DiPS: A Tool for Data-Informed Parameter Synthesis for Markov Chains from Multiple-Property Specifications

Matej Hajnal[1,3](✉), David Šafránek[3], and Tatjana Petrov[1,2]

[1] Department of Computer and Information Sciences, University of Konstanz, Konstanz, Germany
374185@mail.muni.cz

[2] Centre for the Advanced Study of Collective Behaviour, University of Konstanz, 78464 Konstanz, Germany

[3] Systems Biology Laboratory, Faculty of Informatics, Masaryk University, Botanická 68a, 602 00 Brno, Czech Republic

Abstract. We present a tool for inferring the parameters of a Discrete-time Markov chain (DTMC) with respect to properties written in probabilistic temporal logic (PCTL) informed by data observations. The tool combines, in a modular and user-friendly way, the existing methods and tools for parameter synthesis of DTMCs. On top of this, the tool implements several hybrid methods for the exploration of the parameter space based on utilising the intermediate results of parametric model checking – the symbolic representation of properties' satisfaction in the form of rational functions. These methods are combined to support three different parameter exploration methods: (i) optimisation, (ii) parameter synthesis, (iii) Bayesian parameter inference. Each of the available methods makes a different trade-off between scalability and inference quality, which can be chosen by the user depending on the application context. In this paper, we present the implementation, the main features of the tool, and we evaluate its performance on several benchmarks.

1 Introduction

Modelling stochastic dynamical systems such as a biological cell, epidemic spread in a population, or a randomised communication protocol is challenging, especially when parameters are not available, subject to uncertainty, and when exper-

TP's research is supported by the Ministry of Science, Research and the Arts of the state of Baden-Württemberg, the DFG Centre of Excellence 2117 'Centre for the Advanced Study of Collective Behaviour' (ID: 422037984), and AFF (Committee on Research, University of Konstanz), MH's research is supported by Young Scholar Fund (YSF), project no. $P83943018FP430\_/18$, and Max Planck Institute of Animal Behaviour. DŠ's research is supported by the Czech Grant Agency grant no. GA18-00178S. The authors acknowledge Denis Repin, Nhat-Huy Phung, and Stefano Tognazzi for functional testing, feedback, and solving code issues.

P. Ballarini et al. (Eds.): EPEW 2021/ASMTA 2021, LNCS 13104, pp. 79–95, 2021.
https://doi.org/10.1007/978-3-030-91825-5_5

imental data measurements are scarce. *Parameter synthesis* is particularly useful in this context, as it determines the regions of parameter space, for which a high-level property holds. Such high-level property is typically a functional specification (e.g. 'error states are reached with small probability'). Parameter synthesis of discrete-time Markov chains (DTMCs) is supported by several existing tools for probabilistic verification [8,9,12,22]. Most of these implementations specialise in the case when a single qualitative or quantitative property is of interest. In practice, there is an emerging need to reason about multiple properties at the same time. One such situation is when multiple functional properties should be satisfied simultaneously. Another scenario occurs when a functional property (specification) is additionally constrained by properties derived from experimental data, typical for modelling biological systems or in the context of grey-box system verification and testing [1]. For instance, a qualitative summary of experimental observations at a steady-state such as 'a certain group of states is eventually reached as a terminal state with probability greater than a threshold', can be used to additionally constrain the parameter synthesis procedure. Data observations alone can be encoded in the form of multiple temporal properties, e.g. steady-state observations in a chain with more than one bottom strongly connected component (BSCC) [14].

In this paper, we present DiPS[1] – a tool for data-informed parameter synthesis for *parametric discrete time Markov chains* (pMC) from multiple-property specifications. For a single property expressed in Probabilistic Computation Tree Logic (PCTL) [17], the standard parameter synthesis procedures provide a symbolic representation of satisfaction probability in the form of *rational functions*, which will evaluate exactly to the satisfaction probability for that single property in the given chain. We leverage existing tools PRISM [22] and Storm [9] to obtain the rational functions characterising satisfaction probability of each among the multiple properties in the specification and before incorporating threshold constraints available from the data measurements. Resulting rational functions are the cornerstone of the tool as all further analyses are based on them. In the next step, the (experimental) data are used as thresholds for constraining the rational functions, for given confidence level and based on frequentist statistics interpretation. The resulting algebraic constraints are finally employed to explore the parameter space for which the chain behaviour agrees with the observations.

To explore parameter values respecting given specification supported with data, DiPS employs several different methods working with the synthesised algebraic constraints – rational functions and confidence intervals obtained from experimentally observed satisfaction of the specification. The computational workflow utilises the following methods: optimisation, parameter space refinement, parameter space sampling, and Metropolis-Hastings. In Fig. 1, it is shown how these methods are combined in the tool to tackle the complex data-informed specification-driven procedures including optimisation, parameter synthesis, and Bayesian inference. In particular, the tool implements the following tasks:

[1] https://github.com/xhajnal/DiPS.

- marking single points in the parameter space sat (green) or unsat (red) wrt. the algebraic constraints satisfaction [8] (*space sampling*), or
- marking entire regions (hyper-rectangles) in the parameter space safe (green), unsafe (red) wrt. the algebraic constraints satisfaction with SMT solver [8, 11, 19, 20] or interval arithmetics (*space refinement*), or
- identifying a single point in the parameter space with the least distance wrt. data (*optimisation*), or
- identifying a distribution over possible parametrisations based on their relative likelihood wrt. data using Bayesian inference (*Metropolis-Hastings*).
- providing a novel hybrid method that combines Bayesian inference with space refinement (*HMH*),
- facilitating a user-friendly interface allowing to visualise the results and adapt the workflow by combining the tasks above,
- exploring the potential of the methods for efficient parallel processing on a multi-core hardware.

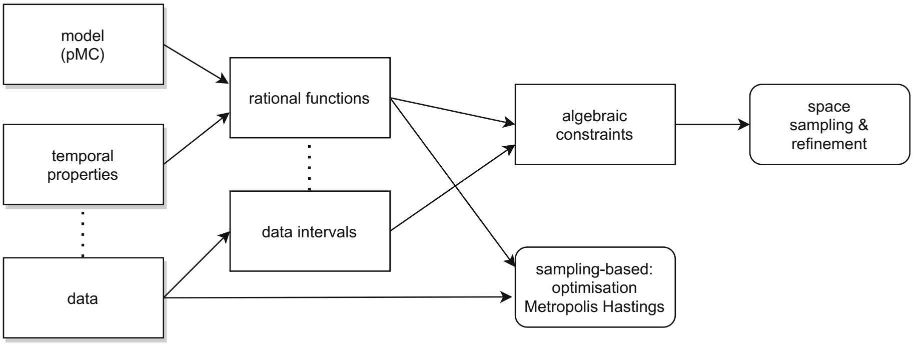


Fig. 1. The main workflow of DiPS. Parametric model checking produces a rational function, encoded as a symbolic expression representing the satisfaction probability of temporal properties, which can be observed at execution time (in the data). The data can be used to compute confidence intervals and set thresholds to rational functions, resulting in a set of algebraic constraints. The space of parameters satisfying the algebraic constraints is computed by sampling and refinement technique that partitions the space into rectangular regions. The data can be applied directly (without computing confidence intervals) with rational functions to find parameter points minimising the distance between these two inputs (optimisation) or to approximate the posterior distribution of the parameters using Bayesian inference (Metropolis-Hastings).

We start with briefly introducing the key theoretical concepts of the implemented methods (Sect. 2) and follow by stating the original features of DiPS in detail (Sect. 3). The implementation of the tool is described in Sect. 4 including the evaluation of the key tool features conducted on several models from different domains. The tool is available as open source (see footnote 1) including a ready to run virtual machine with instructions on how to run the experimental evaluation. A tutorial containing detailed information on the tool's functionality accompanied with a running example is bundled with the tool.

1.1 Related Work

Parametric model checking has been continuously developed, starting with state elimination similar to finite-state automata reduction to regular expressions [6, 15], later enhanced with set-based state elimination simplifying the intermediate results [13], all the way to the SCC decomposition technique with a special structure to store individual factors [18] that improved the speed and memory efficiency.

PRISM [22] is a well-established tool for modelling and model checking DTMCs, Continuous-Time Markov Chains (CTMCs), Markov decision processes (MDPs), and Probabilistic Timed Automata (PTA). As PRISM is easy to be installed, we use it as the first option to obtain rational functions - parametric model checking. PRISM also provides space partitioning using sampling. We leverage this functionality and add a visualisation of the result.

Storm [9] is a command-line tool for analysis of DTMCs, CTMCs, MDPs, and Markov automata (MA). It improves memory efficiency, speed, and output usability of parametric model checking by implementing efficient methods proposed in [18]. In DiPS, one can use Storm instead of PRISM to improve the performance of parametric model checking.

Storm also provides efficient parameter synthesis of Markov chains with multi-affine parametrisations – *parameter lifting* [25]. DiPS can call Storm to refine the parameter space. Storm output consists of separate results for each property while considering the lower and upper bound of the interval of the respective algebraic constraint separately. To that end, DiPS can merge these partial results to obtain (and visualise) the overall result for the conjunction of properties.

Parameter lifting technique was updated with monotonicity checking in [27].

PARAM [12] is another tool for parametric model checking of DTMCs employing state elimination and state-lumping techniques, however, it is not that efficient as Storm – see benchmarks in [8].

PROPhESY [8] supports discrete-time models with safety and liveness properties. It provides space sampling and refinement employing SMT-solvers; however, the usability is limited to properties with exactly two parameters and by the dependencies/VM environment.

In [24], Bayesian inference ideas were used to constrain the parameter values directly from data (without using rational functions). It improves the results of Statistical Model Checking (SMC), especially in the case of sparse data.

PRISM-PSY [5] implements parametric uniformisation to explore parameter space for parametrised CTMCs with Continuous Stochastic Logic (CSL) specification and employs GPU hardware. It was reused for robust design synthesis in RODES [4].

U-check [2] employs Bayesian statistical algorithm and smoothed model checking for CTMCs with Metric Interval Temporal Logic (MiTL) specification.

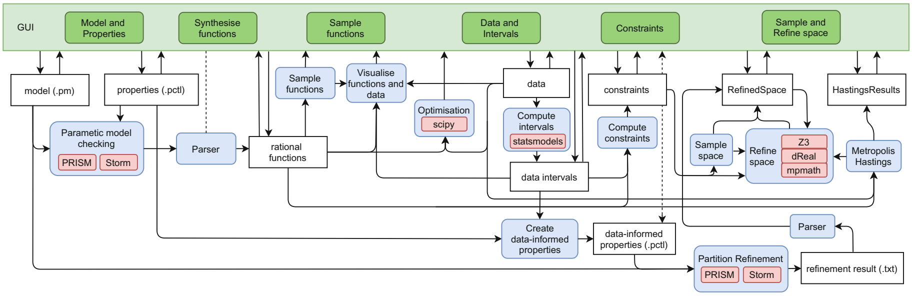


Fig. 2. The architecture of DiPS. Main GUI components, six tabs (in green), main functionality components (in blue), and leveraged tools and libraries (in red). (Color figure online)

2 Methods

In this section, we briefly recall incorporated methods introduced in former works and explain the methods and concepts in detail. Moreover, we describe a novel method based on a combination of Monte Carlo and refinement-based approaches. Additionally, we add information on the parallelisation potential of the individual methods. More information about methods' outputs and their settings in DiPS can be seen in the tutorial which is a part of the tool package.

2.1 Model Checking

Model checking verifies whether a given model satisfies a given specification, while the specification is often formalised in the form of temporal property. Probabilistic operators within the property answer questions such as whether a probability of reaching target state is higher than a given threshold, e.g. 0,4: $P_{>0.4}[F\ Target]$. In the second form of temporal properties, quantitative properties, we can ask for the value of probability itself: $P_{=?}[F\ Target]$.

When the values of probabilities within the models are unknown, a parameter can be used to address this uncertainty. Model checking parametrised models using quantitative property, in the form $P_{=?}$, results in a symbolic expression over model parameters. For pMCs, the expressions are in the form of rational functions, f_i. To obtain rational functions, we leverage already existing tools PRISM and Storm. All the methods implemented in DiPS build upon calculated rational functions.

2.2 Data

Data, $[d_1, \ldots, d_m]$, represents empirical estimates of satisfaction probabilities or reward for each of the respective properties. Data points are used to compute intervals constraining the rational function or directly within optimisation and Metropolis-Hastings.

2.3 Optimisation

Optimisation returns a single parametrisation $\hat{\theta} \in \mathbb{R}^n$ which minimises the sum of distances between the rational functions, $f(\hat{\theta}) \in \mathbb{R}$, and data, d:

$$\hat{\theta} := \arg\min_{\theta \in \Theta} \sum_{i \in \{1,\ldots,m\}} w_i \cdot dist(f_i(\theta), d_i) \tag{1}$$

where distance function $dist : [0,1] \times [0,1] \rightarrow \mathbb{R}_{\geq 0}$ can be redefined. Currently, we support the least mean squares distance as provided by `scipy` library. To reflect that an observation can be more influential or desired to achieve, w_i is a weight term to scale the distance of the respective data point.

2.4 Data Intervals

Data intervals, $[I_1, \ldots, I_m]$, are confidence intervals with given number of measurements, N, and confidence level, C. We currently support six methods for confidence intervals for proportions: standard (CLT/Wald), Agresti-Coull (default) [3], Wilson [3], Jeffreys [3], and Clopper-Pearson [3] implemented in library `stats` [26], and Rule of three [16]. The standard confidence intervals, defined as

$$I_k = d_k \pm \left(z_{\alpha/2} \sqrt{\frac{d_k(1-d_k)}{N}} \right) \tag{2}$$

where d_k is the k-th data point, and $\alpha = 1 - C$ is the chosen alpha level, are not generally reaching expected coverage of selected confidence level [3,7]. Therefore, we provide more suitable options with Agresti-Coull method as the default one. When the number of observation is low (below 40), Wilson or Jeffreys method may be more suitable [3]. All the methods are directly applicable when the data estimates probability, the observed value of a quantitative probabilistic property - $P_{=?}$.

Data intervals are then used to constrain the rational functions, $\forall k \in \{1, \ldots, m\}, f_k \in I_k$, and the algebraic constraints are used in the space sampling and refinement methods.

2.5 Sampling

The decision problem whether the instantiation of pMC model satisfies a given PCTL property can be answered by model checking the instantiated DTMC. With the knowledge of rational functions, this problem boils down to evaluating algebraic constraints.

In the sampling, a uniform grid of points is created and in each point we evaluate the constraints to mark the point *sat*(green)/*unsat*(red). Where for the sat point, all the constraints are satisfied and for unsat point at least one of the constraints is violated.

For one or two parameters, the result is visualised as green and red dots in phase space - see Fig. 4a, c. In the multidimensional case, each parametrisation satisfying the properties is visualised as a scatter-line plot where each parameter is plotted against its index in the parameter space, i.e. parametrisation $\hat{\theta} \in \Theta$ is plotted as a function: $i \mapsto \hat{\theta}_i$ for each parameter index $i \in \{1, 2, \ldots, n\}$.

Trivial parallelisation of sampling is based on the independence of algebraic constraint evaluation in the points to be sampled.

2.6 Quantitative Sampling

This method is very similar to ordinary sampling. The only difference is that instead of checking satisfaction, L1 distance to violate each of the algebraic constraints is summed to give a numeric value. For each pair of algebraic constraints derived from lower and upper bound of intervals, a lower distance is used for the pair. Positive values in the sum represent that the respective algebraic constraint is satisfied (in the given point). Note that this assumption does not hold for the whole sum. In each of the sampled points, the sum of distances is visualised by a colour spectrum.

This method hence provides quantitative estimation to better describe the satisfaction landscape of the parameter space. The parallelisation of this method benefits from the same fact as sampling: the algebraic constraints are evaluated in parallel for each point to be sampled.

2.7 Space Refinement

Here we address the problem of inferring parameter values for quantitative properties globally, not only in separate points. This problem is usually solved by space partitioning [20]. For multi-affine parametrisations, Storm implements a efficient method, *parameter lifting* [25]. Prophesy uses SMT solvers and PRISM uses sampling of the partitions to solve the problem approximatively. We provide similar methods for partitioning of space with aim to solve multiple properties in a CEGAR like style, while providing an option to run PRISM or Storm for multiple properties as well.

DiPS supports two SMT solvers, `z3` and `dreal`, and interval arithmetics as proposed in [10] (implemented by library `mpmath`) to solve the satisfaction of individual regions[2]. This is done in two steps, `check safe`, verifying whether all the points within the region are satisfying, and `check unsafe`, verifying whether all the points within the region are not satisfying. If neither of these holds, we `split` the region (in the longest dimension into two rectangles with equal volume). As verifying of the region can be expensive, we provide an option to sample the region before calling the solver - *sampling-guided refinement*. In all cases of sampling result, one of the solver calls can be skipped and if both sat and unsat samples are found, both solver calls are skipped[3] and the region is

[2] In comparison with SMT solvers, interval arithmetics provide faster iterations for price of higher probability to mark a region unknown.

[3] If the sampling contains an unsat point, it is a counterexample of safeness and vice versa if the sampling contains a sat point, it is a counterexample of unsafeness of the region under consideration.

`split` based on the position of sat vs unsat points. In more detail, we calculate rectangular hulls of the sat and unsat points. If the two hulls have no overlap there is a single line/plane dividing these two hulls and we split the region along the line/plane. If one of the two hulls is inside the other hull, we cut the space along the borders of the smaller hull. And finally, if none of two previous holds, we cut the space in all dimensions. As we use two sample points in each dimension the cutting lines/planes are always in the middle of dimension(s).

To choose a new region to check, we select all unknown regions with the biggest volume. Refinement parallelisation relies on the independence of refining these selected regions.

For one and two parameters a phase space of safe (green), unsafe (red), and unknown (white) rectangles is shown - see Fig. 4b, c. For more dimensions, overapproximation of safe or unsafe space as a projection to each of dimensions is visualised.

2.8 Metropolis-Hastings

Metropolis-Hastings [23] is a Markov chain Monte Carlo (MCMC) algorithm for approximating the posterior distribution over model parametrisations wrt. available data. For a given number of iterations, it walks through the parameter space and compares the posterior probability of the current and the next parameter point. It results in a sequence of accepted points predicting the true parameter value.

Importantly, the rational functions $f_i(\theta)$ allow us to evaluate the data likelihood $P(D \mid \theta)$ for each parametrisation and data outcome exactly. Without the rational functions, we would have to hypothesise a class of distributions proportional to the likelihood or simulate the chain to approximate the likelihood which is computationally expensive and/or imprecise.

For one or two parameters, posterior distribution is visualised as rectangularised space where the number of accepted points within each of the rectangle is visualised by a colour gradient - see Fig. 4d. For more dimensions DiPS shows scatter-line plot connecting values of parameters for each of the accepted points.

This visualisation is accompanied by two metadata visualisations. In the first one, the sequence of the accepted point with a histogram is shown for each parameter. In the latter, the sequence of all points, accepted and rejected, is shown as a projection for each of the parameters. For one or two dimensions also the sequence is shown in phase space for both accompanying visualisations.

2.9 Metropolis-Hastings-guided Refinement – HMH

The newly proposed method for parameter synthesis, Hajnal-Metropolis-Hastings (HMH), combines two interconnected methods: Metropolis-Hastings and space refinement. Posterior distribution as the result of Metropolis-Hastings serves to split the space into rectangles with marking of the number of accepted points within each of the rectangles. The discrepancy of this value and the expected number of accepted points in each of the rectangles serves to quantify expectation of probability of the rectangle to be safe. Safe rectangles are expected to contain more accepted points than unsafe rectangles. This aids the refinement procedure to select rectangles: 1. with a higher probability to be either safe or unsafe and 2. to find a safe region faster, as on many occasions, one is interested in finding a safe area rather than validate the whole parameter space.

3 Key Tool Features and Contributions

The main contribution of this paper is a tool offering a palette of parameter inference methods for pMCs. In comparison with the state-of-art we extend existing tools and workflows in the following aspects:

1. DiPS provides a fully automated computation of the confidence intervals serving as thresholds for probability satisfaction of the specified property. The user can pick one of the six methods, with the default option, Agresti Coull method [3], performing much better than the standard (Wald) method.
2. DiPS allows two satisfaction probability thresholds (a lower and an upper bound of the satisfaction probability, e.g., bounds of the confidence interval) per each single temporal property. These bounds constrain the rational functions provided that both of the respective inequalities must be satisfied. DiPS supports the conjunction of multiple constrained rational functions – algebraic expressions of satisfaction probability of *multiple properties*. Multiple properties further allow the experts to maximise the predictive power of sparse data in order to find satisfactory parameter values. We have demonstrated the necessity of such a setting in our case study of the population model of honey bee mass stinging [14] containing several different BSCCs. In that case, it was necessary to constrain reachability probabilities for each of the BSCCs.

   Native support for multiple properties allows DiPS to reach desired coverage of space refinement, while PRISM and Storm tends to reach lower than desired coverage without the possibility of continuation of the refinement to enhance the coverage.
3. We extend the palette of methods with optimisation, Metropolis-Hastings, and HMH allowing to work with large model instances. For instance, Bayesian inference will always give some information within the available time frame, even though it cannot provide a global partitioning of parameter space, as is the case with the space refinement method. In addition, Metropolis-Hastings gives the quantitative result providing more information than the qualitative sat/unsat answer.
4. Precision and efficiency of optimisation and Metropolis-Hastings is enhanced by the knowledge of rational functions. To obtain the probability of satisfaction of a PCTL formula without having rational functions, one needs to run the pMC. As this has to be done in each parameter point to be analysed, the estimation becomes imprecise and/or expensive.

5. Modularity of DiPS provides an option to begin the procedure in any phase of the workflow, starting with:
   - model, properties, and data/intervals,
   - (rational) functions and data/intervals,
   - algebraic constraints, or
   - refined space.

   This allows using manually computed confidence intervals instead of data and generalisation of input functions, adding more rational functions and/or algebraic constraints or even to load previously refined parameter space, and continue refinement with a different setting and/or algebraic constraints.
6. Modularity of the design and its multiple methods allow interconnecting the results; Metropolis-Hastings can be initialised from the optimised point, space refinement can start with initial partitioning based on sampling results (presampled refinement) or Metropolis-Hastings result (HMH), etc. – see Fig. 2.
7. Finally, DiPS is able to analyse and visualise output even for models with more than two parameters - more details with example in tutorial.

4 Implementation and Experiments

4.1 Implementation

DiPS is an open source Python project, which is capable to communicate with and leverage PRISM and Storm. The command-line interface (CLI) serves for optimal performance and fast development. It is supplied with a GUI, divided in 6 functionally different tabs. The GUI provides user-friendly access to the workflow implemented by the CLI.

In the main workflow - see Fig. 1, *models* (PRISM models in `.p` format) and *properties* (in `.pctl` format) are fed to PRISM or Storm to run parametric model checking resulting in *rational functions*. *Data* are used directly with rational functions to search for parameter point minimising the distance between the inputs (*optimisation*) or in *Metropolis-Hastings* to compute posterior distribution. For other methods (*space sampling* and *space refinement*), *data intervals* (e.g. *confidence intervals*) are computed from the data to either create *data-informed properties* or combine with rational function to create *algebraic constraints*. Moreover, *data-informed properties* (combination of properties and data intervals) and the model are used for the partitioning using PRISM or Storm, while DiPS uses algebraic constraints directly. DiPS's functional units and their connections are depicted in more details in Fig. 2.

Modularity of DiPS allows starting the workflow at any given point, allowing to adjust or to create a new input. Visualisations provide information on results of implemented methods as well as the output of partitioning results of PRISM and Storm. To parallelise the methods, we use `multiprocessing` Python library using Pools with pool.map.

4.2 Experiments

We have evaluated the performance of our tool on a variant of the well-known Knuth's die [21] and a model of stinging bees presented in [14]. The evaluation consists of three parts: (1) runtimes of parallelisation results of sampling, (2) sampling guided refinement vs regular refinement and its parallelisation, and (3) a comparison of refinement methods implemented in DiPS with PRISM and Storm implementations. Shown results were obtained using a tower PC, Skadi, with 64 bit Ubuntu 20.04.2, i9-9900K CPU, 32 GB RAM, SSD disk.

The first case study is Knuth's die [21] which emulates a 6-sided die with a coin. To obtain the result, at least three flips of the coin are used, where we used three biased coins, one for each of the flips, to generate the data – the probability of rolling a side of the die. The parameters are probabilities of tossing heads with each coin – p_1, p_2, p_3. We scale this model using a version with single parameter ($p_1 = p_2 = p_3$), two parameters ($p_1, p_2, p_3 = 0.5$), and a version with all three parameters.

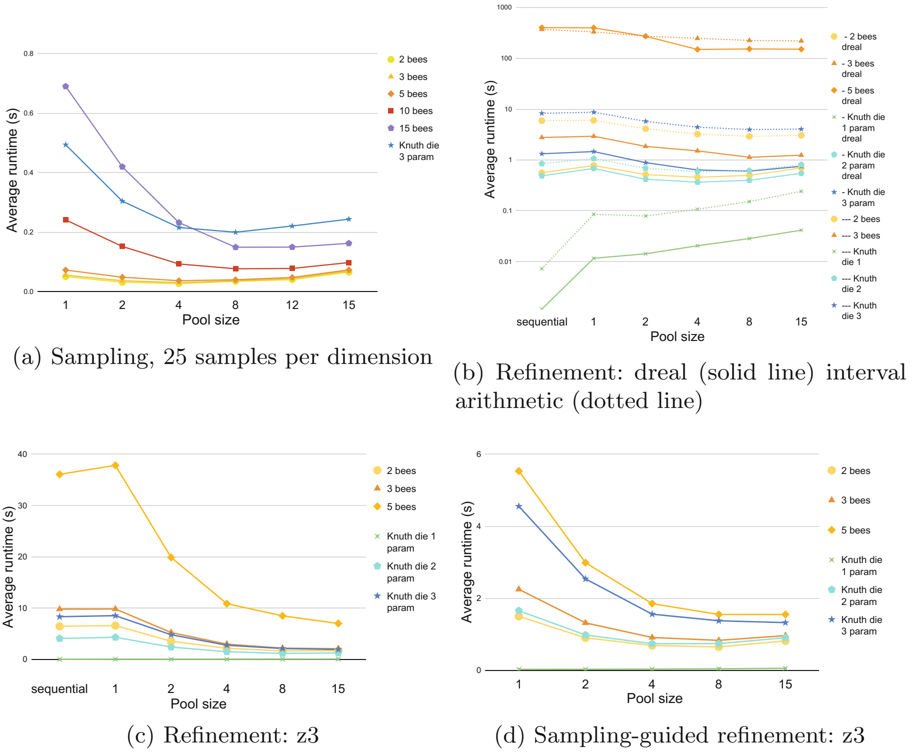
In the second case study, we look at the population model of honeybees [14]. Honeybees protect their hive against vertebrates by mass-stinging. This action costs a bee life. Collective decision which bee stings and which does not is crucial for the vitality of the colony. When the hive is attacked, a bee decides to sting with an unknown probability p. A stinging bee releases an alarm pheromone, which promotes the stinging of other bees. A bee that initially decided not to sting can sense this pheromone and opt for stinging with probability q. In the refined version of the model (multiparam), this parameter, q_i, is modulated by the number of already stinging bees, i. Here we investigate the semisynchronous version of the model, which means that in the first transition (before sensing alarm pheromone) all bees make a decision to sting or not to sting - synchronous update. Afterwards, in each transition only a single bee makes a decision - asynchoronous update. To generate synthetic data, we simulate the chain and obtain probabilities of reaching every possible number of stinging bees from zero to m: $d_0, d_1, \dots d_m$, where d_i is the fraction of simulations that ended up with i stinging bees. In the model, this is equal to the probability of reaching the respective BSCC and it can be encoded in terms of a PCTL property, $\varphi_i = P_{=?}F(BSCC_i)$. The distribution of the number of stinging bees is reflected as a conjunction of respective probabilities, $\varphi_0 = d_0 \wedge \varphi_1 = d_1 \wedge \dots \wedge \varphi_m = d_m$.

For easier comparison with results in [14] we use the same settings for confidence intervals – the Wald method with a correction term. All intervals were computed using N - number of samples: 100, C - confidence level: 0.95.

A script reproducing experiments, examined models, properties, and data is included in the tool package.

Sampling Parallelisation. Results visualised in Fig. 3a show that for smaller models the overhead of parallelisation is higher than the benefits, but in absolute values, the speedup for bigger models is much more dominant. Moreover, the overhead of more cores diminishes as the model and hence rational functions increase in size.

Refinement Parallelisation. In Fig. 3b, c we can see that the most advantage gaining solver is z3. Refinement with z3 requires fewer but more exhaustive calls; hence the overhead of creating processes is minimised. The overhead of more cores diminishes as the model and hence rational functions increase in size.



(a) Sampling, 25 samples per dimension

(b) Refinement: dreal (solid line) interval arithmetic (dotted line)

(c) Refinement: z3

(d) Sampling-guided refinement: z3

Fig. 3. Runtimes of sampling and refinement. Time in seconds (vertical axis), the sequential version and the number of processes - Pool size (horizontal axis). The curves display the average of 300 runs (sampling) and 20 runs (refinement).

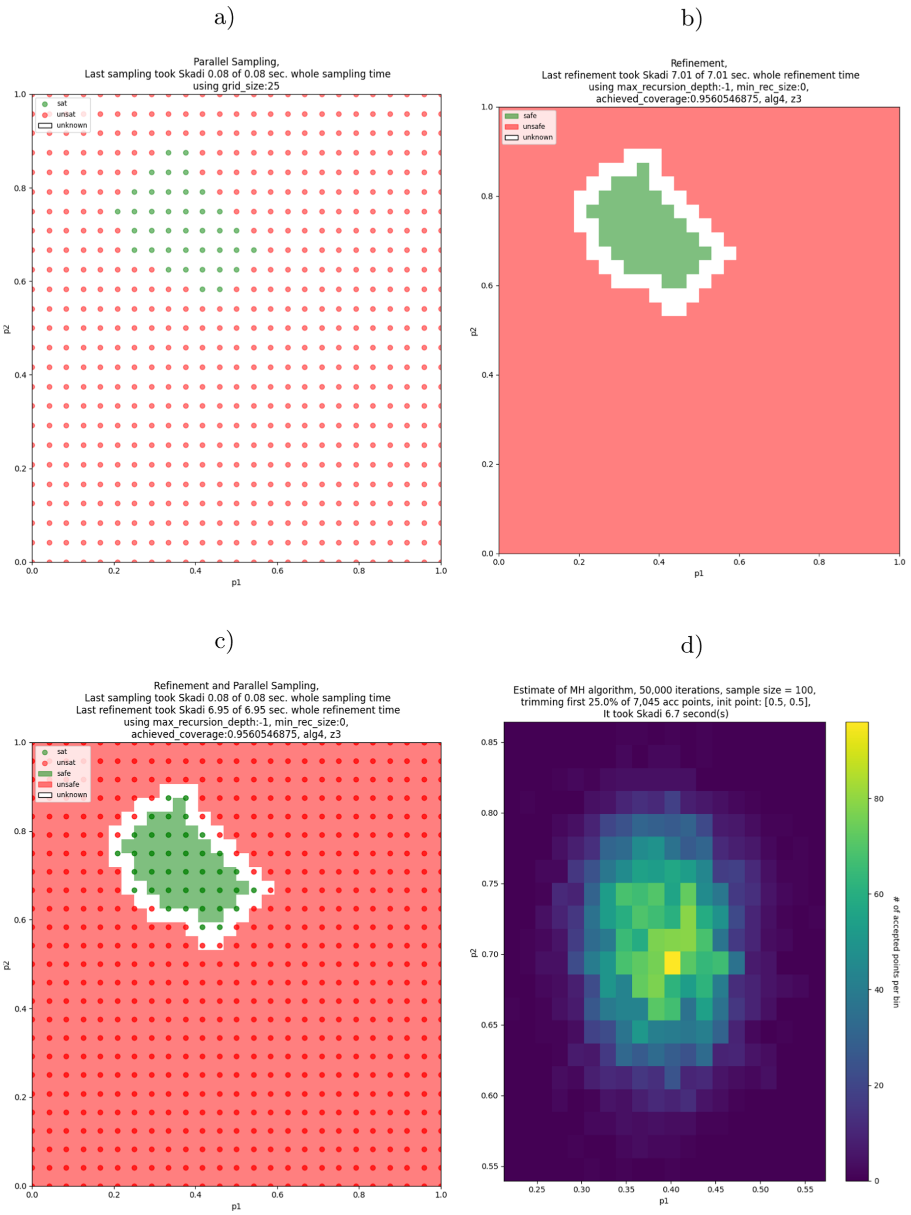
Sampling-Guided Refinement Parallelisation. We show the benefits of sampling-guided refinement using z3 solver in Fig. 3 (a vs b). In all instances the sampling-guided refinement performs better. The same effect with slightly lower amplitude can be observed using dreal. Finally, calls of interval arithmetic are so fast that the overhead of sampling overweights the benefits. We recommend using the standard (not sampling-guided) parallel refinement in this case.

Comparison of Refinement Using DiPS, Storm, and PRISM. Refinement implemented in Storm does not reach selected coverage because the merging of the refinements for the respective property may create more unknown

Table 1. Runtimes of refinement (fastest setting). Space refinement using SMT solver (z3 and dreal) and interval arithmetic. The fastest method shown in bold. Times in seconds. Timeout (TO) 1 h. In all experiments, a property specifying the reachability of all BSCCs in the model is employed (formulated as a conjunction of reachability of individual BSCCs). Number of samples, $N = 100$, confidence level, $C = 0.95$. Not shown models timed out for all three methods.

Model	Refinement SMT solver: z3, dreal	Refinement interval arithmetic
Knuth die, true point $p_1 = 0.4, p_2 = 0.7, p_3 = 0.5$ data, [0.208, 0.081, 0.1, 0.254, 0.261, 0.096]		
# states: 13, # transitions: 20, # BSCCs: 6		
Knuth unfair 1-param	0.01177, **0.001163**	0.007269
Knuth unfair 2-param	0.741, **0.3656**	0.5874
Knuth unfair 3-param	1.326, **0.6033**	3.963
Honeybee model of m agents, 2 parameters, dataset 1		
# states: 9,13,24, # transitions: 12,19,39, # BSCCs: m+1		
semisyn 2	0.6514, **0.3593**	3.46
semisyn 3	0.8337, **0.6943**	201.3
semisyn 5	**1.552**, 63.55	TO
Honeybee model of m agents, n parameters, dataset 1		
# states: 13, # transitions: 19, # BSCCs: m+1		
semisyn 3	0.3419, 0.1206	**0.1155**

regions, e.g., merging a safe and an unknown rectangle. Surprisingly, we have been unable to obtain the desired coverage for multiple properties input with PRISM as well. Hence manual tweaking of the coverage value and rerunning the analysis is necessary to obtain the desired coverage. Moreover, parameter lifting is limited to multi-affine transition functions. In conclusion, in Table 1 we show the fastest average runtimes (number of processes, sampling-guided vs regular) of refinements that reached the desired coverage. DiPS tackles the scalability problem of refinement with many parameters or large rational functions by using other methods – optimisation, sampling, and Metropolis-Hastings.



Parallel Sampling,
Last sampling took Skadi 0.08 of 0.08 sec. whole sampling time
using grid_size:25

Refinement,
Last refinement took Skadi 7.01 of 7.01 sec. whole refinement time
using max_recursion_depth:-1, min_rec_size:0,
achieved_coverage:0.9560546875, alg4, z3

Refinement and Parallel Sampling,
Last sampling took Skadi 0.08 of 0.08 sec. whole sampling time
Last refinement took Skadi 6.95 of 6.95 sec. whole refinement time
using max_recursion_depth:-1, min_rec_size:0,
achieved_coverage:0.9560546875, alg4, z3

Estimate of MH algorithm, 50,000 iterations, sample size = 100,
trimming first 25.0% of 7,045 acc points, init point: [0.5, 0.5],
It took Skadi 6.7 second(s)

Fig. 4. Visualisation (screenshot from GUI of DiPS) of sampling (a), refinement (b), sampling and refinement (c), and Metropolis-Hastings (d) as a result of 2-param Knuth model. Examples of multidimensional visualisations are shown and explained in the tutorial. (Color figure online)

5 Conclusions and Future Work

We presented a new open source tool, DiPS, dedicated to parameter exploration for pMCs. It focuses on multiple temporal logic properties informed by data. To this aim, we automatically compute rational functions – symbolic representations of satisfaction of each property, by leveraging the existing parameter synthesis tools, as well as their respective probability thresholds, through the confidence intervals derived from data following frequentist statistics interpretation. These two elements are coupled into algebraic constraints over unknown parameters. DiPS solves the algebraic constraints by partitioning the parameter space and can leverage PRISM or Storm for parameter synthesis as well. We add the visualisation of the synthesis results including merging of the Storm bound-wise results. The tool implements two additional methods for parameter exploration, optimisation and Metropolis-Hastings, to tackle the scalability problem of Space Refinement. Moreover, we proposed a new method, HMH, for parameter synthesis combining Metropolis-Hastings and refinement. Finally, parallelisation, modularity, and interconnections of the methods provide further advantages.

In comparison with the mentioned tools, DiPS improves analysis for multiple observations using the conjunction of properties, visualisation of functions in selected points to compare with data, and it complements the analyses with optimisation and Bayesian inference. The possibility to apply different approaches to explore the parameters is especially useful because these analyses have a different trade-off between computational efficiency and the type of information they provide, and the modeller may want to explore different approaches. For example, Bayesian inference will always give some information within the available time frame, but it cannot provide a global partitioning of parameter space, as it is the case with the space refinement method.

We illustrate the applicability of the tool on a variation of Knuth's die[21] and a case study of honeybee mass-stinging behaviour [14].

References

1. Ashok, P., Daca, P., Křetínský, J., Weininger, M.: Statistical model checking: black or white? In: Margaria, T., Steffen, B. (eds.) ISoLA 2020. LNCS, vol. 12476, pp. 331–349. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-61362-4_19
2. Bortolussi, L., Milios, D., Sanguinetti, G.: U-check: model checking and parameter synthesis under uncertainty. In: Campos, J., Haverkort, B.R. (eds.) QEST 2015. LNCS, vol. 9259, pp. 89–104. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-22264-6_6
3. Brown, L.D., Cai, T.T., DasGupta, A.: Interval estimation for a binomial proportion. Stat. Sci. **16**, 101–117 (2001)
4. Calinescu, R., Češka, M., Gerasimou, S., Kwiatkowska, M., Paoletti, N.: RODES: a robust-design synthesis tool for probabilistic systems. In: Bertrand, N., Bortolussi, L. (eds.) QEST 2017. LNCS, vol. 10503, pp. 304–308. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66335-7_20

5. Češka, M., Pilař, P., Paoletti, N., Brim, L., Kwiatkowska, M.: PRISM-PSY: precise GPU-accelerated parameter synthesis for stochastic systems. In: Chechik, M., Raskin, J.-F. (eds.) TACAS 2016. LNCS, vol. 9636, pp. 367–384. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49674-9_21
6. Daws, C.: Symbolic and parametric model checking of discrete-time Markov chains. In: Liu, Z., Araki, K. (eds.) ICTAC 2004. LNCS, vol. 3407, pp. 280–294. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-31862-0_21
7. Dean, N., Pagano, M.: Evaluating confidence interval methods for binomial proportions in clustered surveys. J. Surv. Stat. Methodol. **3**(4), 484–503 (2015)
8. Dehnert, C., et al.: PROPhESY: a PRObabilistic ParamEter SYnthesis tool. In: Kroening, D., Păsăreanu, C.S. (eds.) CAV 2015. LNCS, vol. 9206, pp. 214–231. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21690-4_13
9. Dehnert, C., Junges, S., Katoen, J.-P., Volk, M.: A Storm is coming: a modern probabilistic model checker. In: Majumdar, R., Kunčak, V. (eds.) CAV 2017. LNCS, vol. 10427, pp. 592–600. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63390-9_31
10. Gainer, P., Hahn, E.M., Schewe, S.: Accelerated model checking of parametric Markov chains. In: Lahiri, S.K., Wang, C. (eds.) ATVA 2018. LNCS, vol. 11138, pp. 300–316. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01090-4_18
11. Hahn, E.M., Han, T., Zhang, L.: Synthesis for PCTL in parametric Markov decision processes. In: Bobaru, M., Havelund, K., Holzmann, G.J., Joshi, R. (eds.) NFM 2011. LNCS, vol. 6617, pp. 146–161. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20398-5_12
12. Hahn, E.M., Hermanns, H., Wachter, B., Zhang, L.: PARAM: a model checker for parametric Markov models. In: Touili, T., Cook, B., Jackson, P. (eds.) CAV 2010. LNCS, vol. 6174, pp. 660–664. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14295-6_56
13. Hahn, E.M., Hermanns, H., Zhang, L.: Probabilistic reachability for parametric Markov models. Int. J. Softw. Tools Technol. Transf. **13**(1), 3–19 (2011)
14. Hajnal, M., Nouvian, M., Petrov, T., Šafránek, D.: Data-informed parameter synthesis for population Markov chains. In: Bortolussi, L., Sanguinetti, G. (eds.) CMSB 2019. LNCS, vol. 11773, pp. 383–386. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-31304-3_32
15. Han, Y.S.: State elimination heuristics for short regular expressions. Fund. Inform. **128**(4), 445–462 (2013)
16. Hanley, J., Lippman-Hand, A.: If nothing goes wrong, is everything all right? Interpreting zero numerators. JAMA **249**(13), 1743–1745 (1983)
17. Hansson, H., Jonsson, B.: A logic for reasoning about time and reliability. Formal Aspects Comput. **6**(5), 512–535 (1994). https://doi.org/10.1007/BF01211866
18. Jansen, N., et al.: Accelerating parametric probabilistic verification. In: Norman, G., Sanders, W. (eds.) QEST 2014. LNCS, vol. 8657, pp. 404–420. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10696-0_31
19. Junges, S., et al.: Parameter synthesis for Markov models. CoRR abs/1903.07993 (2019). http://arxiv.org/abs/1903.07993
20. Katoen, J.P.: The probabilistic model checking landscape. In: Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, pp. 31–45. ACM (2016)
21. Knuth, D., Yao, A.: The complexity of nonuniform random number generation. In: Algorithms and Complexity: New Directions and Recent Results. Academic Press (1976)

22. 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
23. Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H., Teller, E.: Equation of state calculations by fast computing machines. J. Chem. Phys. **21**(6), 1087–1092 (1953)
24. Polgreen, E., Wijesuriya, V.B., Haesaert, S., Abate, A.: Data-efficient Bayesian verification of parametric Markov chains. In: Agha, G., Van Houdt, B. (eds.) QEST 2016. LNCS, vol. 9826, pp. 35–51. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-43425-4_3
25. Quatmann, T., Dehnert, C., Jansen, N., Junges, S., Katoen, J.-P.: Parameter synthesis for Markov models: faster than ever. In: Artho, C., Legay, A., Peled, D. (eds.) ATVA 2016. LNCS, vol. 9938, pp. 50–67. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46520-3_4
26. Seabold, S., Perktold, J.: Statsmodels: econometric and statistical modeling with Python. In: 9th Python in Science Conference (2010)
27. Spel, J., Junges, S., Katoen, J.P.: Finding provably optimal Markov chains. Tools Algorithms Const. Anal. Syst. **12651**, 173 (2021)