Precise Parameter Synthesis for Stochastic Petri Nets with Interval Rate Parameters *

Milan Češka jr.[1], Milan Češka[1], and Nicola Paoletti[2]

[1] Faculty of Information Technology, Brno University of Technology, Czech Republic
[2] Department of Computer Science, Stony Brook University, USA

Abstract. We consider the problem of synthesising parameters affecting transition rates and probabilities in generalised Stochastic Petri Nets (GSPNs). Given a time-bounded property expressed as a probabilisitic temporal logic formula, our method allows computing the parameters values for which the probability of satisfying the property meets a given bound, or is optimised. We develop algorithms based on reducing the parameter synthesis problem for GSPNs to the corresponding problem for continuous-time Markov Chains (CTMCs), for which we can leverage existing synthesis algorithms, while retaining the modelling capabilities and expressive power of GSPNs. We evaluate the usefulness of our approach by synthesising parameters for two case studies.

1 Introduction

Various extensions of Stochastic Petri Nets (SPNs), e.g. generalised SPNs [12] (GSPNs), have been introduced to model complex and concurrent systems in many areas of science. In biochemistry, quantitative models of genetic networks can be expressed as SPNs [8]. In engineering, GSPNs are used to study various reliability and performance aspects of manufacturing processes, computer networks and communication protocols [1, 3]. Assuming certain restrictions on their structure, the dynamics of SPNs as well as GSPNs can be described using finite-state continuous-time Markov chains (CTMCs) [12]. This allows modellers and designers to perform quantitative analysis and verification using well-established formal techniques for CTMCs, above all probabilistic model checking [11].

Traditionally, formal analysis of SPNs and GSPNs assumes that transition rates and probabilities are known a priori. This is often not the case and one has to consider ranges of parameter values instead, for example, when the parameters result from imprecise measurements, or when designers are interested in finding parameter values such that the model fulfils a given specification. In this paper, we tackle the parameter synthesis problem for GSPNs, described as follows: *"Given a time-bounded temporal formula describing the required behaviour and a parametric GSPN (pGSPN) whose transition rates and probabilities are functions of the parameters, automatically find parameter values such that the satisfaction probability of the formula meets a given threshold, is maximised, or minimised"*.

Importantly, this problem requires effective reasoning about uncountable sets of GSPNs, arising from the presence of continuous parameter ranges. We show that, under restrictions on the structure of pGSPNs (i.e. requiring a finite number

* This work has been supported by the Czech Grant Agency grant No. GA-24707Y and the IT4Innovations Excellence in Science project No. LQ1602.

of reachable markings or avoiding Zeno behaviour) and on predicates appearing in the temporal formulas, we can describe the dynamics of a pGSPN by a finite-state parametric CTMC (pCTMC). The parameter synthesis problem for pGSPNs can be then reduced to the equivalent problem for pCTMCs and thus, we can employ existing synthesis algorithms that combine computation of probability bounds for pCTMCs with iterative parameter space refinement in order to provide arbitrarily precise answers [5]. We further demonstrate that pGSPNs provide an adequate modelling formalism for designing complex systems where parameters of the environment (e.g., request inter-arrival times) and those inherent to the system (e.g. service rates) can be meaningfully expressed as intervals. We also show that pGSPNs can be used for the *in silico* analysis of stochastic biochemical systems with uncertain kinetic parameters.

The main contribution of the paper can be summarised as follows:

- formulation of the parameter synthesis problem for GSPNs using pGSPNs;
- solution method based on translation of pGSPNs into pCTMCs; and
- evaluation on two case studies from different domains, through which we demonstrate the usefulness and effectiveness of our method.

2 Problem Formulation

In our work we consider the problem of parameter synthesis for Generalised Stochastic Petri Nets (GSPNs). GSPNs naturally combine stochastic (i.e. timed) transitions and immediate (i.e. untimed and probabilisitic) transitions [4] and thus provide an adequate formalism for modelling engineered and biological systems alike. Below we introduce the model of *parametric* GSPNs (*p*GSPNs), which extends GSPNs with parameters that affect transitions rates and probabilities.

Definition 1 (*p*GSPN). *Let K be a set of parameters. A pGSPN over K is a tuple $(L, T, A, M_{in}, \mathbf{R}, \mathbf{P})$, where:*

- *L is a finite set of places inducing a set of markings M, where for each $m \in M$, $m = (m_1, m_2, \ldots, m_n) \in \mathbb{R}^n$, with $n = |L|$;*
- *$T = T_{st} \cup T_{im}$ is a finite set of transitions partitioned into stochastic transitions T_{st} and immediate transitions T_{im};*
- *$A \subseteq (P \times T) \cup (T \times P)$ is a finite set of arcs connecting transitions and places;*
- *$m_{in} \in M$ is the initial marking;*
- *$\mathbf{R} : T_{st} \to (M \to \mathbb{R}[K])$ is the parametric, marking-dependent rate matrix, where $\mathbb{R}[K]$ is the set of polynomials over the reals $\mathbb{R}$ with variables $k \in K$;*
- *$\mathbf{P} : T_{im} \to (M \to \mathbb{R}[K])$ is the parametric, marking-dependent probability matrix.*

The domain of each parameter $k \in K$ is given by a closed real interval describing the range of possible values, i.e, $[k^{\perp}, k^{\top}] \subseteq \mathbb{R}$. The parameter space $\mathcal{P}$ induced by K is defined as the Cartesian product $\mathcal{P} = \bigtimes_{k \in K} [k^{\perp}, k^{\top}]$. Subsets of $\mathcal{P}$ are called *subspaces*. Given a *p*GSPN and a parameter space $\mathcal{P}$, we denote with $\mathcal{N}_{\mathcal{P}}$ the set $\{\mathcal{N}_p \mid p \in \mathcal{P}\}$ where $\mathcal{N}_p = (L, T, A, M_{in}, \mathbf{R}_p, \mathbf{P}_p)$ is the instantiated GSPN obtained by replacing the parameters in $\mathbf{R}$ and $\mathbf{P}$ with their valuation

in p. The rate (probability) matrix for marking m and parameter valuation p is denoted by $\mathbf{R}_{m,p}$ ($\mathbf{P}_{m,p}$).

For all markings $m \in M$ reachable from m_{in}, we require that: 1) For all $t \in T_{st}$, it holds that either $\mathbf{R}_{p,m}(t) > 0$ for all $p \in \mathcal{P}$, or $\mathbf{R}_{p,m}(t) = 0$ for all $p \in \mathcal{P}$. 2) For all $t \in T_{im}$ it holds that either $\mathbf{P}_{p,m}(t) > 0$ for all $p \in \mathcal{P}$ or $\mathbf{P}_{p,m}(t) = 0$ for all $p \in \mathcal{P}$, and $\sum_{t \in T_{im}} P_{p,m}(t) = 1$. Note that $\mathbf{R}_{p,m}(t) > 0$ ($\mathbf{P}_{p,m}(t) > 0$, respectively) if and only if the transition t is enabled, i.e. there is a sufficient number of tokens in each of its input places. In other words, parameters do not affect the enabledness of transitions. Further, we use the notion of *capacity* C to indicate, for each place $l \in L$, the maximal number of tokens $C(l)$ in l, thus determining when a transition is enabled.

Vanishing and tangible markings. As in GSPNs, a marking $m \in M$ is called *vanishing* if there is an immediate transition $t \in T_{in}$ that is enabled in m, or *tangible* otherwise. In a vanishing marking m, all stochastic transitions are blocked, and the enabled immediate transitions are fired in zero time according the probability distribution $\mathbf{P}_{p,m}$. To avoid Zeno behaviour, we require that there are no cycles over vanishing markings. In a tangible marking m, the sojourn time is exponentially distributed with average time $E_p(m)^{-1}$ where $E_p(m) = \sum_{t \in T_{st}} \mathbf{R}_{p,m}(t)$ is the exit rate. The probability that a transition $t \in T_{st}$ is fired first is given by $\mathbf{R}_{p,m}(t) \cdot E_p(m)^{-1}$.

Specification language. We consider the time-bounded fragment of Continuous Stochastic Logic (CSL) [2] to specify behavioural properties of GSPNs, with the following syntax. A state formula Φ is given by $\Phi ::= \mathsf{true} \mid a \mid \neg\Phi \mid \Phi \wedge \Phi \mid P_{\sim r}[\phi] \mid P_{=?}[\phi]$, where ϕ is a path formula, given by $\phi ::= \mathsf{X}\ \Phi \mid \Phi\ \mathsf{U}^I\ \Phi$, a is an atomic proposition defined over the markings $m \in M$, $\sim \in \{<, \leq, \geq, >\}$, $r \in [0,1]$ is a probability threshold, and $I \in \mathbb{R}$ is a bounded interval. As explained in [5], our parameter synthesis methods also support time-bounded rewards [11], which we omit in the following for the sake of clarity.

Let ϕ be a CSL path formula and $\mathcal{N}_{\mathcal{P}}$ be a pGSPN over a space $\mathcal{P}$. We denote with $\Lambda_\phi : \mathcal{P} \to [0,1]$ the *satisfaction function* such that $\Lambda_\phi(p) = P_{=?}[\phi]$, that is, $\Lambda_\phi(p)$ is the probability of ϕ being satisfied over the GSPNs $\mathcal{N}_p$. Note that the path formula ϕ may contain nested probabilistic operators, and therefore the satisfaction function is, in general, not continuous.

Synthesis problems. We consider two parameter synthesis problems: the *threshold synthesis* problem that, given a threshold $\sim r$ and a CSL path formula ϕ, asks for the subspace where the probability of ϕ meets $\sim r$; and the *max synthesis* problem that determines the subspace within which the probability of the input formula is guaranteed to attain its maximum, together with probability bounds that contain that maximum. Solutions to the threshold synthesis problem admit parameter points left undecided, while, in the max synthesis problem, the actual set of maximising parameters is contained in the synthesised subspace. Importantly, the undecided and maximising subspaces can be made arbitrarily precise through user-defined tolerance values.

For $\mathcal{N}_{\mathcal{P}}$, ϕ, an initial state s_0, a threshold $\sim r$ and a volume tolerance $\varepsilon > 0$, the *threshold synthesis* problem is finding a partition $\{\mathcal{T}, \mathcal{U}, \mathcal{F}\}$ of $\mathcal{P}$, such that:

$\forall p \in \mathcal{T} : \Lambda_\phi(p) \sim r$; $\forall p \in \mathcal{F} : \Lambda_\phi(p) \not\sim r$; and $\mathrm{vol}(\mathcal{U})/\mathrm{vol}(\mathcal{P}) \leq \varepsilon$, where $\mathcal{U}$ is an undecided subspace and $\mathrm{vol}(A) = \int_A 1 d\mu$ is the volume of A.

For $\mathcal{N}_\mathcal{P}$, ϕ, s_0, and a probability tolerance $\epsilon > 0$, the *max synthesis* problem is finding a partition $\{\mathcal{T}, \mathcal{F}\}$ of $\mathcal{P}$ and probability bounds $\Lambda_\phi^\perp$, $\Lambda_\phi^\top$ such that: $\forall p \in \mathcal{T} : \Lambda_\phi^\perp \leq \Lambda_\phi(p \leq \Lambda_\phi^\top$; $\exists p \in \mathcal{T} : \forall p' \in \mathcal{F} : \Lambda_\phi(p) > \Lambda_\phi(p')$; and $\Lambda_\phi^\top - \Lambda_\phi^\perp \leq \epsilon$. The min synthesis problem is defined in a symmetric way to the max case.

3 Parameter Synthesis for Stochastic Petri Nets

First, we introduce a novel automated translation from pGSPNs into parametric CTMCs (pCTMCs), able to preserve important quantitative temporal properties. This allows us to reduce the pGSPN synthesis problem to the equivalent pCTMC synthesis problem. Second, we provide an overview of our recent results on parameter synthesis for CTMCs [5].

3.1 Translation of pGSPNs into pCTMCs

CTMCs represent purely stochastic processes and thus, in contrast to GSPNs, they do not allow any immediate transitions. The dynamics of a CTMC is given by a transition rate matrix defined directly over its set of states S. *Parametric CTMCs* [5] extend the notion of CTMCs by allowing transitions rates to depend on model parameters. Formally, for a set of parameters K, the parametric rate matrix is defined as $\mathbf{M} : S \times S \to \mathbb{R}[K]$. Similarly as in the case pGSPNs, for a given parameter space $\mathcal{P}$ the pCTMC $\mathcal{C}_\mathcal{P}$ defines an uncountable set $\{\mathcal{C}_p \mid p \in \mathcal{P}\}$ where $\mathcal{C}_p = (S, \mathbf{M}_p, s_0)$ is the instantiated CTMC obtained by replacing the parameters in $\mathbf{M}$ with their valuation in p and where s_0 denotes the initial state.

We introduce a translation method from pGSPNs to pCTMCs that builds on the translation for non-parametric GSPNs [4] and exploits the fact that parameters do not affect the enabledness of transitions and thus do not alter the set of reachable markings. Therefore we can map the set of markings M in the pGSPN $\mathcal{N}_\mathcal{P}$ to the set of states S in the pCTMC $\mathcal{C}_\mathcal{P}$. This mapping allows us to construct the parametric rate matrix $\mathbf{M}$ of the pCTMC such that $\mathcal{C}_\mathcal{P}$ preserves the dynamic of pGSPN over tangible markings. Formally, for $p \in \mathcal{P}$, $\mathbf{M}_p(m, m') = \sum_{t \in T(m,m')} \mathbf{R}_{p,m}(t)$, where $T(m, m') = \{t \in T_{st} \mid m' \text{ is a marking obtained by firing } t \text{ in marking } m\}$.

The crucial difficulty of this translation lies in handling the vanishing markings. Since any state in a CTMC has a non-zero waiting time, in order to map pGSPN markings into pCTMC states we need to eliminate the vanishing markings. Specifically, for each vanishing marking, we merge the incoming and outgoing transitions and combine the corresponding parameters (if present).

Although we merge at the level of markings, we first explain the intuition of the merging at the level of places, which is illustrated in Figure 1 (left). The reasoning behind this operation is that the immediate probabilistic transitions t_2 and t_3 take zero time and thus they do not affect the total exit rate in the resulting, merged transitions t_4 and t_5, i.e., $\mathbf{R}(t_1) = \mathbf{R}(t_4) + \mathbf{R}(t_5)$. The transition probabilities k_2 and $1 - k_2$ of t_2 and t_3, respectively, are used to determine the probability that the transition t_4 and t_5, respectively, is fired when the sojourn time in the place 1 is passed.

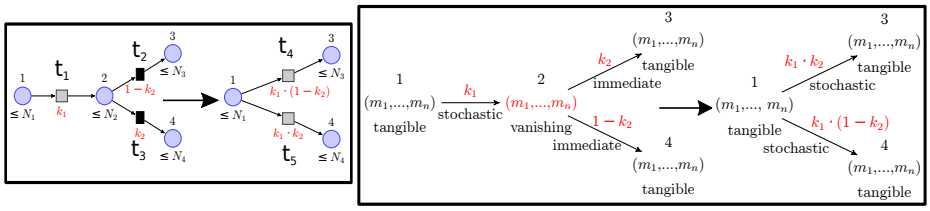

Fig. 1: **Left:** Merging at the level of places. **Right:** Merging at the level of markings.

Since the transition rates and probabilities are marking-dependent, elimination and merging is actually performed at the level of markings. Figure 1 (right) illustrates these operations. The principle is same as in the case of merging at the level of places, but it allows us to reflect the dependencies on the markings and thus to preserve the dynamic of pGSPN over tangible markings. Clearly, we cannot reason about the vanishing markings and thus trajectories in pGSPN that differ only in vanishing markings are indistinguishable in the resulting pCTMC. To preserve the correctness of our approach we have to restrict the set of properties. In particular, we support only properties with atomic propositions defined over the set of tangible markings. This allow us to compute the satisfaction function Λ_ϕ using the satisfaction function Γ_ϕ defined over the pCTMC.

3.2 Parameter synthesis for pCTMCs

We now give an overview of the parameter synthesis method for pCTMCs [5]. The method builds on the computation of safe bounds for the satisfaction function Γ_ϕ. In particular, for a pCTMC $\mathcal{C}_\mathcal{P}$, the procedure efficiently computes an interval $[\Gamma^\mathcal{P}_\bot, \Gamma^\mathcal{P}_\top]$ that is guaranteed to contain the minimal and maximal probability that pCTMC $\mathcal{C}_\mathcal{P}$ satisfies ϕ, i.e. $\Gamma^\mathcal{P}_\bot \leq \min_{p\in\mathcal{P}} \Gamma_\phi(p)$ and $\Gamma^\mathcal{P}_\top \geq \max_{p\in\mathcal{P}} \Gamma_\phi(p)$.

This safe over-approximation is computed through an algorithm that extends the well-established time-discretisation technique of *uniformization* [11] for the transient analysis of CTMCs, which is the foundation of CSL model checking. Our technique is efficient because the computation of maximal and minimal probabilities is based on solving a series of local and independent optimisation problems at the level of each transition and each discrete uniformisation step. This approach is much more feasible than solving the global optimisation problem, which reduces to the optimisation of a high-degree multivariate polynomial.

The complexity of our approach depends on the degree of the polynomials appearing in the transition matrix $\mathbf{M}$. We restrict to *multi-affine* polynomial rate functions for which we have a complexity of $\mathcal{O}(2^{n+1} \cdot t_{CSL})$, where n is the number of parameters and t_{CSL} is the complexity of the standard non-parametric CSL model-checking algorithm, which mainly depends on the size of the underlying model and the number of discrete time steps (the latter depends on the maximum exit rate of the CTMC and time bound in the CSL property). For linear rate functions, we have an improved complexity of $\mathcal{O}(n \cdot t_{CSL})$.

Crucially, the approximation error of this technique depends linearly on the volume of the parameter space and exponentially on the number of discrete

time steps [5], meaning that the error can be controlled by refining the parameter space (i.e. the error reduces with the volume of the parameter region). In particular, the synthesis algorithms are based on the iterative refinement of the parameter space $\mathcal{P}$ and the computation of safe bounds for Γ_ϕ (as per above).

Threshold synthesis. We start with $\mathcal{U}$ containing $\mathcal{P}$ (i.e. the whole parameter space is undecided). For each $\mathcal{R} \in \mathcal{U}$, we compute the safe bounds $\Gamma_\bot^\mathcal{R}$ and $\Gamma_\top^\mathcal{R}$. Then, assuming a threshold $\leq r$, if $\Gamma_\top^\mathcal{R} \leq r$ then for all $p \in \mathcal{R}$ the threshold on the property ϕ is satisfied and $\mathcal{R}$ is added to $\mathcal{T}$. Similarly, if $\Gamma_\bot^\mathcal{R} > r$ then $\mathcal{R}$ is added to $\mathcal{F}$. Otherwise, $\mathcal{R}$ is decomposed into subspaces that are added to $\mathcal{U}$. The algorithm terminates when $\mathcal{U}$ satisfies the required volume tolerance ε. Termination is guaranteed by the shape of approximation error.

Max synthesis. The algorithm starts with $\mathcal{T} = \mathcal{P}$ and iteratively refines $\mathcal{T}$ until the probability tolerance ϵ on the bounds $\Lambda_\phi^\bot = \Gamma_\bot^\mathcal{T}$ and $\Lambda_\phi^\top = \Gamma_\top^\mathcal{T}$ is satisfied. To provide faster convergence, at each iteration it computes an under-approximation M of the actual maximal probability, by randomly sampling a set of probability values and setting M to the maximal sample. In this way, each subspace whose maximal probability is below M can be safely added to $\mathcal{F}$. Otherwise, $\mathcal{R}$ is added to a newly constructed set $\mathcal{T}$ and the bounds $\Lambda_\phi^\bot$ and $\Lambda_\phi^\top$ are updated accordingly. Since that the satisfaction function Λ_ϕ is in general discontinuous, the algorithm might not terminate. This is detected by extending the termination criterion using the volume tolerance ε.

The complexity of the synthesis algorithms depends mainly on the number of subspaces to analyse in order to achieve the desired precision, since this number directly affects how many times the procedure computing the bounds on Γ_ϕ has to be executed. Therefore, complex instances of the synthesis problem can be computationally very demanding. To overcome this problem, we redesigned the sequential algorithms to enable state space and parameter space parallelisation, resulting in data-parallel algorithms providing dramatic speed-up on many-core architectures [6]. Thanks to the translation procedure previously described, we can exploit parallelisation also for the parameter synthesis of GSPNs.

4 Experimental Results

All the experiments were performed using an extended version of the GPU-accelerated tool PRISM-PSY [6]. They run on a Linux workstation with an AMD PhenomTM II X4 940 Processor @ 3GHz, 8 GB DDR2 @ 1066 MHz RAM and an NVIDIA GeForce GTX 480 GPU with 1.5 GB of GPU memory.

Google File System. We consider a case study of the replicated file system used in the Google search engine known as Google File System (GFS). The model, introduced in [3] as a non-parametric GSPN, reproduces the life-cycle of a single chunk (representing a part of the file) within the file system. The chunk exists in several copies, located in different chunk servers. There is one master server that is responsible for keeping the locations of the chunk copies and replicating the chunks if a failure occurs. In [6] we introduced a parametric version of the model, manually derived the corresponding pCTMC, and performed parameter synthesis using the tool PRISM-PSY.

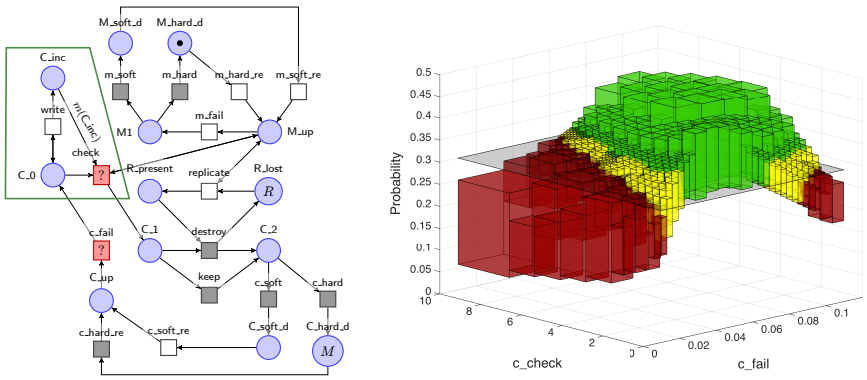

Fig. 2: **Left:** pGSPN for a new variant of the GFS model [3]. Red boxes with question marks indicate parametric transitions. **Right:** Results of the threshold synthesis.

In this paper we exploit the modelling capabilities of pGSPNs and introduce an extension of the model. In particular, we integrate into the model a message monitoring inspired by the original paper on GFS [7]: the master server periodically send so-called *HeartBeat* messages to check for chunk inconsistency, i.e. when a write request occur between a failure and its acknowledgement. Figure 2 (left) depicts the pGSPN describing the parametric model, where the part in the green-bordered box corresponds to the extension.

We are interested in the probability that the first chunk inconsistency occurs minutes 15 and 45, and how this probability depends on the rate of the HeartBeat messages (c_check) and the rate of the chunk server failure (c_fail). We solve this as a threshold synthesis problem with threshold $\geq 30\%$, path formula $\phi \equiv (\mathrm{C}_{inc} = 0)\ U^{[15,45]}\ (\mathrm{C}_{inc} > 0)$, parameter intervals c_check $\in [0.01, 10]$ and c_fail $\in [0.01, 0.11]$, and volume tolerance $\varepsilon = 10\%$.

Results are shown in Figure 2 (right), namely, the decomposition of the parameter space into subspaces satisfying the property (green), not satisfying (red), and uncertain (yellow). The pGSPN has around 139K states and 740K transitions, and the synthesis algorithm required around 11K time steps and produced 460 final subspaces. The data-parallel GPU computation took 25 minutes, corresponding to more than 7-fold speedup with respect to the sequential algorithm.

Mitogen-Activated-Protein-Kinases cascade. In our second case study we consider the Mitogen-Activated-Protein-Kinases (MAPK) cascade [10], one of the most important signalling pathways that controls molecular growth through activation (i.e. phosphorylation) cascade of kinases. We use a SPN model introduced in [9] and study how two key reactions, namely activation by MAPKK-PP and deactivation by Phosphatase, affect the activation of the final kinases. We want to find rates of these reactions that maximise the probability that, within 50 and 55 minutes, the number of the activated kinases is between 25% and 50%. To this purpose, we formulate a max synthesis problem for property $\phi \equiv G[50, 55]\ (25\% \leq \gamma \leq 50\%)$, where γ is the percentage of the activated kinases. The interval for both reaction rates is $[0.01, 0.1]$ and the probability tolerance $\epsilon = 5\%$.

Figure 3 illustrates the results of max synthesis, namely, it shows the decomposition of the parameter space into subspaces maximising the property (green) and not maximising (red). The bounds on maximal probability are 57.4% and 62.4%. The pGSPN has around 100K states and 911K transitions, and the synthesis algorithm required around 121K time steps and produced 259 final subspaces. The parallel GPU computation took 5 hours, corresponding to more than 22-fold speedup with respect to the sequential CPU algorithm.

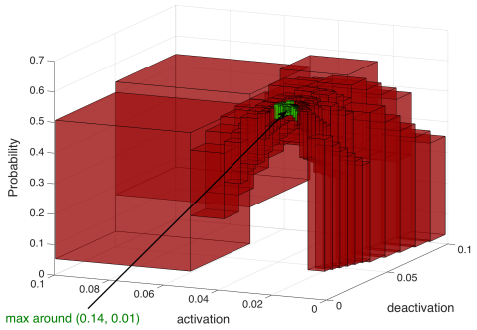

Fig. 3: Results of the max synthesis for the MAPK cascade.

5 Conclusion

We have developed efficient algorithms for synthesising parameters in GSPNs, building on the automated translation of parametric GSPNs into parametric CTMCs. The experiments show that our approach allows us to exploit existing data-parallel algorithms for scalable synthesis of CTMCs, while retaining the modelling power provided by parametric GSPNs.

References

1. Robert Y Al-Jaar and Alan A Desrochers. Performance evaluation of automated manufacturing systems using generalized stochastic petri nets. *IEEE Transactions on Robotics and Automation*, 6(6):621–639, 1990.
2. Adnan Aziz et al. Verifying Continuous Time Markov Chains. In *Proc. of CAV'96*, pages 269–276. 1996.
3. Christel Baier et al. Model checking for performability. *Mathematical Structures in Computer Science*, 23(04):751–795, 2013.
4. Gianfranco Balbo. Introduction to generalized stochastic petri nets. In *Proc. of SFM'07*, volume 4486 of *LNCS*, pages 83–131. Springer, 2007.
5. Milan Češka et al. Precise parameter synthesis for stochastic biochemical systems. *Acta Informatica*, pages 1–35, 2016.
6. Milan Češka et al. PRISM-PSY: Precise GPU-accelerated parameter synthesis for stochastic systems. In *Proc. of TACAS'16*, pages 367–384. 2016.
7. Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. The google file system. In *Proc. of SOSP '03*, pages 29–43. ACM, 2003.
8. Peter JE Goss and Jean Peccoud. Quantitative modeling of stochastic systems in molecular biology by using stochastic petri nets. *PNAS*, 95(12):6750–6755, 1998.
9. M Heiner, D Gilbert, and R Donaldson. *Petri Nets for Systems and Synthetic Biology*, volume 5016 of *LNCS*, pages 215–264. Springer, 2008.
10. C. Huang and J. Ferrell. Ultrasensitivity in the mitogen-activated protein kinase cascade. *Proc. Natl. Acad. Sci.*, 93:10078–10083, 1996.
11. Marta Kwiatkowska, Gethin Norman, and David Parker. Stochastic Model Checking. In *Proc. of SFM'07*, volume 4486 of *LNCS*, pages 220–270. Springer, 2007.
12. Marco Ajmone Marsan et al. *Modelling with generalized stochastic Petri nets*. John Wiley & Sons, Inc., 1994.