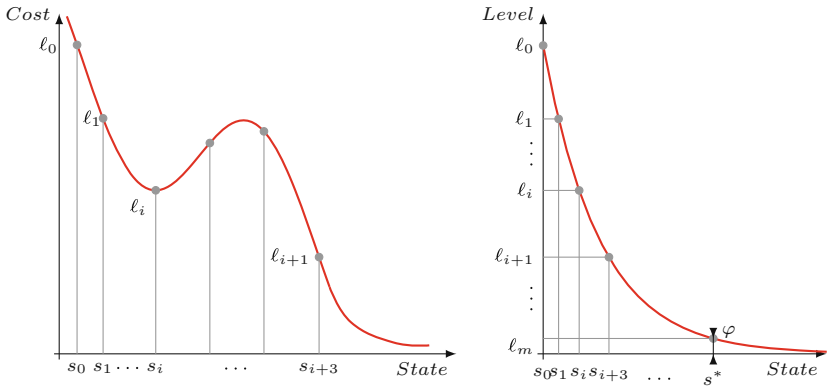

Fig. 7. Left: If state s_0 has cost ℓ_0 and its successor-state s_1 has cost less than ℓ_1, then a horizon of length 1 is appropriate. If, however, s_i has a local-minimum cost ℓ_i, one has to pass over the cost ridge in order to reach level ℓ_{i+1}, and therefore ARES has to adaptively increase the horizon to 3. Right: The cost of the initial state defines ℓ_0 and the given threshold φ defines ℓ_m. By choosing m equal segments on an asymptotically converging (Lyapunov) function (where m is empirically determined), one obtains on the vertical cost-axis the levels required for ARES to converge.

The particular planning and control problems addressed here are concerned with V-formation in a flock of birds, a quintessential example of emergent behavior in a distributed stochastic system. V-formation brings numerous benefits to the flock. It is primarily known for being energy-efficient due to the upwash benefit a bird in the flock enjoys from its frontal neighbor. It also offers each bird a clear frontal view, unobstructed by any flockmate. Moreover, the collective spatial mass of a V-formation can be intimidating to potential predators.

5.1 The Optimal Plan Synthesis Problem

In [30] we presented ARES, a general *adaptive, receding-horizon synthesis algorithm* that given an MDP and one of its initial states, generates an optimal plan (action sequence) taking that state to a state whose cost is below a desired threshold. To improve the probability of reaching a V-formation in a bird flock via ARES-based planning, we considered this phenomenon as a rare event and incorporated importance splitting into the core of the ARES algorithm. This level-based approach allows ARES to steer the. We then use SMC system towards the desired configuration to estimate this reachability probability.

Definition 3. *The* ***optimal plan synthesis problem*** *for an MDP $\mathcal{M}$, an arbitrary initial state s_0 of $\mathcal{M}$, and a threshold φ, is to synthesize a sequence of actions $\boldsymbol{a}^i$ of length $1 \leqslant i \leqslant m$ taking s_0 to a state s^* such that cost $J(s^*) \leqslant \varphi$.*

ARES uses the particle swarm optimization (PSO) algorithm [27] at each time step to incrementally generate a plan. Each particle in a PSO swarm is

a realization of a random variable that is taken as a candidate optimal action (acceleration) sequence which can be used to simulate $\mathcal{M}$. The model is cloned into $\mathcal{M}_k$ instances, $k = 1, \ldots, n$, and a PSO swarm is assigned to each clone. These clones are later considered as independent simulation runs in the fashion of importance sampling.

This incremental approach to optimal-plan construction is in principle unnecessary, as one could generate an optimal plan in its entirety by calling PSO only once and running it until the global optimum is found or time bound is reached. Such an approach, however, is impractical, as each (transition-based) unfolding of the MDP adds a number of new dimensions to the search space. Consequently, to obtain adequate coverage of the monolithic optimal-plan search space, one needs a very large number of particles, a number that is either going to exhaust available memory or require a prohibitive amount of time to find an optimal plan.

A simple solution to this problem is to use a short horizon, typically of size two or three. This is indeed the current practice in model-predictive control (MPC) [13]. This approach, however, has at least three major drawbacks. First, and most importantly, it does not guarantee convergence and optimality, as one may oscillate or become stuck in a local optimum. Second, in some of the steps, the window size is unnecessarily large thereby negatively impacting performance. Third, in other steps, the window size may not be large enough to guide the optimizer out of a local minimum; see Fig. 7(left). One would therefore like to find the proper window size adaptively, but the question is how can one do this?

5.2 Adaptive Receding-Horizon Synthesis of Optimal Plans

Inspired by the SMC-based *importance splitting* technique (ISp) described in Sect. 4.2, we introduce the notion of a *level-based horizon*, where level ℓ_0 equals the cost of the initial state, and level ℓ_m equals the target threshold φ. By using an asymptotic cost-convergence function ranging from ℓ_0 to ℓ_m, and dividing its graph into m equal segments, we can determine on the vertical axis a sequence of levels ensuring convergence. See Fig. 7(right).

The asymptotic function ARES implements is essentially $\ell_i = \ell_0\,(m-i)/\,m$, but specifically tuned for each simulation of $\mathcal{M}$. Formally, if simulation k, $k = 1, \ldots, n$, has previously reached level $J_k(s_{i-1})$, then its next target level is within the distance $\Delta_k = J_k(s_{i-1})/(m-i+1)$. After passing the thresholds assigned to the simulations, the values of the cost function in the current state s_i are sorted in ascending order $\{\widehat{J}_k\}_{k=1}^{n}$. The lowest cost $\widehat{J}_1$ should be at least Δ_1 apart from the previous level ℓ_{i-1} for the algorithm to proceed to the next level $\ell_i := \widehat{J}_1$.

The levels serve two purposes. First, they implicitly define a Lyapunov function, which guarantees convergence. If desired, this function can be explicitly generated for all states, up to some topological equivalence. Second, the levels ℓ_i help PSO overcome local minima; see Fig. 7(left). If reaching the next level requires PSO to pass over a state-cost ridge, then ARES incrementally increases the size of the horizon h, up to a maximum size h_{max}. For simulation k, passing

the thresholds Δ_k means that it reaches a new level, and the definition of Δ_k ensures a smooth degradation of its threshold.

Another idea imported from the statistical model checking of rare-event properties is *importance sampling* (IS). In our context, it means that we maintain n clones $\{\mathcal{M}_k\}_{k=1}^n$ of the MDP $\mathcal{M}$ (and its initial state) at any time t, and run PSO for prediction horizon h on each h-unfolding $\mathcal{M}_k^h$ of $\mathcal{M}$. This results in an action sequence $\boldsymbol{a}_k^h$ of length h (see Algorithm 2). This approach allows us to call PSO for each simulation and desired horizon, with a very small number p of particles per simulation.

To check which simulations have overcome their associated thresholds, we sort the simulation traces according to their current cost, and split them into two sets: the successful set, having the indexes $\mathcal{I}$ and whose costs are lower than the median among all clones; and the unsuccessful set with indexes in $\{1,\ldots,n\}\setminus\mathcal{I}$, which are discarded. The unsuccessful ones are further replenished, by sampling uniformly at random from the successful set $\mathcal{I}$ (see Algorithm 3).

The number of particles in PSO is increased to $p = p + p_{inc}$ if no simulation trace reaches the next level, for all horizons chosen. When this happens, we reset the horizon to one, and repeat the process. In this way, we adaptively focus our resources on escaping from local minima. From the last level, we choose the state s^* with the minimal cost, and traverse all of its predecessor states to find an optimal plan comprised of actions $\{\boldsymbol{a}^i\}_{1\leqslant i\leqslant m}$ that led MDP $\mathcal{M}$ to the optimal state s^*. In our running example, we select a flock in V-formation and traverse all its predecessor flocks. The overall ARES procedure is shown in Algorithm 4.

Proposition 1 (Optimality and Minimality). *(1) Let $\mathcal{M}$ be an MDP. For any initial state s_0 of $\mathcal{M}$, ARES is able to solve the optimal-plan synthesis problem for $\mathcal{M}$ and s_0. (2) An optimal choice of m in function Δ_k, for some simulation k, ensures that ARES also generates the shortest optimal plan.*

Proof (Sketch; see [30] for the details). (1) The dynamic-threshold function Δ_k ensures that the initial cost in s_0 is continually decreased until it falls below φ. Moreover, for an appropriate number of clones, by adaptively determining the horizon and the number of simulations needed to overcome Δ_k, ARES always converges, with probability 1, to an optimal state, given enough time and memory. (2) This follows from convergence property (1), and from the fact that ARES always gives preference to the shortest horizon while trying to overcome Δ_k.

Algorithm 2. Simulate $(\mathcal{M}, h, i, \{\Delta_k, J_k(s_{i-1})\}_{k=1}^n)$

foreach $\mathcal{M}_k \in \mathcal{M}$ **do**
  $[\boldsymbol{a}_k^h, \mathcal{M}_k^h] \leftarrow$ `particleswarm`$(\mathcal{M}_k, p, h)$; *// use PSO to determine best next action sequence for MDP $\mathcal{M}_k$ with RPH (receding prediction horizon) h*
  $J_k(s_i) \leftarrow$ `Cost`$(\mathcal{M}_k^h, \boldsymbol{a}_k^h, h)$; *// calculate cost function if applying the sequence of optimal actions of length h*
  if $J_k(s_{i-1}) - J_k(s_i) > \Delta_k$ **then**
    $\Delta_k \leftarrow J_k(s_i)/(m-i)$; *// new level-threshold*

Algorithm 3. Resample $(\{\mathcal{M}_k^h, J_k(s_i)\}_{k=1}^n)$

$\mathcal{I} \leftarrow$ Sort ascending $\mathcal{M}_k^h$ by their current costs; *// find indexes of MDPs whose costs are below the median among all the simulations*
for $k = 1$ **to** n **do**
  if $k \notin \mathcal{I}$ **then**
    Sample r uniformly at random from $\mathcal{I}$; $\mathcal{M}_k \leftarrow \mathcal{M}_r^h$;
  else
    $\mathcal{M}_k \leftarrow \mathcal{M}_k^h$; *// Keep more successful MDPs unchanged*

Algorithm 4. ARES

Input : $\mathcal{M}, \varphi, p_{start}, p_{inc}, p_{max}, h_{max}, m, n$
Output: $\{\boldsymbol{a}^i\}_{1 \leqslant i \leqslant m}$ *// synthesized optimal plans*

Initialize $\ell_0 \leftarrow \inf$; $\{J_k(s_0)\}_{k=1}^n \leftarrow \inf$; $p \leftarrow p_{start}$; $i \leftarrow 1$; $h \leftarrow 1$; $\Delta_k \leftarrow 0$;

while $(\ell_i > \varphi) \vee (i < m)$ **do**
  // find and apply best actions with RPH h
  $[\{\boldsymbol{a}_k^h, J_k(s_i), \mathcal{M}_k^h\}_{k=1}^n] \leftarrow$`Simulate(`$\mathcal{M}, h, i, \{\Delta_k, J_k(s_{i-1})\}_{k=1}^n$`)`;
  $\widehat{J}_1 \leftarrow sort(J_1(s_i), \ldots, J_n(s_i))$; *// find minimum cost among all simulations*
  if $\ell_{i-1} - \widehat{J}_1 > \Delta_1$ **then**
    $\ell_i \leftarrow \widehat{J}_1$; *// new level has been reached*
    $i \leftarrow i + 1$; $h \leftarrow 1$; $p \leftarrow p_{start}$; *// reset adaptive parameters*
    $\{\mathcal{M}_k\}_{k=1}^n \leftarrow$ `Resample(`$\{\mathcal{M}_k^h, J_k(s_i)\}_{k=1}^n$`)`;
  else
    if $h < h_{max}$ **then**
      $h \leftarrow h + 1$; *// improve time exploration*
    else
      if $p < p_{max}$ **then**
        $h \leftarrow 1$; $p \leftarrow p + p_{inc}$; *// improve space exploration*
      else
        break;

Take a clone in the state with minimum cost $\ell_i = J(s_i^*) \leqslant \varphi$ at the last level i;
foreach i **do**
  $\{s_{i-1}^*, \boldsymbol{a}^i\} \leftarrow Pre(s_i^*)$; *// find predecessor and corresponding action*

We assess the rate of success in generating optimal plans in form of an (ε, δ)-approximation scheme, for the desired error margin ε, and confidence ratio $1-\delta$. Moreover, we can use the state-action pairs generated during the assessment (and possibly some additional new plans) to construct an explicit (tabled) optimal policy, modulo some topological equivalence. Given enough memory, one can use this policy in real time, as it only requires a table lookup.

To experimentally validate our approach, we have applied ARES to the problem of V-formation in a flock of birds (with a deterministic MDP) as described in [37,38]. The cost function to be optimized is defined as a weighted sum of

the (flock-wide) clear-view (CV), velocity alignment (VA), and upwash benefit (UB) metrics. CV means that no bird's frontal view is obstructed by a flockmate, whereas VA is essential for maintaining formation (like V-formation) once it has been reached. Regarding UB, by flapping its wings, a bird generates a trailing upwash region off its wing tips; a bird flying in this region (left or right) can save energy. Note that in V-formation, all birds but one (the leader) enjoy UB.

We ran ARES on 8000 initial states chosen uniformly at random, such that they are packed closely enough to benefit from UB, but not too close to colliding. We succeeded to generate a V-formation 95% of the time, with an error margin of 0.05 and a confidence ratio of 0.99 computed using SMC. These statistics improve significantly if we consider all generated states as independent initial states. The fact that each state within a plan is independent of the states in all other plans allows us to do this.

6 Importance Splitting for Optimal Control

As in Sect. 5 where our focus was on optimal planning, in this section, we show how SMC-style importance splitting can be brought to bear on the problem of optimal control. In particular, we present the AMPC algorithm, short for level-based Adaptive-horizon Model-Predictive Control. We also consider stochastic two-player reachability games on MDPs (between a controller and an attacker) and demonstrate resiliency of AMPC control in this setting. As in Sect. 5, we consider the problem of V-formation in a flock of B birds as a motivating example.

6.1 Adaptive-Horizon Model-Predictive Control

The AMPC algorithm performs step-by-step control of a given MDP $\mathcal{M}$ by looking h steps ahead and predicting the next best state to move to [35]. We use PSO to identify the potentially best actions $\boldsymbol{a}^h$ in the current state achieving the optimal value of the fitness function in the next state. The fitness function, $\texttt{Fitness}(\mathcal{M}, \boldsymbol{a}^h, h)$ of $\boldsymbol{a}^h$ is defined as the minimum fitness metric J obtained within h steps by applying $\boldsymbol{a}^h$ on $\mathcal{M}$. Formally, we have

$$\texttt{Fitness}(\mathcal{M}, \boldsymbol{a}^h, h) = \min_{1 \leq \tau \leq h} J(s^{\tau}_{\boldsymbol{a}^h}) \tag{8}$$

where $s^{\tau}_{\boldsymbol{a}^h}$ is the state after apply the τth action of $\boldsymbol{a}^h$ on $\mathcal{M}$. For horizon h, PSO searches for the best sequence of 2-dimensional acceleration vector of length h, thus having $p = 2Bh$ parameters to be optimized. The number of particles used in PSO is proportional to the number of parameters, i.e., $p = 2\beta Bh$.

The pseudocode for the AMPC algorithm is given in Algorithm 5. A novel feature of AMPC is that, unlike classical MPC that uses a fixed horizon h, AMPC adaptively chooses an h depending on whether it is able to reach a fitness value that is lower than the current fitness by our chosen quanta $\Delta_i, \forall\, i \in \{0, \ldots, m\}$.

AMPC is hence an adaptive MPC procedure that uses level-based horizons introduced in Sect. 5, which was in turn inspired by the importance-splitting technique introduced in Sect. 4.2. It employs PSO to identify the potentially

Algorithm 5. AMPC: Adaptive Model-Predictive Control

Input : $\mathcal{M}, \varphi, h_{max}, m, B, \texttt{Fitness}$
Output: $\{\boldsymbol{a}^i\}_{1 \leq i \leq m}$ *// optimal control sequence*

Initialize $\ell_0 \leftarrow J(s_0)$; $\widehat{J} \leftarrow \inf$; $p \leftarrow 2\beta Bh$; $i \leftarrow 1$; $h \leftarrow 1$; $\Delta_0 \leftarrow (\ell_0 - \varphi)/m$;

while $(\ell_{i-1} > \varphi) \wedge (i < m)$ **do**
  // find and apply first best action out of the horizon sequence of length h
  $[\boldsymbol{a}^h, \widehat{J}] \leftarrow \texttt{particleswarm(Fitness}, \mathcal{M}, p, h\texttt{)}$;
  if $\ell_{i-1} - \widehat{J} > \Delta_i \vee h = h_{max}$ **then**
    // if a new level or the maximum horizon is reached
    $\boldsymbol{a}^i \leftarrow \boldsymbol{a}^h_1$; $\mathcal{M} \leftarrow \mathcal{M}^{\boldsymbol{a}^i}$; *// apply the action and move to the next state*
    $\ell_i \leftarrow J(s(\mathcal{M}))$; *// update ℓ_i with the fitness of the current state*
    $\Delta_i \leftarrow \ell_i/(m-i)$; *// update the threshold on reaching the next level*
    $i \leftarrow i+1$; $h \leftarrow 1$; $p \leftarrow 2\beta Bh$; *// update parameters*
  else
    $h \leftarrow h+1$; $p \leftarrow 2\beta Bh$; *// increase the horizon*

best next actions. If the chosen actions improve (decrease) the fitness of the next state $J(s_{k+h})$, $\forall\ k \in \{0, \ldots, m \cdot h_{max}\}$, in comparison to the fitness of the previous state $J(s_k)$ by the predefined Δ_i, the controller considers these actions to be worthy of an optimal solution.

In this case, the controller applies the actions to each agent (bird) and transitions to the next state of the MDP. The threshold Δ_i determines the next level $\ell_i = J(s_{k+\widehat{h}})$ of the algorithm, where $\widehat{h} \leq h$ is the horizon with the best fitness. The prediction horizon h is increased iteratively if the fitness has not been decreased enough. Upon reaching a new level, the horizon is reset to one (see Algorithm 5). Having the horizon $\widehat{h} > 1$ means it will take multiple transitions in the MDP in order to reach a solution with improved fitness. However, when finding such a solution with $\widehat{h} > 1$, we only apply the first action to transition the MDP to the next state. This is explained by the need to allow the other player (environment or an adversary) to apply their action before we obtain the actual next state. If no new level is reached within h_{max} horizons, the first action of the best $\boldsymbol{a}^h$ using horizon h_{max} is applied.

The dynamic threshold Δ_i is defined as in [30]. Its initial value Δ_0 is obtained by dividing the fitness range to be covered into m equal parts, that is, $\Delta_0 = (\ell_0 - \ell_m)\,/\,m$, where $\ell_0 = J(s_0)$ and $\ell_m = \varphi$. Subsequently, Δ_i is determined by the previously reached level ℓ_{i-1}, as $\Delta_i = \ell_{i-1}/(m - i + 1)$. This way AMPC advances only if $\ell_i = J(s_{k+\widehat{h}})$ is at least Δ_i apart from $\ell_{i-1} = J(s_k)$.

This approach allows us to force PSO to escape from a local minimum, even if this implies passing over a fitness-function ridge (see also Fig. 7(left), by gradually increasing the exploration horizon h. We assume that the MDP is controllable and that the set G of goal states is nonempty, which means that from any state, it is possible to reach a state whose fitness decreased by at least Δ_i. Algorithm 5 presents our approach.

Theorem 1 (AMPC Convergence). *Let $\mathcal{M} = (S, A, T, J)$ be an MDP with a positive and continuous fitness function J, and let $G \subset S$ be a nonempty set of target states with $G = \{s \mid J(s) < \varphi\}$. If the transition relation T is controllable with actions in A, then there is a finite maximum horizon h_{max} and a finite number of execution steps m such that AMPC is able to find a sequence of actions $a_1, \ldots, a_m$ that brings a state in S to a state in G with probability one.*

Proof. In each (macro-)step of horizon length h, from level $\ell_{i-1} = J(s_k)$ to level $\ell_i = J(s_{k+\widehat{h}})$, AMPC decreases the distance to φ by $\Delta_i \geq \Delta$, where $\Delta > 0$ is fixed by the number of steps m chosen in advance. Hence, AMPC converges to a state in G in a finite number of steps for a properly chosen m. AMPC is able to decrease the fitness in a macro step by Δ_i by the controllability assumption and the fairness assumption about the PSO algorithm. Since AMPC is a randomized algorithm, the result is probabilistic.

Note that AMPC is a general procedure that performs adaptive MPC using PSO for dynamical systems that are controllable, come with a fitness metric, and have at least one optimal solution.

6.2 Resiliency of the AMPC Algorithm

Inspired by the emerging problem of CPS security, we introduced the concept of *controller-attacker games* [35]. A controller-attacker game is a two-player stochastic game, where the two players, a controller and an attacker, have antagonistic objectives. A controller-attacker game is formulated in terms of an MDP, with the controller and the attacker jointly determining the transition probabilities.

Definition 4. *Let $\mathcal{M} = (S, A, T, J, I)$ be an MDP. A **randomized strategy** σ over $\mathcal{M}$ is a function of the form $\sigma : S \mapsto PD(A)$, where $PD(A)$ is the set of probability distributions over A. That is, σ takes a state s and returns an action consistent with the probability distribution $\sigma(s)$.*

Definition 5. *A **controller-attacker game** is an MDP $\mathcal{M} = (S, A, T, J, I)$ with $A = C \times D$, where C and D are action sets of the controller and the attacker, respectively. The transition probability $T(s, c \times d, s')$ is jointly determined by actions $c \in C$ and $d \in D$.*

We also introduced a class of controller-attacker games we call V-formation games, where the goal of the controller is to maneuver the plant (a simple model of flocking dynamics) into a V-formation, and the goal of the attacker is to prevent the controller from doing so.

Let $\boldsymbol{x}_i(t), \boldsymbol{v}_i(t), \boldsymbol{a}_i(t)$, and $\boldsymbol{d}_i(t)$ respectively denote the position, velocity, acceleration, and displacement of the i-th bird at time t, $1 \leqslant i \leqslant B$. The behavior of bird i in discrete time is modeled as follows:

$$\begin{aligned} \boldsymbol{x}_i(t+1) &= \boldsymbol{x}_i(t) + \boldsymbol{v}_i(t+1) + \boldsymbol{d}_i(t) \\ \boldsymbol{v}_i(t+1) &= \boldsymbol{v}_i(t) + \boldsymbol{a}_i(t) \end{aligned} \tag{9}$$

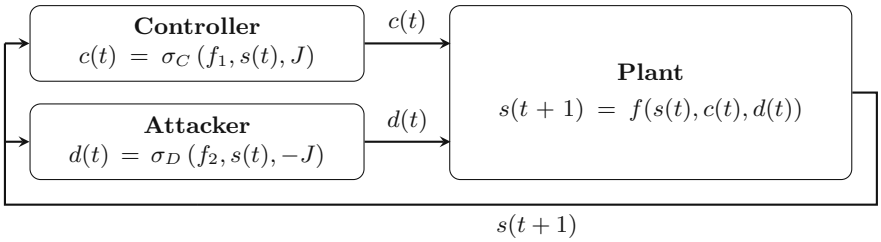

Fig. 8. Controller-Attacker Game Architecture. The controller and the attacker use randomized strategies σ_C and σ_D to choose actions $c(t)$ and $d(t)$ based on dynamics $f_1 = f(s(t), c(t), 0)$ and $f_2 = f(s(t), 0, d(t))$, respectively, where $s(t)$ is the state at time t, and f is the dynamics of the plant model. The controller tries to minimize the cost J, while the attacker tries to maximize it.

The next state of the flock is jointly determined by the accelerations and the displacements based on the current state following Eq. 9.

Controllers in V-formation games utilize AMPC, giving them extraordinary power: we prove that under certain controllability conditions, an AMPC controller can attain V-formation with probability 1.

Definition 6. *A **V-formation game** is a controller-attacker game $\mathcal{M} = (S, A, T, J, I)$, where $S = \{s \mid s = \{\boldsymbol{x}_i, \boldsymbol{v}_i\}_{i=1}^{B}\}$ is the set of states for a flock of B birds, $A = C \times D$ with the controller choosing accelerations $\boldsymbol{a} \in C$ and the attacker choosing displacements $\boldsymbol{d} \in D$, T and J are given in Eqs. 9 and 8, respectively.*

We define several classes of attackers, including those that in one move can remove a small number R of birds from the flock, or introduce random displacement (perturbation) into the flock dynamics, again by selecting a small number of victim agents. We consider both *naive attackers*, whose strategies are purely probabilistic, and *AMPC-enabled attackers*, putting them on par strategically with the controller. The architecture of a V-formation game with an AMPC-enabled attacker is shown in Fig. 8.

While an AMPC-enabled controller is expected to win every game with probability 1, in practice, it is *resource-constrained*: its maximum prediction horizon and the maximum number of execution steps are fixed in advance. Under these conditions, an attacker has a much better chance of winning a V-formation game.

In Sect. 5, we presented a procedure for synthesizing plans (sequences of actions) that take an MDP to a desired set of states (defining a V-formation). The procedure adaptively varied the settings of various parameters of an underlying optimization routine. Since we did not consider any adversary or noise, there was no need for a control algorithm. Here we consider V-formation in the presence of attacks, and hence we developed a generic adaptive control procedure, AMPC, and evaluate its resilience to attacks.

Our extensive performance evaluation of V-formation games uses statistical model checking to estimate the probability that an attacker can thwart the

controller. Our results show that for the bird-removal game with 1 bird being removed, the controller almost always wins (restores the flock to a V-formation). When 2 birds are removed, the game outcome critically depends on which two birds are removed. For the displacement game, our results again demonstrate that an intelligent attacker, i.e. one that uses AMPC in this case, significantly outperforms its naive counterpart that randomly carries out its attack.

Traditional feedback control is, by design, resilient to noise, and also certain kinds of attacks; as our results show, however, it may not be resilient against smart attacks. Adaptive-horizon control helps to guard against a larger class of attacks, but it can still falter due to limited resources. Our results also demonstrate that statistical model checking represents a promising approach toward the evaluation of CPS resilience against a wide range of attacks.

7 Conclusions

The field of Statistical Model Checking (SMC) is now more than 15 years old, and has experienced significant theoretical and practical development during this time. In this chapter, we have presented a review of SMC as an efficient technique for the model checking of stochastic systems, and presented three algorithms representing both the quantitative and qualitative versions of SMC. We also discussed one of the major challenges facing the SMC approach, namely the treatment of rare events. Taking advantage of sequential Monte Carlo methods, we presented efficient procedures for rare-event probability estimation, and illustrated their utility via our implementation in Plasma. We further demonstrated the applicability of the SMC-inspired rare-event approach to tackling plan- and control-synthesis problems for stochastic systems. Looking forward, there are a wealth of challenges facing the SMC community, such as the verification of cyber-physical systems, where SMC can play a crucial role in developing robust and efficient algorithms.

References

1. Baier, C., Haverkort, B.R., Hermanns, H., Katoen, J.-P.: Model-checking algorithms for continuous-time Markov chains. IEEE Trans. Softw. Eng. **29**(6), 524–541 (2003)
2. Basu, A., Bensalem, S., Bozga, M., Caillaud, B., Delahaye, B., Legay, A.: Statistical abstraction and model-checking of large heterogeneous systems. In: Hatcliff, J., Zucca, E. (eds.) FMOODS/FORTE - 2010. LNCS, vol. 6117, pp. 32–46. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13464-7_4
3. Basu, A., Bensalem, S., Bozga, M., Delahaye, B., Legay, A., Sifakis, E.: Verification of an AFDX infrastructure using simulations and probabilities. In: Barringer, H., et al. (eds.) RV 2010. LNCS, vol. 6418, pp. 330–344. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-16612-9_25
4. Boyer, B., Corre, K., Legay, A., Sedwards, S.: PLASMA-lab: a flexible, distributable statistical model checking library. In: Joshi, K., Siegle, M., Stoelinga, M., D'Argenio, P.R. (eds.) QEST 2013. LNCS, vol. 8054, pp. 160–164. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40196-1_12

5. Cérou, F., Del Moral, P., Furon, T., Guyader, A.: Sequential Monte Carlo for rare event estimation. Stat. Comput **22**, 795–808 (2012)
6. Cérou, F., Guyader, A.: Adaptive multilevel splitting for rare event analysis. Stoch. Anal. Appl. **25**, 417–443 (2007)
7. Ciesinski, F., Baier, C.: Liquor: a tool for qualitative and quantitative linear time analysis of reactive systems. In: Proceedings of 3rd International Conference on Quantitative Evaluation of Systems (QEST), pp. 131–132. IEEE (2006)
8. Ciesinski, F., Größer, M.: On probabilistic computation tree logic. In: Baier, C., Haverkort, B.R., Hermanns, H., Katoen, J.-P., Siegle, M. (eds.) Validation of Stochastic Systems. LNCS, vol. 2925, pp. 147–188. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24611-4_5
9. Courcoubetis, C., Yannakakis, M.: The complexity of probabilistic verification. J. ACM **42**(4), 857–907 (1995)
10. David, A., Larsen, K.G., Legay, A., Mikučionis, M., Wang, Z.: Time for statistical model checking of real-time systems. In: Gopalakrishnan, G., Qadeer, S. (eds.) CAV 2011. LNCS, vol. 6806, pp. 349–355. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22110-1_27
11. Del Moral, P.: Feynman-Kac Formulae: Genealogical and Interacting Particle Systems with Applications. Probability and Its Applications. Springer, New York (2004). https://doi.org/10.1007/978-1-4684-9393-1
12. Doucet, A., de Freitas, N., Gordon, N.: Sequential Monte Carlo Methods in Practice. Springer, New York (2001). https://doi.org/10.1007/978-1-4757-3437-9
13. Garca, C.E., Prett, D.M., Morari, M.: Model predictive control: theory and practice - a survey. Automatica **25**(3), 335–348 (1989)
14. Gimbert, H.: Pure stationary optimal strategies in Markov decision processes. In: Thomas, W., Weil, P. (eds.) STACS 2007. LNCS, vol. 4393, pp. 200–211. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-70918-3_18
15. Glasserman, P., Heidelberger, P., Shahabuddin, P., Zajic, T.: Multilevel splitting for estimating rare event probabilities. Oper. Res. **47**(4), 585–600 (1999)
16. Grosu, R., Smolka, S.A.: Monte Carlo model checking. In: Halbwachs, N., Zuck, L.D. (eds.) TACAS 2005. LNCS, vol. 3440, pp. 271–286. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-31980-1_18
17. Younes, H.L.S., Clarke, E.M., Zuliani, P.: Statistical verification of probabilistic properties with unbounded until. In: Davies, J., Silva, L., Simao, A. (eds.) SBMF 2010. LNCS, vol. 6527, pp. 144–160. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19829-8_10
18. Havelund, K., Roşu, G.: Synthesizing monitors for safety properties. In: Katoen, J.-P., Stevens, P. (eds.) TACAS 2002. LNCS, vol. 2280, pp. 342–356. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-46002-0_24
19. Jansen, D.N., Katoen, J.-P., Oldenkamp, M., Stoelinga, M., Zapreev, I.: How fast and fat is your probabilistic model checker? An experimental performance comparison. In: Yorav, K. (ed.) HVC 2007. LNCS, vol. 4899, pp. 69–85. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-77966-7_9
20. Jegourel, C., Legay, A., Sedwards, S.: Importance splitting for statistical model checking rare properties. In: Sharygina, N., Veith, H. (eds.) CAV 2013. LNCS, vol. 8044, pp. 576–591. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39799-8_38
21. Jegourel, C., Legay, A., Sedwards, S.: An effective heuristic for adaptive importance splitting in statistical model checking. In: Margaria, T., Steffen, B. (eds.) ISoLA 2014. LNCS, vol. 8803, pp. 143–159. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45231-8_11

22. Jégourel, C., Legay, A., Sedwards, S.: Command-based importance sampling for statistical model checking. Theoret. Comput. Sci. **649**, 1–24 (2016)
23. Kahn, H.: Stochastic (Monte Carlo) attenuation analysis. Technical report P-88, Rand Corporation, July 1949
24. Kahn, H.: Random sampling (Monte Carlo) techniques in neutron attenuation problems. Nucleonics **6**(5), 27 (1950)
25. Kahn, H., Harris, T.E.: Estimation of particle transmission by random sampling. In: Applied Mathematics. Series 12, vol. 5. National Bureau of Standards (1951)
26. Kahn, H., Marshall, A.W.: Methods of reducing sample size in Monte Carlo computations. Oper. Res. **1**(5), 263–278 (1953)
27. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of 1995 IEEE International Conference on Neural Networks, pp. 1942–1948 (1995)
28. Kwiatkowska, M.Z., Norman, G., Parker, D.: Prism 2.0: a tool for probabilistic model checking. In: QEST, pp. 322–323. IEEE (2004)
29. Kwiatkowska, M., Norman, G., Parker, D.: PRISM 4.0: verification of probabilistic real-time systems. In: Gopalakrishnan, G., Qadeer, S. (eds.) CAV 2011. LNCS, vol. 6806, pp. 585–591. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22110-1_47
30. Lukina, A., Esterle, L., Hirsch, C., Bartocci, E., Yang, J., Tiwari, A., Smolka, S.A., Grosu, R.: ARES: adaptive receding-horizon synthesis of optimal plans. In: Legay, A., Margaria, T. (eds.) TACAS 2017. LNCS, vol. 10206, pp. 286–302. Springer, Heidelberg (2017). https://doi.org/10.1007/978-3-662-54580-5_17
31. Okamoto, M.: Some inequalities relating to the partial sum of binomial probabilities. Ann. Inst. Stat. Math. **10**, 29–35 (1959)
32. Rosenbluth, M.N., Rosenbluth, A.W.: Monte Carlo calculation of the average extension of molecular chains. J. Chem. Phys. **23**(2), 356–359 (1955)
33. Sen, K., Viswanathan, M., Agha, G.: Statistical model checking of black-box probabilistic systems. In: Alur, R., Peled, D.A. (eds.) CAV 2004. LNCS, vol. 3114, pp. 202–215. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-27813-9_16
34. Sen, K., Viswanathan, M., Agha, G.: On statistical model checking of stochastic systems. In: Etessami, K., Rajamani, S.K. (eds.) CAV 2005. LNCS, vol. 3576, pp. 266–280. Springer, Heidelberg (2005). https://doi.org/10.1007/11513988_26
35. Tiwari, A., Smolka, S.A., Esterle, L., Lukina, A., Yang, J., Grosu, R.: Attacking the V: on the resiliency of adaptive-horizon MPC. In: D'Souza, D., Narayan Kumar, K. (eds.) ATVA 2017. LNCS, vol. 10482, pp. 446–462. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68167-2_29
36. Wald, A.: Sequential tests of statistical hypotheses. Ann. Math. Stat. **16**(2), 117–186 (1945)
37. Yang, J., Grosu, R., Smolka, S.A., Tiwari, A.: Love thy neighbor: V-formation as a problem of model predictive control. In: LIPIcs-Leibniz International Proceedings in Informatics, vol. 59. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik (2016)
38. Yang, J., Grosu, R., Smolka, S.A., Tiwari, A.: V-formation as optimal control. In: Proceedings of Biological Distributed Algorithms Workshop 2016 (2016)
39. Younes, H.L.S.: Verification and planning for stochastic processes with asynchronous events. Ph.D. thesis, Carnegie Mellon University (2005)
40. Younes, H.L.S.: Error control for probabilistic model checking. In: Emerson, E.A., Namjoshi, K.S. (eds.) VMCAI 2006. LNCS, vol. 3855, pp. 142–156. Springer, Heidelberg (2005). https://doi.org/10.1007/11609773_10

41. Younes, H.L.S., Simmons, R.G.: Probabilistic verification of discrete event systems using acceptance sampling. In: Brinksma, E., Larsen, K.G. (eds.) CAV 2002. LNCS, vol. 2404, pp. 223–235. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45657-0_17
42. Zuliani, P., Baier, C., Clarke, E.M.: Rare-event verification for stochastic hybrid systems. In: Proceedings of 15th ACM International Conference on Hybrid Systems: Computation and Control, HSCC 2012, pp. 217–226. ACM, New York (2012)