Poster: AMYTISS: A Parallelized Tool on Automated Controller Synthesis for Large-Scale Stochastic Systems

Abolfazl Lavaei*
Department of Computer Science
Ludwig Maximilian University of Munich
lavaei@lmu.de

Mahmoud Khaled*
Department of Electrical Engineering
Technical University of Munich
khaled.mahmoud@tum.de

Sadegh Soudjani
School of Computing
Newcastle University
Sadegh.Soudjani@newcastle.ac.uk

Majid Zamani
Department of Computer Science
University of Colorado Boulder
Ludwig Maximilian University of Munich
Majid.Zamani@Colorado.EDU

KEYWORDS

Automated Controller Synthesis, Discrete-Time Stochastic Systems, Parallel Algorithms, High Performance Computing Platform.

ACM Reference Format:
Abolfazl Lavaei*, Mahmoud Khaled*, Sadegh Soudjani, and Majid Zamani. 2020. Poster: AMYTISS: A Parallelized Tool on Automated Controller Synthesis for Large-Scale Stochastic Systems. In *23rd ACM International Conference on Hybrid Systems: Computation and Control (HSCC '20), April 22–24, 2020, Sydney, NSW, Australia.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3365365.3383469

1 ABSTRACT

Large-scale stochastic systems have recently received significant attentions due to their broad applications in various safety-critical systems such as *traffic networks* and *self-driving cars*. In this poster, we describe the software tool AMYTISS, implemented in C++/OpenCL, for designing correct-by-construction controllers for large-scale discrete-time stochastic systems. This tool is employed to (i) build *finite* Markov decision processes (MDPs) as finite abstractions of given original systems, and (ii) synthesize controllers for the constructed finite MDPs satisfying bounded-time safety, reachability, and reach-avoid specifications. In AMYTISS, scalable parallel algorithms are designed such that they support the parallel execution within CPUs, GPUs and hardware accelerators (HWAs). Unlike all existing tools for stochastic systems, AMYTISS can utilize high-performance computing (HPC) platforms and cloud-computing services to mitigate the effects of the state-explosion problem, which is always present in analyzing large-scale stochastic systems. We benchmark AMYTISS against the most recent tools in the literature using several physical case studies including robot examples, room temperature and road traffic networks. We also apply our algorithms to a 3-dimensional autonomous vehicle and a 7-dimensional *nonlinear* model of a BMW 320i car by synthesizing autonomous parking controllers.

Related works. There exist a limited number of software tools on the verification and synthesis of stochastic systems with different classes of models. SReachTools [1] performs the stochastic reachability analysis for linear, potentially time-varying, discrete-time stochastic systems. FAUST^2 [2] generates formal abstractions for continuous-space discrete-time stochastic processes, and performs the verification and synthesis for safety and reachability specifications. However, FAUST^2 is originally implemented in MATLAB and handles finite-horizon specifications. StocHy [3] deals with a class of discrete-time stochastic hybrid systems, constructs finite abstractions, and performs the verification and synthesis for both finite- and infinite-horizon safety and reachability specifications. AMYTISS differs from FAUST^2 and StocHy in two main directions. First, AMYTISS implements novel parallel algorithms and data structures targeting HPC platforms to reduce the undesirable effects of the state-explosion problem. Accordingly, it is able to perform the parallel execution in different heterogeneous computing platforms including CPUs, GPUs and hardware accelerators (HWAs). Whereas, FAUST^2 and StocHy can only run serially in one CPU, and consequently, they are limited to small systems. Additionally, AMYTISS can handle the abstraction construction and controller synthesis for two and a half player games (e.g., stochastic systems with bounded disturbances), whereas FAUST^2 and StocHy only handle one and a half player games (disturbance-free systems).

Unlike all existing tools, AMYTISS offers highly scalable, distributed execution of parallel algorithms utilizing all available processing elements (PEs) in any heterogeneous computing platform. To the best of our knowledge, AMYTISS is the only tool of its kind for continuous-space stochastic systems that is able to utilize simultaneously all types of compute units (CUs).

A comparison between AMYTISS, FAUST^2 and StocHy based on their native features is provide in Table 1.

Main Contribution. AMYTISS is an *open-source* and self-contained tool and requires only a modern C++ compiler. It supports three major operating systems: Windows, Linux and Mac OS. The source of AMYTISS and detailed instructions on its building and running can be found in:

https://github.com/mkhaled87/pFaces-AMYTISS

The main merits of this work are:

(1) We propose a novel data-parallel algorithm for constructing finite MDPs from discrete-time stochastic systems and storing them in efficient distributed data containers.
(2) We propose parallel algorithms for synthesizing discrete controllers using the constructed MDPs to satisfy safety, reachability, or reach-avoid specifications. More specifically,

*Both authors have contributed equally to the development of the tool. This work was supported in part by the H2020 ERC Starting Grant AutoCPS (grant No. 804639).

Table 1: Comparison between AMYTISS, FAUST^2 and StocHy based on native features.

Aspect	FAUST^2	StocHy	AMYTISS
Platform	CPU	CPU	All platforms
Algorithms	Serial on HPC	Serial on HPC	Parallel on HPC
Model	Stochastic control systems: linear, bilinear	Stochastic hybrid systems: linear, bilinear	Stochastic control systems: nonlinear
Specification	Safety, reachability	Safety, reachability	Safety, reachability, reach-avoid
Stochasticity	Additive noise	Additive noise	Additive & multiplicative noises
Distribution	Normal, user-defined	Normal, user-defined	Normal, uniform, exponential, beta, user-defined
Disturbance	Not supported	Not supported	Supported

we introduce novel parallel algorithms for the iterative computation of Bellman equation in the standard dynamic programming [4].

(3) Unlike the existing tools in the literature, AMYTISS accepts bounded disturbances and natively supports both additive and multiplicative noises with different distributions including normal, uniform, exponential, and beta.

2 PARALLEL ALGORITHM FOR CONSTRUCTING FINITE MDPS

We propose a novel parallel algorithm to efficiently construct and store a probability transition matrix $\hat{T}_{\mathsf{x}}$ (cf. [5, Algorithm 1] for $\hat{T}_{\mathsf{x}}$).

2.1 Less Memory for Constructing $\hat{T}_{\mathsf{x}}$

In our proposed algorithm, we significantly reduce the memory usage by setting a cutting probability threshold $\gamma \in [0,1]$ to control how many partition elements around the mean of the system should be stored. Such an approximation allows controlling the sparsity of the columns of $\hat{T}_{\mathsf{x}}$. The closer the value of γ to zero, the more accurate $\hat{T}_{\mathsf{x}}$ in representing the transitions of finite MDPs. On the other hand, the closer the value of γ to one, less state values need to be stored as columns in $\hat{T}_{\mathsf{x}}$.

3 PARALLEL SYNTHESIS OF CONTROLLERS

We employ a parallel standard dynamic programming to synthesize controllers for the constructed finite MDPs satisfying safety, reachability, or reach-avoid properties.

3.1 On-the-Fly Construction of $\hat{T}_{\mathsf{x}}$

In AMYTISS during the synthesis process, we also propose another technique that further reduces the required memory for computing $\hat{T}_{\mathsf{x}}$. We refer to this approach as *on-the-fly abstractions* (OFA). In OFA version of our proposed algorithm, we skip computing and storing the probability transition matrix $\hat{T}_{\mathsf{x}}$. We instead compute the required entries of $\hat{T}_{\mathsf{x}}$ on-the-fly as they are needed for the synthesis part via the standard dynamic programing. This reduces the required memory for $\hat{T}_{\mathsf{x}}$ but at the cost of repeated computation of their entries in each time step from 1 to a finite time horizon T_d. This gives the user an additional control over the trade-off between the computation time and memory usage.

3.2 Supporting Multiplicative Noises and Practical Distributions

In addition to additive noises, AMYTISS natively supports multiplicative noises and some practical distributions such as normal, uniform, exponential, and beta distributions. Since AMYTISS is designed for extensibility, it allows also for customized distributions. Users need to specify their desired probability density functions and hyper-rectangles enclosing their supports so that AMYTISS can include them in the parallel computation of $\hat{T}_{\mathsf{x}}$.

AMYTISS also supports multiplicative noises as introduced in [6, equation (1)]. Currently, the memory reduction technique of Subsection 2.1 is disabled for systems with multiplicative noises. This means users should expect larger memory requirements for systems with multiplicative noises. However, users can still benefit from the proposed OFA version to compensate for the increase in the memory requirement. We plan to include this feature for multiplicative noises in a future update of AMYTISS.

4 BENCHMARKING AND CASE STUDIES

To show the applicability of our results to large-scale systems, we apply our algorithms to several physical case studies. We synthesize controllers for 3- and 5-dimensional *room temperature networks* to keep the temperatures in a comfort zone. Furthermore, we synthesize controllers for *road traffic networks* with 3 and 5 dimensions to maintain the density of traffic below some desired level. In addition, we apply our algorithms to a 2-dimensional nonlinear robot and synthesize controllers satisfying safety and reach-avoid specifications. Finally, we consider 3- and 7-dimensional *nonlinear* models of autonomous vehicles and synthesize reach-avoid controllers to automatically park the vehicles.

We make comparisons between AMYTISS, FAUST^2, and StocHy for all the aforementioned case studies. AMYTISS outperforms FAUST^2 and StocHy in all the case studies with maximum speedups respectively up to 1680000 and 676000 times (in the case of running AMYTISS on an NVIDIA Tesla V100 Machine with 5120 processing elements and 0.8 GHz frequency). Moreover, AMYTISS is the only tool that can utilize the available HW resources for stochastic systems. The OFA feature in AMYTISS reduces dramatically the required memory, while still solves the problems in a reasonable time. FAUST^2 and StocHy fail to solve many of the problems since they lack the native support for nonlinear systems, they require large amounts of memory, or they do not finish computing within 24 hours.

REFERENCES

[1] A. P. Vinod, J. D. Gleason, and M. M. Oishi, "SReachTools: A MATLAB stochastic reachability toolbox," in *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*, 2019, pp. 33–38.

[2] S. Soudjani, C. Gevaerts, and A. Abate, "FAUST^2: Formal abstractions of uncountable-state stochastic processes," in *TACAS'15*, ser. Lecture Notes in Computer Science, 2015, vol. 9035, pp. 272–286.

[3] N. Cauchi and A. Abate, "StocHy: Automated verification and synthesis of stochastic processes," in *TACAS'19*, ser. Lecture Notes in Computer Science, 2019, vol. 11428, pp. 247–264.

[4] S. Soudjani, "Formal abstractions for automated verification and synthesis of stochastic systems," Ph.D. dissertation, Technische Universiteit Delft, The Netherlands, 2014.

[5] A. Lavaei, S. Soudjani, and M. Zamani, "From dissipativity theory to compositional construction of finite Markov decision processes," in *Proceedings of the 21st ACM International Conference on Hybrid Systems: Computation and Control*, 2018, pp. 21–30.

[6] W. Li, E. Todorov, and R. E. Skelton, "Estimation and control of systems with multiplicative noise via linear matrix inequalities," in *Proceedings of the 2005, American Control Conference, 2005.*, 2005, pp. 1811–1816.