On Optimizing Simulation-Based Verification of Cyber-Physical Systems via Statistical Model Checking: a Preliminary Work

Leonardo Picchiami[1,*]

[1]Computer Science Department, Sapienza University of Rome, via Salaria 113, 00198, Italy

Abstract

Exhaustive simulation-based verification of safety/mission-critical systems often requires unviable overall completion times due to the unmanageable number of operational scenarios to be considered. Statistical Model Checking (SMC), instead, entails randomly evaluating scenarios deemed of interest until statistical guarantees are achieved.

In this short paper, we gather on estimating expected value approximations of system-level properties through Adaptive Stopping Algorithms (SAs), a particular class of SMC that learns the number of scenarios to be considered. Our analysis shows that determining upfront the algorithm that will require the fewest samples for the verification task at hand would be greatly beneficial, as it would allow significant reductions in the overall time. However, although qualitative criteria have been proposed to guide the choice of the right algorithm, its actual performance may depend on unknown information (*e.g.*, the actual value to be estimated).

Keywords

Formal verification, Cyber-Physical Systems, Statistical Model Checking, Adaptive Stopping Algorithms, Monte Carlo estimation

1. Introduction

The ever-increasing deployment of industrial systems in safety/mission critical domains, such as smart grids [1, 2, 3, 4], automotive [5, 6], healthcare [7, 8, 9, 10, 11, 12, 13, 14], or avionics [15, 16, 17], among many others, exasperates the need for *intelligent* strategies to certify such systems as Quality Guaranteed (QG) systems [18]. Industrial systems are typically modelled as Cyber-Physical Systems (CPSs) [19], *i.e.*, systems integrating a continuous physical part, *e.g.*, interconnected hardware, with a discrete cyber part, *i.e.*, the software part. In many CPSs, the software part consists of a controller that continuously senses the state of the system and sends commands to the hardware part to meet system-level specifications when the system operates in its operational environment, *i.e.*, in presence of inputs and/or additional uncontrolled events (such as faults, noise signals, collectively referred to as *disturbances*).

The complexity of industry-level safety/mission critical systems typically makes white-box model-based approaches such as symbolic techniques, logic or automata (*e.g.*, those discussed in [20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30]) unviable. In such a context, the system model simulation is the only viable alternative for evaluating the system's behaviour in its environment. It is usually supported by off-the-shelf design tools (*e.g.*, Modelica or Simulink) that enable the mathematical modelling of the physical part of the system through Ordinary Differential Equations (ODEs).

Among the limitations of the simulation-based verification of CPSs, verifying the system under all time sequences of disturbances which can possibly materialize (*operational scenarios*) easily leads to years of experimental campaigns (*e.g.*, see [31, 32]) and it is a key obstacle to overcome.

Statistical Model Checking (SMC), instead, is a Monte Carlo that evaluates randomly-chosen operational scenarios until the desired statistical guarantees are obtained. In particular, we focus on Adaptive Stopping Algorithms (SAs), which compute (ϵ, δ)-approximations of the expected value of

OVERLAY 24: 6th International Workshop on Artificial Intelligence and fOrmal VERification, Logic, Automata, and sYnthesis
*Corresponding author.
picchiami@di.uniroma1.it (L. Picchiami)
https://raise.uniroma1.it (L. Picchiami)
0000-0001-5477-6419 (L. Picchiami)

given Key Performance Indicators (KPIs) using as few as possible *i.i.d.* samples. An (ϵ, δ)-approximation guarantees that, with probability at least $1 - \delta$, the relative error on the expected value of the system KPI is bounded by ϵ.

SAs do not fix the number of required samples in advance but continuously monitor their own progress and gradually ask for new samples until enough knowledge has been accumulated to compute an approximation of the quantity of interest with the sought statistical properties. In other words, SAs are Artificial Intelligence (AI) algorithms that continuously learn when enough samples have been processed to output the estimate according to complex stopping criteria. The fact that such stopping criteria depend on the KPI distribution and take into account the progress of the algorithm is necessary but is not enough in itself to guarantee that the final sample size will not be larger than necessary. This must be proven by the statistical result from which the stopping criterion has been derived, hence the difficulty in coming up with new and useful SAs. On such a basis, we consider two state-off-the-art SAs: the *Approximation Algorithm* ($\mathcal{AA}$) [33] and the *Empirical Bernstein Stopping* (EBStop) [34]. These algorithms are (quasi-) *optimal*, *i.e.*, they are probabilistically guaranteed to terminate after a number of samples within a constant factor from the minimum number of samples theoretically needed to estimate the quantity of interest. In particular, $\mathcal{AA}$ is well-known to be optimal, whereas EBStop is quasi-optimal in that sense.

Given the time required to simulate each scenario (from several seconds to minutes) and the huge number of scenarios expected to be needed for these algorithms to converge, selecting the SA requiring the least number of samples for the verification at hand may have a significant impact on the overall completion time, even when parallelization of system model simulations is in place. However, although qualitative criteria have been proposed to guide the choice of the right algorithm, its actual performance may depend on unknown information (*e.g.*, the actual value to be estimated). Hence, this paper explores the benefits and performance gain we would obtain if the right algorithm can be actually determined.

2. State of the art

Statistical Model Checking [35, 36] comprises a set of Monte Carlo–based methods to compute formal (quality-) guarantees for a variety of purposes such as formal verification [37], optimization of control strategies [38] and AI-search algorithms [39, 18]. In the context of CPSs, *white-box* [40] as well as *black-box* [41, 42] methodologies based on system modelling formalisms (*e.g.*, see [43, 44, 45]) have been developed to check satisfaction of formal specifications [46] in many application domains, *e.g.*, biological systems [47, 48], automotive [49, 50] or smart grids [51], among others.

Model-based formal verification via SMC can be performed either through off-the-shelf SMC tools (see, *e.g.*, MultiVeSta [52], PRISM [53], UPPAAL-SMC [54], COSMOS [55], Ymer [56]), or by interfacing simulator- and model-agnostic algorithms (see, *e.g.*, [57]) to existing commercial (*e.g.*, Dymola, Simulink, OpalRT) and non-commercial simulators (*e.g.*, OpenModelica), which makes simulable CPS models available only through such tools (*i.e.*, *black-box*) [58]. In general, SMC-driven Monte Carlo methods hold the promise to reduce the time and cost with respect to simulation campaigns that exhaustively encompass all operational scenarios deemed of interest [59, 60, 61]. Clearly, scaling-up the verification time can be addressed either by improving or developing new algorithms and approaches [62, 63] or designing highly parallel architectures safely distributing simulations among several cores (possibly) over different machines (see, *e.g.*, [64, 65, 66]).

3. Formal Framework

In this section we introduce the mathematical background and the problem statement. We denote with $\mathbb{R}$, $\mathbb{R}_{0+}$, and $\mathbb{R}_{+}$, respectively, the sets of all, non-negative and positive reals. Given sets A an B, A^B denotes the set of functions from B to A.

3.1. System Under Verification, Environment and Key Performance Indicators

In our setting, a (black-box) CPS model is described by three connected components: the System Under Verification (SUV), a stochastic environment and the KPI of interest.

In the following, we treat the SUV as a continuous- or discrete-time black-box input-output deterministic causal dynamical system (see, *e.g.*, [67, 68, 69]) that takes as input an operational scenario defined as a function of controllable and uncontrollable inputs (*input functions*) and produces a time function of the system outputs (*i.e.*, *output functions* or *trajectories*).

Input functions for the SUV are *operational scenarios*, and belong to a finitely parametrizable set of possible scenarios that the system must withstand (stochastic environment), which can be modeled via a finite real- or discrete-valued parameter vector. A probability density function is defined over the system operational environment, representing the likelyhoold that each scenario materializes. Note that an environment is finitely parametrizable when a bijection exists between the environment parameter space and the space of system inputs. In other words, the operational environment translates a parametrization of an operational scenarios into an input function for the SUV.

A KPI encodes a metric measure of properties of interest for which system-level specifications must be satisfied. Since the values of the system KPI is the only output we need from the system, we model its computation within the system model and define it to be the only output of the system. This leads to Definition 1, which defines a black-box CPS model.

Definition 1. *A black-box CPS model $\mathcal{S}$ is a tuple $(\mathcal{H}, W, P)$ where $W \subseteq \mathbb{R}^n$ is the environment parameter space with $n > 0$, $P : W \to [0, 1]$ is a probability density function over W, and $\mathcal{H} : W \to [0, 1]$ is the system I/O function that computes a metric KPI for a parametrization of an operational scenario. Namely, $\mathcal{H}(w)$ defines the values of the system KPI (normalized between 0 and 1) when the SUV is given as input the operational scenario encoded by parameter w.*

3.2. QG-verification problem

In our framework, ensuring that a system is quality guaranteed means establishing whether the expected value of the KPI of interest is below a given threshold T.

Definition 2. *A QG-verification problem is a tuple $(\mathcal{H}, W, P, T)$, where*

- *$(\mathcal{H}, W, P)$ is a black-box CPS model.*
- *T is a specification threshold that the expected value of the KPI must not exceed.*

A solution for the QG-verification problem is an yes/no answer based on establishing whether $\mu = \mathbb{E}_{w \in W}[\mathcal{H}(w)] \leq T$ holds, where $\mathbb{E}_{w \in W}$ is the expectation over the environment parameter space W. As a result, solving the QG-verification problem means computing the expected value μ over the *entire* set of operational scenarios.

4. Estimating KPIs via Statistical Model Checking

4.1. SMC-based QG-verification problem

An exact computation of μ, however, may be impossible since the scenario parameter space could be infinite or, anyway, prohibitive. To address this obstacle, Monte Carlo methods can be used to compute an estimate $\hat{\mu}$ of μ. Whereas empirical Monte Carlo methods do not provide any formal (quality-) guarantees of the computed approximation (and this is an unviable option for safety/mission-critical domains), SMC algorithms in [33, 34] are (ϵ, δ)-approximation algorithms designed to output, with probability at least $1 - \delta$, an estimate $\hat{\mu}$ for which $|\hat{\mu} - \mu| < \mu\epsilon$ holds given $\epsilon, \delta \in (0, 1)$. Namely, $Pr\{(1 - \epsilon)\mu \leq \hat{\mu} \leq (1 + \epsilon)\mu\} \geq 1 - \delta$. From this, we obtain that, with probability at least $1 - \delta$, $\frac{\hat{\mu}}{1+\epsilon} \leq \mu \leq \frac{\hat{\mu}}{1-\epsilon}$ holds. Hence, the QG-verification problem is cast to an SMC-based QG-verification problem, whose solution is a yes/no answer that consists of establishing whether $\frac{\hat{\mu}}{1-\epsilon} \leq T$ holds.

4.2. $\mathcal{AA}$ and EB(G)Stop

According to Definition 1, we restrict our analysis to SAs such that: 1) they output $\hat{\mu} \in (0, 1]$; 2) they use mean and variance to adaptively determine when enough samples have been produced. In particular, we focus on the *Approximation Algorithm* ($\mathcal{AA}$) [33] and the *Empirical Bernstein Stopping* (EBStop) [34].

$\mathcal{AA}$ is a three-phase algorithm that computes an approximation $\hat{\mu}$ of the mean $\mu > 0$ of a non-negative random variable in $[0, 1]$. With $\epsilon, \delta \in (0, 1)$ given as input, the first phase outputs an initial $(\min\{1/2, \sqrt{\epsilon}\}, \delta/3)$-approximation $\tilde{\mu}$ with N_1 samples, which is in turn exploited to provide an approximation $\hat{\rho}$ of the variance ρ with N_2 samples in the second phase. Finally, the third phase uses $\tilde{\mu}$ and $\hat{\rho}$ to compute an (ϵ, δ)-approximation $\hat{\mu}$ of μ. Hence, the overall number of samples required to produce the estimate is $N = N_1 + N_2 + N_3$.

EBStop is a sequential algorithm that outputs an (ϵ, δ)-approximation $\hat{\mu}$ of the expected value $\mu \neq 0$ of a random variable in $[a, b]$ with $\epsilon, \delta \in (0, 1)$ given as input. It uses the Bernstein Inequality, together with the sample mean and variance, to establish when enough samples have been produced and the stopping criterion has been reached. In [34], EBGStop, an improved version of EBStop, is provided. It incorporates the geometric sampling schedule, resulting in a tighter confidence interval. This feature ensures EBGStop reaches the stopping condition after fewer samples than EBStop. Hence, we select EBGStop for our purposes.

In [34], the authors claim that if $\ln(\frac{R}{\mu\epsilon})$ is significantly larger than $1/\delta$, then $\mathcal{AA}$ would outperform EBGStop, where R is the range of the random variable (R is always 1 in our framework), Being, however, a qualitative analysis, it is not possible to select *a priori* the most suitable algorithm for a given verification problem; the KPI distribution is typically *unknown*, and which algorithm takes fewer samples between $\mathcal{AA}$ and EBStop for given ϵ, δ is hard to predict. This is a crucial drawback, especially in the context of simulation-based verification of CPSs where producing each sample requires to numerically simulate the system model under the randomly generated scenarios, and each simulation may take seconds to minutes, depending on the complexity of the system.

5. Experimental results

This section outlines the impact of using $\mathcal{AA}$ and EBGStop for SMC-driven simulation-based verification on a Simulink/Stateflow case study. Our experiments show the benefit (*i.e.*, the reduction in the number of samples and completion time) if we would able to select in advance the algorithm that takes fewer samples at hand for several configuration of ϵ and δ.

5.1. Case Study

We selected the Apollo Lunar Module Autopilot (ALMA) model as an industry-scale system that has been widely adopted in the literature on simulation-based verification (*e.g.*, see [59, 60]). It defines the logic of the phase-plane control algorithm for the autopilot program of the lunar module used in the Apollo 11 mission. The ALMA model is equipped with 3 sensors (yaw, pitch, roll) and 16 reaction jets that actuate a rotation over one or more axes. In every space mission (system simulation), the controller takes as input a request to change the module's attitude and computes which reaction jets need to be activated to achieve the desired rotation. The goal of the system is to successfully perform the rotation and stabilize itself at the new attitude.

We supplied the system with a stochastic environment that injects temporary faults (white noise signals) on yaw, pitch and roll sensors using finite states machines (Stateflow charts). The faults occurrences are chosen according to an exponential distribution whose mean corresponds to a Mean Time Between Failures (MTBF) of 4 s, whereas the fault recovery happens after a duration determined by an exponential distribution with a mean of 1 s.

The system KPI quantifies the outcome of a given space mission in terms of the average module's attitude error over the entire simulation. Let $a_s(t)$ be the system attitude at time t, and let R_s be the target attitudes for the axes $s \in \{y, p, r\}$ (*i.e.*, yaw, pitch and roll). We define the error attitudes as

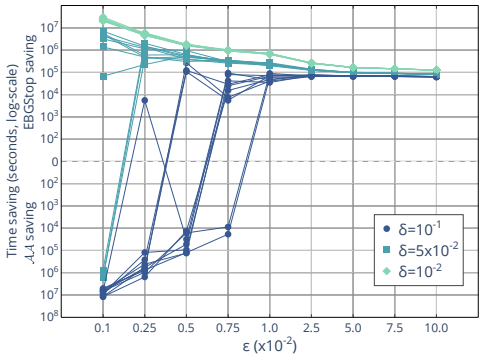

Figure 1: Sample saving (log-scale Y-axis).

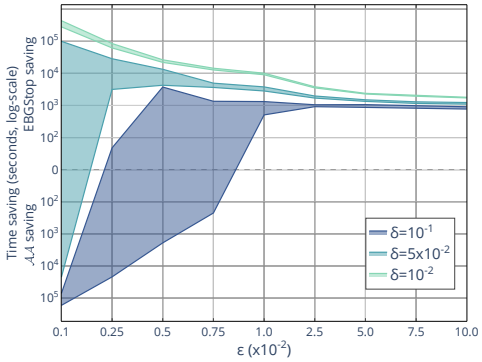

Figure 2: Time saving (log-scale Y-axis) with 512 simulators.

$e_s(t) = |a_s(t) - R_s|$ for each axis s. We define our KPI on top of such indicators, that is, the Normalised Mean Attitude Module Error (NMAME) at time t defined as $\frac{1}{2\pi t}\int_0^t \max\{e_y(\tau), e_p(\tau), e_r(\tau)\}d\tau$. It represents the average over time of the maximum error on the three axes. Note that 2π is a normalization factor since the axes values within a rotation range in $[0, 2\pi]$.

5.2. Results

We conducted simulation campaigns with a horizon $h = 60$ seconds to compute (ϵ, δ)-approximations for the ALMA model for 9 different values of ϵ (ranging from 1×10^{-3} to 1×10^{-1}) and 3 values for δ ($1 \times 10^{-2}, 5 \times 10^{-2}, 1 \times 10^{-1}$). Each experiment has been repeated 10 times, with 10 different sequences of $i.i.d.$ scenarios. Simulating the system model under each scenario took on average 7.1654 on each of our identical machines (2 AMD EPYC 7301 CPUs with 32 cores, and 256 GB RAM). To avoid simulating the same sequence of scenarios multiple times, we instrumented our tool to store both simulation time and KPI values for each scenario in a centralized database. This allowed us to accurately compute the completion time of our algorithm under different values for ϵ and δ, when run starting from each of the 10 random seeds.

Figure 1 and Figure 2 show, respectively, the absolute saving in the number of samples and completion time for several values of ϵ and δ on the ALMA system. If an experiment lands on the upper side of the (log-scale) y-axis, EBGStop is the winner, *i.e.*, it takes the lower number of samples (Figure 1) or time (Figure 2) to output an (ϵ, δ)-approximation of the KPI; otherwise, $\mathcal{AA}$ wins. For each δ, Figure 1 describes the behaviour of all *i.i.d.* sequences of scenarios (random seeds) obtained for all values of ϵ. It shows a sample saving ranging from 10^3 to 10^7 samples (*i.e.*, simulations) up to a maximum of 31 300 400 with a relative reduction as large as 69.13%. Figure 2, instead, quantifies the actual benefit we would obtain on a highly parallel architecture with 512 simulators. For each value of δ, it illustrates the reduction of the verification time (across the 10 *i.i.d.* random sequences of scenarios) obtained when using the algorithm requiring fewer samples. The figure shows that although both algorithms are known to be (quasi)-optimal (in the sense of Section 1), if we were able to select upfront the algorithm requiring the fewest samples for the actual verification task at hand, we would have cut down completion times up to 68.41% and up to more than 10^5 seconds (*i.e.*, more than 1 day of computation).

Figure 1 and Figure 2 delineate that several existing factors may determine the winning algorithm between $\mathcal{AA}$ and EBGStop. Based on the qualitative analysis in [34], $\ln(\frac{R}{\mu\epsilon})$ must be significantly larger than (*i.e.*, ideally several orders of magnitude greater) than $1/\delta$ so that $\mathcal{AA}$ would outperform EBGStop. However, if we consider $\mu = \frac{0.0128}{1-\epsilon}$, where 0.0128 is the average $\hat{\mu}$ estimated with the lowest ϵ and δ, we can draw different observations. For example, with $\epsilon = 10^{-3}$ and $\delta = 10^{-1}$, $\mathcal{AA}$ always outperforms EBGStop, where $\ln(\frac{R}{\mu\epsilon}) = 11.26$ is not significantly larger than $1/\delta = 10$. With $\epsilon = 5 \times 10^{-3}$ and $\delta = 10^{-1}$, EBGStop outperforms $\mathcal{AA}$ and vice versa according to the *i.i.d.* sequence of operational scenarios taken into account for the estimate with $\ln(\frac{R}{\mu\epsilon}) = 9.65$ smaller than $1/\delta = 10$. This introduces the sequence of scenarios as an additional source of uncertainty for a priori selecting

the fastest (ϵ, δ)-approximation algorithm, which makes such a prediction unviable in practice.

6. Conclusion

In this short paper, we focused on optimizing the model-based verification of CPSs through SAs. We observed a substantial reduction in the verification cost if the most suitable algorithm were used. However, our preliminary results suggest that predicting such an algorithm for SMC-driven verification of CPSs is an obstacle to overcome.

We are currently working on developing solutions to assist the user in selecting the most suitable algorithm to formally verify CPSs in safety/mission-critical domains. Indeed, in very recent work [70], we propose an approach that runs in parallel a finite set of SAs in a similar fashion to AI-based techniques such as *ensemble learning*. This approach is particularly effective if the numerical simulation of the system model is the the slowest step in the verification pipeline, as is the case for complex CPSs.

Acknowledgments

This work was partially supported by: Italian Ministry of University and Research under grant "Dipartimenti di eccellenza 2018–2022" of the Department of Computer Science of Sapienza University of Rome; INdAM "GNCS Project 2023"; Sapienza University projects RG12117A8B393BDC, RG11916B892E54DB, RG120172B9329D33, RM120172B9F35634, RG123188B482D2D9; Lazio POR FESR projects E84G20000150006, F83G17000830007; Project funded under the National Recovery and Resilience Plan (NRRP), Mission 4 Component 2 Investment 1.5 - Call for tender No. 3277 of 30 December 2021 of the Italian Ministry of University and Research funded by the European Union – NextGenerationEU, award number: project ECS 0000024 Rome Technopole, Concession Decree No. 1051 of 23 June 2022 adopted by the Italian Ministry of University and Research, CUP B83C22002820006, Rome Technopole.

References

[1] B. Hayes, I. Melatti, T. Mancini, M. Prodanovic, E. Tronci, Residential demand management using individualised demand aware price policies, IEEE Trans. Smart Grid 8 (2017). doi:10.1109/TSG.2016.2596790.

[2] T. Mancini, F. Mari, I. Melatti, I. Salvo, E. Tronci, J. Gruber, B. Hayes, M. Prodanovic, L. Elmegaard, Demand-aware price policy synthesis and verification services for smart grids, in: SmartGridComm 2014, IEEE, 2014. doi:10.1109/SmartGridComm.2014.7007745.

[3] I. Melatti, F. Mari, T. Mancini, M. Prodanovic, E. Tronci, A two-layer near-optimal strategy for substation constraint management via home batteries, IEEE Trans. Ind. Elect. 69 (2022). doi:10.1109/TIE.2021.3102431.

[4] T. Mancini, F. Mari, I. Melatti, I. Salvo, E. Tronci, J. Gruber, B. Hayes, M. Prodanovic, L. Elmegaard, User flexibility aware price policy synthesis for smart grids, in: DSD 2015, IEEE, 2015. doi:10.1109/DSD.2015.35.

[5] D. Goswami, R. Schneider, A. Masrur, M. Lukasiewycz, S. Chakraborty, H. Voit, A. Annaswamy, Challenges in automotive cyber-physical systems design, in: SAMOS 2012, IEEE, 2012. doi:10.1109/SAMOS.2012.6404199.

[6] S. Chakraborty, M. Al Faruque, W. Chang, D. Goswami, M. Wolf, Q. Zhu, Automotive cyber–physical aystems: A tutorial introduction, IEEE Des.&Test 33 (2016). doi:10.1109/MDAT.2016.2573598.

[7] M. Hengartner, T. Kruger, K. Geraedts, E. Tronci, T. Mancini, F. Ille, M. Egli, S. Roeblitz, R. Ehrig, L. Saleh, K. Spanaus, C. Schippert, Y. Zhang, B. Leeners, Negative affect is unrelated to fluctuations in hormone levels across the menstrual cycle: Evidence from a multisite observational study across two successive cycles, J. Psycho. Res. 99 (2017). doi:10.1016/j.jpsychores.2017.05.018.

[8] B. Leeners, T. Krüger, K. Geraedts, E. Tronci, T. Mancini, M. Egli, S. Röblitz, L. Saleh, K. Spanaus, C. Schippert, Y. Zhang, F. Ille, Associations between natural physiological and supraphysiological estradiol levels and stress perception, Front. Psycol. 10 (2019). doi:10.3389/fpsyg.2019.01296.
[9] F. Maggioli, T. Mancini, E. Tronci, SBML2Modelica: Integrating biochemical models within open-standard simulation ecosystems, Bioinformatics 36 (2020). doi:10.1093/bioinformatics/btz860.
[10] T. Mancini, F. Mari, A. Massini, I. Melatti, I. Salvo, S. Sinisi, E. Tronci, R. Ehrig, S. Röblitz, B. Leeners, Computing personalised treatments through in silico clinical trials. A case study on downregulation in assisted reproduction, in: RCRA 2018, volume 2271 of *CEUR W.P.*, CEUR, 2018.
[11] S. Sinisi, V. Alimguzhin, T. Mancini, E. Tronci, F. Mari, B. Leeners, Optimal personalised treatment computation through in silico clinical trials on patient digital twins, Fundam. Inform. 174 (2020). doi:10.3233/FI-2020-1943.
[12] M. Esposito, L. Picchiami, Simulation-based synthesis of personalised therapies for colorectal cancer, in: OVERLAY 2021, volume 2987 of *CEUR W.P.*, CEUR, 2021.
[13] M. Esposito, L. Picchiami, Intelligent search for personalized cancer therapy synthesis: an experimental comparison, in: IPS/RCRA 2021, volume 3065 of *CEUR W.P.*, CEUR, 2021.
[14] M. Esposito, L. Picchiami, A comparative study of AI search methods for personalised cancer therapy synthesis in COPASI, in: AI*IA 2021, Rev. Sel. Papers., volume 13196 of *LNCS*, Springer, 2022. doi:10.1007/978-3-031-08421-8_44.
[15] L. Planke, Y. Lim, A. Gardi, R. Sabatini, T. K., N. Ezer, A cyber-physical-human system for one-to-many uas operations: Cognitive load analysis, Sensors 20 (2020). doi:10.3390/s20195467.
[16] Z. Wu, N. Huang, X. Zheng, X. Li, Cyber-physical avionics systems and its reliability evaluation, in: IEEE CYBER 2014, IEEE, 2014. doi:10.1109/CYBER.2014.6917502.
[17] K. Sampigethaya, R. Poovendran, Aviation cyber-physical systems: Foundations for future aircraft and air transport, Proc. IEEE 101 (2013). doi:10.1109/JPROC.2012.2235131.
[18] M. Esposito, T. Mancini, E. Tronci, Optimizing fault-tolerant quality-guaranteed sensor deployments for UAV localization in critical areas via computational geometry, IEEE Trans. Sys. Man Cyb.: Sys.'s 54 (2024). doi:10.1109/TSMC.2023.3327432.
[19] R. Alur, Principles of Cyber-Physical Systems, MIT, 2015.
[20] G. Della Penna, B. Intrigila, I. Melatti, M. Minichino, E. Ciancamerla, A. Parisse, E. Tronci, M. Venturini Zilli, Automatic verification of a turbogas control system with the Murphi verifier, in: HSCC 2003, volume 2623 of *LNCS*, Springer, 2003. doi:10.1007/3-540-36580-X_13.
[21] G. Della Penna, B. Intrigila, I. Melatti, E. Tronci, M. Venturini Zilli, Finite horizon analysis of Markov chains with the Murphi verifier, STTT 8 (2006). doi:10.1007/s10009-005-0216-7.
[22] M. Cadoli, T. Mancini, F. Patrizi, SAT as an effective solving technology for constraint problems, in: ISMIS 2006, volume 4203 of *LNCS*, Springer, 2006. doi:10.1007/11875604_61.
[23] M. Cadoli, T. Mancini, Combining relational algebra, SQL, constraint modelling, and local search, TPLP 7 (2007). doi:10.1017/S1471068406002857.
[24] T. Mancini, P. Flener, J. Pearson, Combinatorial problem solving over relational databases: View synthesis through constraint-based local search, in: SAC 2012, ACM, 2012. doi:10.1145/2245276.2245295.
[25] G. Gottlob, G. Greco, T. Mancini, Conditional constraint satisfaction: Logical foundations and complexity, in: IJCAI 2007, 2007.
[26] T. Mancini, M. Cadoli, D. Micaletto, F. Patrizi, Evaluating ASP and commercial solvers on the CSPLib, Constraints 13 (2008). doi:10.1007/s10601-007-9028-6.
[27] L. Bordeaux, M. Cadoli, T. Mancini, CSP properties for quantified constraints: Definitions and complexity, in: AAAI 2005, AAAI, 2005.
[28] T. Mancini, E. Tronci, A. Scialanca, F. Lanciotti, A. Finzi, R. Guarneri, S. Di Pompeo, Optimal fault-tolerant placement of relay nodes in a mission critical wireless network, in: RCRA 2018, volume 2271 of *CEUR W.P.*, CEUR, 2018.
[29] Q. Chen, A. Finzi, T. Mancini, I. Melatti, E. Tronci, MILP, pseudo-boolean, and OMT solvers for optimal fault-tolerant placements of relay nodes in mission critical wireless networks, Fundam.

Inform. 174 (2020). doi:10.3233/FI-2020-1941.
[30] T. Mancini, F. Mari, I. Melatti, I. Salvo, E. Tronci, An efficient algorithm for network vulnerability analysis under malicious attacks, in: ISMIS 2018, Springer, 2018. doi:10.1007/978-3-030-01851-1_29.
[31] T. Mancini, F. Mari, A. Massini, I. Melatti, F. Merli, E. Tronci, System level formal verification via model checking driven simulation, in: CAV 2013, volume 8044 of *LNCS*, Springer, 2013. doi:10.1007/978-3-642-39799-8_21.
[32] T. Mancini, F. Mari, A. Massini, I. Melatti, E. Tronci, System level formal verification via distributed multi-core hardware in the loop simulation, in: PDP 2014, IEEE, 2014. doi:10.1109/PDP.2014.32.
[33] P. Dagum, R. Karp, M. Luby, S. M. Ross, An optimal algorithm for Monte Carlo estimation, SICOMP 29 (2000). doi:10.1137/S0097539797315306.
[34] V. Mnih, C. Szepesvári, J. Audibert, Empirical Bernstein stopping, in: ICML 2008, ACM, 2008. doi:10.1145/1390156.1390241.
[35] G. Agha, K. Palmskog, A survey of statistical model checking, ACM Trans. Model. Comput. Simul. 28 (2018). doi:10.1145/3158668.
[36] A. Pappagallo, A. Massini, E. Tronci, Monte Carlo based statistical model checking of cyber-physical systems: a review, Inf. 11 (2020). doi:10.3390/info11120588.
[37] A. Legay, S. Sedwards, L. Traonouez, Scalable verification of markov decision processes, in: Soft. Engi. and Form. Meth., Springer, 2015. doi:10.1007/978-3-319-15201-1_23.
[38] A. David, D. Du, K. Guldstrand Larsen, A. Legay, M. Mikučionis, Optimizing control strategy using statistical model checking, in: NFM 2013, volume 7871 of *LNCS*, Springer, 2013.
[39] S. Sinisi, V. Alimguzhin, T. Mancini, E. Tronci, B. Leeners, Complete populations of virtual patients for in silico clinical trials, Bioinformatics 36 (2020). doi:10.1093/bioinformatics/btaa1026.
[40] K. Sen, M. Viswanathan, G. Agha, On statistical model checking of stochastic systems, in: CAV 2005, volume 3576 of *LNCS*, Springer, 2005.
[41] K. Sen, M. Viswanathan, G. Agha, "statistical model checking of black-box probabilistic systems", in: CAV 2004, volume 3114 of *LNCS*, Springer, 2004.
[42] H. Younes, Probabilistic verification for “black-box” systems, in: CAV 2005, volume 3576 of *LNCS*, Springer, 2005. doi:10.1007/11513988_25.
[43] H. Younes, R. Simmons, Probabilistic verification of discrete event systems using acceptance sampling, in: CAV 2002, volume 2404 of *LNCS*, Springer, 2002. doi:10.1007/3-540-45657-0_17.
[44] C. Baier, B. Haverkort, H. Hermanns, J.-P. Katoen, Model-checking algorithms for continous-time markov chains, IEEE Trans. Softw. Eng. 29 (2003). doi:10.1109/TSE.2003.1205180.
[45] G. Norman, D. Parker, J. Sproston, Model checking for probabilistic timed automata, Form. Meth. Sys. Des. 43 (2013). doi:10.1007/s10703-012-0177-x.
[46] H. Hansson, B. Jonsson, A logic for reasoning about time and reliability, Form. Asp. Comp. 6 (1994). doi:10.1007/BF01211866.
[47] P. Zuliani, Statistical model checking for biological applications, STTT 17 (2015). doi:10.1007/s10009-014-0343-0.
[48] E. Tronci, T. Mancini, I. Salvo, S. Sinisi, F. Mari, I. Melatti, A. Massini, F. Davi’, T. Dierkes, R. Ehrig, S. Röblitz, B. Leeners, T. Krüger, M. Egli, F. Ille, Patient-specific models from inter-patient biological models and clinical records, in: FMCAD 2014, IEEE, 2014. doi:10.1109/FMCAD.2014.6987615.
[49] E. Clarke, P. Zuliani, Statistical model checking for cyber-physical systems, in: ATVA 2011, volume 11, Springer, 2011.
[50] A. Atallah, G. Hamad, O. Mohamed, Automotive safety verification under temporal failure of adaptive cruise control system using statistical model checking, in: EDiS 2017, IEEE, 2017. doi:10.1109/EDIS.2017.8284024.
[51] J. Martins, A. Platzer, J. Leite, Statistical model checking for distributed probabilistic-control hybrid automata with smart grid applications, in: ICFEM 2011, volume 6991 of *LNCS*, Springer, 2011. doi:10.1007/978-3-642-24559-6_11.
[52] S. Sebastio, A. Vandin, MultiVeStA: Statistical model checking for discrete event simulators, in:

ValueTools 2013, ICST/ACM, 2013. doi:10.4108/icst.valuetools.2013.254377.
[53] M. Kwiatkowska, G. Norman, D. Parker, Prism 4.0: Verification of probabilistic real-time systems, in: CAV 2011, volume 6806 of *LNCS*, Springer, 2011. doi:10.1007/978-3-642-22110-1_47.
[54] P. Bulychev, A. David, K. Larsen, M. Mikučionis, D. Poulsen, A. Legay, Z. Wang, UPPAAL-SMC: Statistical model checking for priced timed automata, EPTCS 85 (2012). doi:10.4204/EPTCS.85.1.
[55] P. Ballarini, B. Barbot, M. Duflot, S. Haddad, N. Pekergin, HASL: a new approach for performance evaluation and model checking from concepts to experimentation, Perf. Eval. 90 (2015). doi:10.1016/j.peva.2015.04.003.
[56] H. Younes, Ymer: A statistical model checker, in: CAV 2005, volume 3576 of *LNCS*, Springer, 2005. doi:10.1007/11513988_43.
[57] R. Grosu, S. Smolka, Monte Carlo model checking, in: TACAS 2005, volume 3440 of *LNCS*, Springer, 2005. doi:10.1007/978-3-540-31980-1_18.
[58] S. Sinisi, V. Alimguzhin, T. Mancini, E. Tronci, Reconciling interoperability with efficient verification and validation within open source simulation environments, Simul. Model. Pract. Theory 109 (2021). doi:10.1016/j.simpat.2021.102277.
[59] T. Mancini, I. Melatti, E. Tronci, Optimising highly-parallel simulation-based verification of cyber-physical systems, IEEE TSE (2023). doi:10.1109/TSE.2023.3298432.
[60] T. Mancini, I. Melatti, E. Tronci, Any-horizon uniform random sampling and enumeration of constrained scenarios for simulation-based formal verification, IEEE TSE 48 (2022). doi:10.1109/TSE.2021.3109842.
[61] T. Mancini, F. Mari, A. Massini, I. Melatti, I. Salvo, E. Tronci, On minimising the maximum expected verification time, Inf. Proc. Lett. 122 (2017). doi:10.1016/j.ipl.2017.02.001.
[62] C. Jegourel, J. Sun, J. Dong, Sequential schemes for frequentist estimation of properties in statistical model checking, ACM Trans. Model. Comput. Simul. 29 (2019). doi:10.1145/3310226.
[63] H. Bu, M. Sun, Clopper-Pearson algorithms for efficient statistical model checking estimation, IEEE TSE 50 (2024). doi:10.1109/TSE.2024.3392720.
[64] T. Mancini, F. Mari, I. Melatti, I. Salvo, E. Tronci, J. Gruber, B. Hayes, L. Elmegaard, Parallel statistical model checking for safety verification in smart grids, in: SmartGridComm 2018, IEEE, 2018. doi:10.1109/SmartGridComm.2018.8587416.
[65] T. Mancini, E. Tronci, I. Salvo, F. Mari, A. Massini, I. Melatti, Computing biological model parameters by parallel statistical model checking, in: IWBBIO 2015, volume 9044 of *LNCS*, Springer, 2015. doi:10.1007/978-3-319-16480-9_52.
[66] M. Esposito, L. Picchiami, Estimation-based verification of cyber-physical systems via statistical model checking, in: HYDRA/RCRA 2022, volume 3281 of *CEUR W.P.*, CEUR, 2022.
[67] E. Sontag, Mathematical Control Theory: Deterministic Finite Dimensional Systems (2nd Ed.), Springer, 1998.
[68] T. Mancini, F. Mari, A. Massini, I. Melatti, E. Tronci, Anytime system level verification via random exhaustive hardware in the loop simulation, in: DSD 2014, IEEE, 2014. doi:10.1109/DSD.2014.91.
[69] T. Mancini, F. Mari, A. Massini, I. Melatti, E. Tronci, Anytime system level verification via parallel random exhaustive hardware in the loop simulation, Microprocessors and Microsystems 41 (2016). doi:10.1016/j.micpro.2015.10.010.
[70] L. Picchiami, M. Parmentier, A. Legay, T. Mancini, E. Tronci, Scaling up statistical model checking of cyber-physical systems via algorithm ensemble and parallel simulations over HPC infrastructures, J. Sys. Soft. 219 (2024). doi:10.1016/j.jss.2024.112238.