RODES: A Robust-Design Synthesis Tool for Probabilistic Systems*

Radu Calinescu[1], Milan Češka[2], Simos Gerasimou[1], Marta Kwiatkowska[3], and Nicola Paoletti[4]

[1] Department of Computer Science, University of York, UK
[2] Faculty of Information Technology, Brno University of Technology, Czech Republic
[3] Department of Computer Science, University of Oxford, UK
[4] Department of Computer Science, Stony Brook University, USA

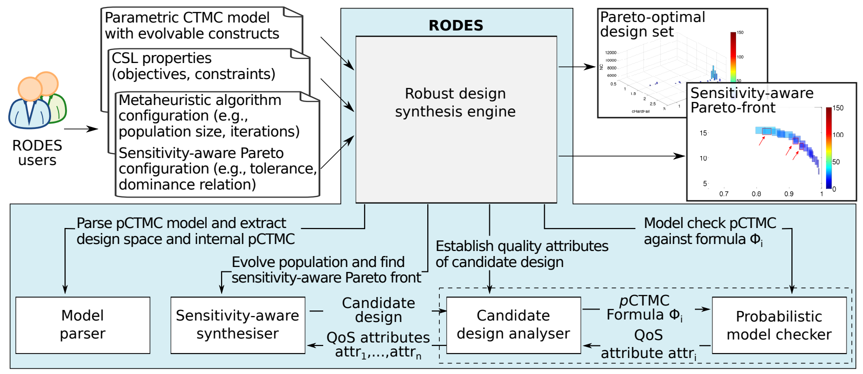
Abstract. We introduce RODES – a tool for the synthesis of probabilistic systems that satisfy strict reliability and performance requirements, are Pareto-optimal with respect to a set of optimisation objectives, and are robust to variations in the system parameters. Given the design space of a system (modelled as a parametric continuous-time Markov chain), RODES generates system designs with low sensitivity to required tolerance levels for the system parameters. As such, RODES can be used to identify and compare robust designs across a wide range of Pareto-optimal tradeoffs between the system optimisation objectives.

1 Introduction

Quantitative verification is an effective technique for analysing the quality attributes (e.g. performance and reliability) of alternative system designs from the early stages of the development lifecycle [5]. The quality attributes of interest are formalised as probabilistic temporal logic properties, and are evaluated over Markov models of different system designs. The model that achieves the best tradeoff between the quality attributes is then used as a basis for the implementation of the system. However, if this implementation cannot precisely match the parameters of the selected model, the quality attributes of the system may differ significantly from the values predicted by the quantitative verification of its model. This limits the applicability of recently proposed approaches for the automated synthesis of probabilistic system designs [4, 7].

Our RObust DEsign Synthesis (RODES) tool addresses this limitation by generating parametric continuous-time Markov chains (pCTMCs) whose transition rates are allowed to vary within small bounded intervals that correspond to user-specified tolerances for the parameters of the system. RODES implements our theoretical results from [1], which combine probabilistic model synthesis [7] and precise parameter synthesis [2] to generate Pareto-optimal sets of pCTMCs (i.e. designs) using a *sensitivity-aware Pareto dominance relation.* This relation [1] acts as a tradeoff between optimality and robustness, and enables adding robust but suboptimal designs into the Pareto-optimal sets. To this end, the relation takes into account both a set of optimisation objectives (requiring the minimisation or maximisation of certain quality attributes) and the benefit

* This work has been supported by the Czech Grant Agency grant No. GA16-17538S.


RODES users
Parametric CTMC model with evolvable constructs
CSL properties (objectives, constraints)
Metaheuristic algorithm configuration (e.g., population size, iterations)
Sensitivity-aware Pareto configuration (e.g., tolerance, dominance relation)
RODES
Robust design synthesis engine
Pareto-optimal design set
Sensitivity-aware Pareto-front
Parse pCTMC model and extract design space and internal pCTMC
Evolve population and find sensitivity-aware Pareto front
Establish quality attributes of candidate design
Model check pCTMC against formula Φ_i
Model parser
Sensitivity-aware synthesiser
Candidate design
QoS attributes attr_1,...,attr_n
Candidate design analyser
pCTMC Formula Φ_i
QoS attribute attr_i
Probabilistic model checker
Fig. 1: High-level RODES architecture

of selecting *robust designs*, i.e., designs with quality attributes insensitive to the tolerance-induced variations in the *p*CTMC transition rates.

The rest of the paper presents RODES and its extensible architecture (Section 2), and the tool scalability and applicability to systems from different domains (Section 3). The RODES code, supplementary case study material, and full experimental results are available at `https://github.com/gerasimou/RODES`.

2 RODES Functionality and Architecture

RODES (Fig. 1) is a Java-based tool with the inputs described below.

1) A *p*CTMC model of the entire design space, expressed in the modelling language of the model checker PRISM [10] extended with the constructs

$$\textsf{evolve double } k\ [k_{\mathsf{min}}..k_{\mathsf{max}}] \quad (1)$$

$$\textsf{evolve int } d\ [d_{\mathsf{min}}..d_{\mathsf{max}}] \quad (2)$$

$$\textsf{evolve module } \mathit{ComponentName} \quad (3)$$

which are used to specify ranges for the continuous and discrete parameters of the system, and alternative component designs, respectively. A RODES design is also a *p*CTMC, obtained from the design-space *p*CTMC by constraining its continuous parameters (1) to small bounded intervals

$$[k_0 - \delta, k_0 + \delta] \subset [k_{\mathsf{min}}, k_{\mathsf{max}}], \quad (4)$$

fixing the values of its discrete parameters (2), and selecting one of the alternative designs (3) for each distinct *ComponentName* value.

2) Continuous stochastic logic (CSL) properties specifying the optimisation objectives and constraints for the quality attributes of the system.

3) Configuration parameters for the design-search metaheuristic algorithm, and the following parameters of the sensitivity-aware Pareto dominance relation:
   - a small *tolerance* $\gamma > 0$ for each continuous parameter (1) such that the allowed parameter-value variation δ from (4) is $\delta = \gamma(k_{\mathsf{max}} - k_{\mathsf{min}})$;
   - a small *sensitivity coefficient* $\epsilon \geq 0$ such that a design needs to have $(1+\epsilon)$ times better quality attributes to dominate a more robust design.

The operation of RODES is managed by a *Robust-design synthesis engine* (Fig. 1). First, a *Model parser* (built using the Antlr parser generator, `www.`

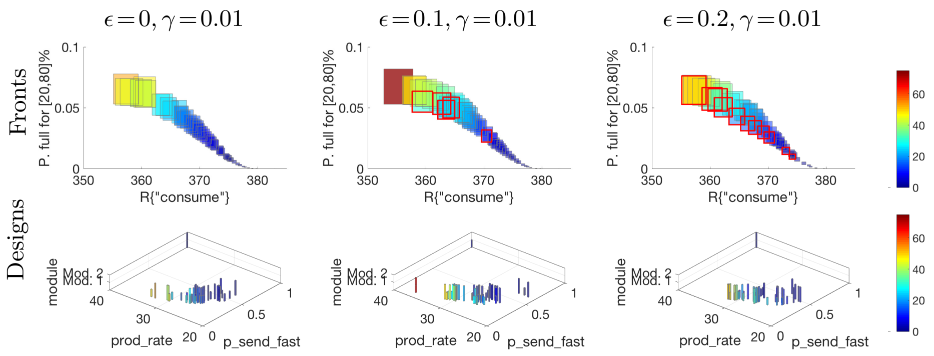


Fig. 2: Sensitivity-aware Pareto fronts (top) for the producer-consumer model, and corresponding synthesised Pareto-optimal designs (bottom). Boxes represent quality-attribute regions, coloured by sensitivity (red: sensitive, blue: robust). Red-bordered boxes indicate sub-optimal robust designs. Designs are compared based on the worst-case quality attribute value (i.e. lower-left corner of each box).

`antlr.org`) preprocesses the design-space pCTMC. Next, a *Sensitivity-aware synthesiser* employs the jMetal Java framework for multi-objective optimisation with metaheuristics (`jmetal.github.io/jMetal`) to evolve an initially random population of *candidate designs*, generating a close approximation of the sensitivity-aware Pareto front. This involves using a *Candidate design analyser*, which invokes the probabilistic model checker PRISM-PSY [3] to obtain the ranges of values for the relevant quality attributes of candidate designs through precise parameter synthesis. The Pareto front and corresponding Pareto-optimal set of designs are then plotted using MATLAB/Octave scripts, as shown in Fig. 2.

A key feature of RODES is its modular architecture. The Sensitivity-aware synthesiser supports several metaheuristics algorithms, including variants of genetic algorithms and swarm optimisers. Further, the sensitivity-aware Pareto dominance relation can be adapted to match better the needs of the system under development (e.g., by comparing designs based on the worst, best or average quality attribute values). Finally, different solvers could be plugged in the probabilistic model checker component, including e.g. the GPU-accelerated version of PRISM-PSY [3], or parameter synthesis tools for DTMCs [6].

3 Case Studies and Experimental Results

We evaluate RODES in three case studies: a variant of the producer-consumer problem;[5] a workstation cluster [9]; and a replicated file system used by Google's search engine [8]. Runtimes (Table 1) depend on the number of evaluations (using more typically improves the quality of the Pareto fronts) and by the time required to analyse a candidate design. These runtimes were obtained using the sequential version of PRISM-PSY, but we are currently integrating the GPU-accelerated version, which will significantly improve the scalability of the tool [3].

Fig. 2 shows Pareto fronts and designs obtained for a producer-consumer model comprising a slow high-capacity buffer and a fast buffer of small capac-

[5] E.W. Dijkstra. "Information Streams Sharing a Finite Buffer" Inf. Proc. Letters, 1972. The model can be found at `https://github.com/gerasimou/RODES/wiki`.

Table 1: Time (mean ± SD) for the synthesis using 10,000 evaluations. **variant**: values of scenario parameters. **#states** (**#trans.**): number of states (transitions) of the underlying pCTMC. $|K|$: number of continuous parameters.

Model/	Google File System ($\|K\|$=2)			Workstation cluster ($\|K\|$=2)			Prod.-cons.
variant:	S=5000	S=10000	S=20000	N=9	N=12	N=15	($\|K\|$=2)
#states	1323	1893	2406	3440	5876	8960	5632
#trans.	7825	11843	15545	18656	32204	49424	21884
Time (m)	104±4	149±4	180±9	185±19	191±27	205±42	29±1

ity. The design space has two continuous parameters—overall production rate, prod_rate, and probability of using the fast buffer, p_send_fast; and a discrete parameter that selects between two alternative designs (3) so either packets stay in the designated buffers, or packets in the slow buffer are redirected to the fast buffer with probability proportional to the slow buffer occupancy. We aim to maximise two objectives: the expected system throughput (x-axis), and the probability that both buffers are utilised at between 20–80% of their capacity (y-axis). Fig. 2 shows results for tolerance $\gamma = 0.01$ and for several values of the sensitivity coefficient ϵ (cf. Section 2). As expected, the number of slightly sub-optimal but more robust solutions increases with ϵ. The sensitivity-aware Pareto fronts provide unique insights into the system behaviour, and facilitates the selection of designs with a wide range of robustness levels, making RODES an effective tool for the synthesis of robust designs from multi-objective specifications.

References

[1] R. Calinescu, M. Češka, S. Gerasimou, M. Kwiatkowska, and N. Paoletti. "Designing Robust Software Systems through Parametric Markov Chain Synthesis". In: *ICSA*. 2017, pp. 131–140.

[2] M. Češka, F. Dannenberg, N. Paoletti, et al. "Precise Parameter Synthesis for Stochastic Biochemical Systems". In: *Acta Informatica* (2016), pp. 1–35.

[3] M. Češka, P. Pilař, N. Paoletti, L. Brim, and M. Kwiatkowska. "PRISM-PSY: Precise GPU-accelerated parameter synthesis for stochastic systems". In: *TACAS*. 2016, pp. 367–384.

[4] T. Chen, E. M. Hahn, T. Han, et al. "Model Repair for Markov Decision Processes". In: *TASE*. 2013, pp. 85–92.

[5] L. Cheung, R. Roshandel, N. Medvidovic, and L. Golubchik. "Early Prediction of Software Component Reliability". In: *ICSE*. 2008, pp. 111–120.

[6] C. Dehnert, S. Junges, N. Jansen, et al. "PROPhESY: A PRObabilistic ParamEter SYnthesis Tool". In: *CAV*. 2015, pp. 214–231.

[7] S. Gerasimou, G. Tamburrelli, and R. Calinescu. "Search-Based Synthesis of Probabilistic Models for QoS Software Engineering". In: *ASE*. 2015.

[8] S. Ghemawat, H. Gobioff, and S.-T. Leung. "The Google File System". In: *SOSP*. 2003, pp. 29–43.

[9] B. R. Haverkort, H. Hermanns, and J.-P. Katoen. "On the use of model checking techniques for dependability evaluation". In: *SRDS*. 2000.

[10] M. Kwiatkowska, G. Norman, and D. Parker. "PRISM 4.0: Verification of Probabilistic Real-time Systems". In: *CAV*. 2011, pp. 585–591.