TAPAS: a Tool for Stochastic Evaluation of Large Interdependent Composed Models with Absorbing States

✉Giulio Masetti,
ISTI-CNR, Pisa, Italy,
giulio.masetti@isti.cnr.it,
Leonardo Robol,
University of Pisa,
leonardo.robol@unipi.it

Silvano Chiaradonna,
Felicita Di Giandomenico,
ISTI-CNR, Pisa,Italy,
silvano.chiaradonna,felicita.digiandomenico@isti.cnr.it

ABSTRACT

TAPAS is a new tool for efficient evaluation of dependability and performability attributes of systems composed of many interconnected components. The tool solves homogeneous continuous time Markov chains described by stochastic automata network models structured in submodels with absorbing states. The measures of interest are defined by a reward structure based on submodels composed through transition-based synchronization. The tool has been conceived in a modular and flexible fashion, to easily accommodate new features. Currently, it implements an array of state-based solvers that addresses the state explosion problem through powerful mathematical techniques, including Kronecker algebra, Tensor Trains and Exponential Sums. A simple, yet representative, case study is adopted, to present the tool and to show the feasibility of the supported methods, in particular from memory consumption point of view.

Keywords

Stochastic modeling and evaluation, stochastic automata network, absorbing states, dependability, performability, state explosion problem

1. INTRODUCTION AND RELATED WORK

This paper proposes the tool TAPAS (Tensor Algorithms for Performability Analysis of large Systems) for efficient evaluation of dependability and performability attributes of complex systems composed of a large number of interconnected components. TAPAS solves homogeneous Continuous Time Markov Chains (CTMCs) described by Stochastic Automata Network (SAN) [1] models structured in submodels with absorbing states.

The measures of interest [2] supported by TAPAS are k-moments (e.g., mean or variance) of: 1) instant-of-time or interval-of-time reward variables to absorbing state, or 2) conditional interval-of-time reward variables to absorbing state, given that the CTMC eventually absorbs into a subset of states. They are defined by a reward structure based on SAN automata (submodels) composed through transition-based synchronization. Examples of such measures are: 1) the reward accumulated to absorption in the set of all absorbing states, e.g., the Mean Time To Failure (MTTF) or the Mean Reward To Absorption (MRTA), 2) the conditional reward accumulated to absorption, given that the model eventually absorbs into a subset of states, or 3) the probability that the CTMC eventually absorbs into a set of states. It is possible to show [3, 4] that these measures can be evaluated solving a sequence of linear systems, which may require manipulating both large matrices and vectors.

In the context of stochastic state-space models, system complexity and largeness are typically managed through the modularity and composition of individual submodels [5, 6, 7, 8, 9]. This offers advantages in terms of both model definition and numerical evaluation. However, even when the overall system model results from the composition of submodels with a small number of states, increasing the number of submodels or the interdependencies among them makes the numerical analysis very challenging for what concerns the state-space explosion.

An approach to tackle this problem consists in representing and manipulating the state-space implicitly, i.e., not listing the states, but describing them through more complex data structures [10, 11].

In this direction, recently the new efficient numerical solution methodology KAES (Kronecker Algebra Exponential Sums) for CTMCs with absorbing states has been proposed [3, 4], resorting to powerful mathematical technologies such as Kronecker algebra [12], Tensor Trains [13] and Exponential sums [14]. Trough Kronecker algebra [12] it is possible to implicitly manipulate the CTMC under analysis and the associated reward structure (i.e., the CTMC reward model), when the CTMC is defined as a SAN [1] composed through transition-based synchronization of stochastic automata [3, 4]. As shown in [3, 4], through Tensor Trains algebra both matrices and vectors can be implicitly represented. The Exponential Sums [14] can be used by the preconditioner to get an approximation of the linear system solution (as shown in Section 6 with the method *amen*) or to solve the system together with an *ad hoc* iteration step deduced from a (non-regular) splitting in order to advance over the method presented in [3] (as described in Section 6 with the methods *tt-regular-splitting* and *tt-expsumst*).

TAPAS implements an array of state-based solvers that addresses the state explosion problem, based on variants of KAES, including methods that have not been published yet and also some methods that construct explicitly the state-space (these last, mainly for cross-validation and performance comparison purposes), as shown in Table 2.

TAPAS has been implemented in the MATLAB evalua-

Copyright is held by author/owner(s).

tion environment [15] and provides a library of MATLAB functions. It has been conceived in a modular and flexible fashion, to be easily expanded to incorporate new features.

A simple, yet representative, case study is adopted as a running example (Figure 1) for the proposed tool.

TAPAS advances current tools for conducting similar analyses, which are based on implicit representation and manipulation of the state space, by extending the implicit representation to the solution vector, in addition to the descriptor matrix. To the best of authors' knowledge, TAPAS is the first tool to tackle complete implicit model representation. Moreover, it addresses the analysis of limiting properties of CTMCs with absorbing states, which is another aspect that makes TAPAS rather peculiar with respect to existing alternatives for the analysis contexts of interest in this paper. Among the most popular tools, those that appear more related to TAPAS are GreatSPN[1] [16], Möbius[2] [17], Nsolve[3] [18], PEPS[4] [10] and SHARPE[5] [19]. However, as already pointed out, the lower degree of implicit representation they allow leaves open further performance improvement, which is actually the gap that TAPAS attempts to cover.

TAPAS is an experimental and academic open-source tool and can be downloaded from GitHub under BSD license. Documentation, background and examples are available at `https://github.com/numpi/tapas`. The implemented methods at the moment rely on the `TT-Toolbox` [13].

The rest of the paper is organized as follows. In Section 2, the context and the measures of interest are described. In Section 3, a brief recap on the SAN formalism is provided and the SAN model used as case study is introduced. Section 4 describes the format of the model representation required by the tool, based on the defined case study. A tutorial (more details are in `examples/tutorial.m` inside the TAPAS package) on how to run the model is in Section 5. The implemented methods are described in Section 6. Section 7 is a teaser of performance potentialities of TAPAS. Conclusions and future work are sketched in Section 8.

2. CONTEXT

Consider the time-homogeneous CTMC $\{X(t) \in \mathcal{S},\, t \geq 0\}$ with discrete (finite) state space $\mathcal{S} = \mathcal{T} \sqcup \mathcal{A}$, where $\mathcal{T}$ is the set of transient states and $\mathcal{A}$ the set of absorbing ones. In reliability models, $X(0)$ is usually a "working" or "up" state with probability 1, thus the initial probability vector π_0 has a single nonzero entry, the one indexed by $X(0)$. In general, the entries of π_0 can be nonnegative reals that sums to one.

Among the dependability properties or performability measures that are usually of interest, here the focus is on those involving limiting behavior of X, such as the MTTF or the MTTF conditioned to the event "X has been absorbed in $\mathcal{B} \subset \mathcal{A}$". More formally, the performability measures [20, 2] that can be evaulated through TAPAS are those defined through a given reward vector $r \in \mathbb{R}^{|\mathcal{S}|}$ with zero entries on $\mathcal{A}$, in terms of the moments of the reward variable

$$Y_\infty := \int_0^\infty r_{X(t)} dt,$$

where X selects which entry of r contributes to the integral. The moments will be called $\mathcal{M}_k := \mathbb{E}\big[Y_\infty^k\big]$. Of particular interest are the MRTA $\mathcal{M}_1$, the variance

$$Var(Y_\infty) = \mathcal{M}_2 - \big(\mathcal{M}_1\big)^2 \qquad (1)$$

and possibly an approximation of order k of the distribution of Y_∞.

Consider systems where there are multiple causes of failure or (in a safety critical system) can be relevant to distinguish between safe and unsafe shutdown or (in a fault-tolerant system) can be relevant to distinguish a failure due to imperfect coverage from a failure due to the exhaustion of redundancy [21]. Thus, given a subset of absorbing states $\mathcal{B} \subset \mathcal{A}$, define $Y_{\infty|\mathcal{B}}$ as the conditional reward accumulated until absorption given that some absorbing markings $\mathcal{B}$ are reached. The measures of interests are then the conditional mean $\text{MRTA}_{|\mathcal{B}} := \mathbb{E}[Y_{\infty|\mathcal{B}}]$ and the probability $\pi_{\mathcal{B}}(\infty)$ that the model eventually absorbs into $\mathcal{B}$.

It is possible to show [4] that all the mentioned measures can be evaluated solving a sequence of linear systems followed by elementary post processing operations. In particular, $\mathcal{M}_k$ requires the solution of

$$\begin{cases} (Q-S)x^{(1)} = r, \\ (Q-S)x^{(i)} = \text{diag}(r)x^{(i-1)}, \quad \text{for } i = 2, \dots, k, \end{cases}$$

followed by the dot product

$$\mathcal{M}_k = k!(-1)^k \pi(0) \cdot x^{(k)}, \qquad (2)$$

$\pi_{\mathcal{B}}(\infty)$ the solution of

$$(Q-S)x^{(1)} = Qe_{\mathcal{B}}$$

followed by the dot product

$$\pi_{\mathcal{B}}(\infty) = -\pi(0) \cdot x^{(1)},$$

and $\text{MRTA}_{|\mathcal{B}}$ the solution of

$$\begin{cases} (Q-S)x^{(1)} &= Qe_{\mathcal{B}}, \\ (Q-S)x^{(2)} &= \text{diag}(r)x^{(1)}, \end{cases}$$

followed by two dot products and a division

$$\text{MRTA}_{|\mathcal{B}} = \frac{\pi(0) \cdot x^{(2)}}{-\pi(0) \cdot x^{(1)}}, \qquad (3)$$

where S is a shift matrix (two ways to define it are shown in [3, 4]), $x^{(i)}$ are column vectors and $e_{\mathcal{B}}$ is the column vector with binary entries whose components are nonzero only if indexed by the elements of $\mathcal{B}$.

When $\mathcal{S}$ grows big, it is not feasible to explicitly list all the states and assemble π_0, r and Q, even in a sparse form. Thus, both the matrix and the vectors have to be treated implicitly, meaning that only chunks of information are stored and manipulated by the linear system solvers to evaluate the measures of interest.

[1]`https://github.com/greatspn/SOURCES.git`
[2]`https://www.mobius.illinois.edu`
[3]`https://ls4-www.cs.tu-dortmund.de/download/buchholz/Programs/Nsolve`
[4]`http://www-id.imag.fr/Logiciels/peps`
[5]`http://sharpe.pratt.duke.edu`

3. MODELING FORMALISM

In order to give an implicit representation of the CTMC under study, the SAN [1] formalism is adopted to define the CTMC model. The SAN reward model is obtained by extending the SAN model with the reward vector r.

In our context, a SAN consists of interacting individual stochastic automata (submodels) and each individual automaton represents a CTMC with absorbing states. Accordingly, individual automata are defined by states and changes of states, i.e., (exponential) transitions between pairs of states. Interactions among automata occur through transition synchronization. A transition between two states can be local to a given automaton or of synchronization (shared) among different automata. A local transition governs the manner in which a single automaton moves from one state to the next. A synchronization transition shared among different automata governs the manner in which all the involved automata move simultaneously from a state to the next, i.e., in a single atomic transition. A transition of an automaton can be synchronized with one or more transitions of another automaton, and vice versa. Transitions are represented with the same label. An automaton moves from a state to the next by means of a synchronization transition ξ, if all the automata where at least one synchronization transition ξ is defined move from a state to the next by means of a transition ξ. The state of an automaton at any time t is given by the state of each of its constituent automata.

Formally, the SAN model is structured in n interacting submodels, namely

$$\tilde{X}(t) := (X^{(1)}(t), \dots, X^{(n)}(t)),$$

where $X^{(i)}(t) \in \mathcal{S}^{(i)}$ and $\mathcal{S}^{(1)} \times \cdots \times \mathcal{S}^{(n)}$ is the potential state-space. An example of SAN model composed by $n = 3$ interacting submodels A_1, A_2 and A_3 is the state-transition diagram shown in Figure 1, where each state is represented by a circle, and local and synchronization transitions are represented as filled and dashed arrows, respectivley. Transition rates, as well as the rate and label pair, are associated with local and synchronization transitions, respectively. Thus, in Figure 1, by means of the transition ξ

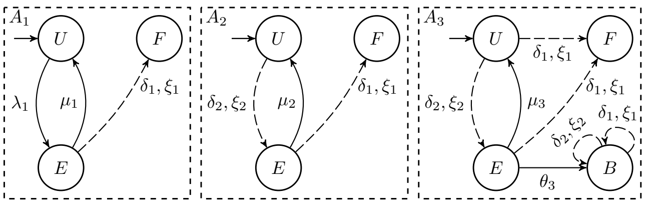

Figure 1: SAN model example. Dashed arrows are synchronization transitions.

with rate δ_1, the submodels A_1, A_2 and A_3 can move from the state E to F, or alternatively, the submodels A_1 and A_2 can move from E to F when the submodel A_3 moves from U to F or from B to B.

With this notation, trough the Kronecker algebra [12], it is possible to decompose the probability vector and the infinitesimal generator matrix as

$$\begin{aligned}\tilde{\pi}(t) &= \pi^{(1)}(t) \otimes \pi^{(2)}(t) \otimes \ldots \otimes \pi^{(n)}(t),\\ \tilde{Q} &= R + W + \Delta,\end{aligned}$$

where

$$R = \bigoplus_{i=1}^{n} R^{(i)}, \qquad W = \sum_{\xi \in \mathcal{ST}} \bigotimes_{i=1}^{n} W^{(\xi,i)}, \tag{4}$$

and R and W are $|\mathcal{S}^{(i)}| \times |\mathcal{S}^{(i)}|$ matrices that represent, respectively, the local contribution and the synchronization contribution (being $\mathcal{ST}$ the set of the synchronization transitions). Δ is the diagonal matrix defined as $\Delta = -\mathrm{diag}((R + W)e)$, where e denotes the column vector with all the entries equal to 1. The operators $\oplus$ and $\otimes$ are the *Kronecker sum* and *product*, respectively. A complete characterization can be found in [12].

Among the available formats for the implicit representation and manipulation of the above matrices and vectors, and in particular, Q, π_0 and r, the Tensor-Trains format (also known as matrix product state) [3] is exploited in the main methodologies supported by the tool TAPAS to numerically solve the SAN reward model.

4. INPUT FORMAT

TAPAS is a library of MATLAB functions that compute the measures of interest. As stated in Section 2, the user has to specify the implicit representation of $\tilde{Q}$ through the definition of R and W, the initial probability vector $\tilde{\pi}(0)$, the reward vector $\tilde{r}$ and the column vector $\tilde{e}_{\mathcal{B}}$. For the example depicted in Figure 1, R and W are

$$\begin{array}{cccc}
R^{(1)} & R^{(2)} & R^{(3)} & \\
\begin{pmatrix} 0 & \lambda_1 & 0 \\ \mu_1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} &
\begin{pmatrix} 0 & 0 & 0 \\ \mu_2 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} &
\begin{pmatrix} 0 & 0 & 0 & 0 \\ \mu_3 & 0 & 0 & \vartheta_3 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} &
\begin{matrix} U \\ E \\ F \\ B \end{matrix} \\
W^{(\xi_1,1)} & W^{(\xi_1,2)} & W^{(\xi_1,3)} & \\
\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & \delta_1 \\ 0 & 0 & 0 \end{pmatrix} &
\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} &
\begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} &
\begin{matrix} U \\ E \\ F \\ B \end{matrix} \\
W^{(\xi_2,1)} & W^{(\xi_2,2)} & W^{(\xi_2,3)} & \\
\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} &
\begin{pmatrix} 0 & \lambda_2 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} &
\begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} &
\begin{matrix} U \\ E \\ F \\ B \end{matrix} \\
U \quad E \quad F & U \quad E \quad F & U \quad E \quad F \quad B &
\end{array}$$

$\tilde{\pi}(0)$ is $(1,0,0) \otimes (1,0,0) \otimes (1,0,0,0)$, and $\tilde{r}$ and $\tilde{e}_{\mathcal{B}}$ depend on the measure. In [4] it has been shown that, given a reward vector, it is always possible to rewrite it as

$$\tilde{r} = \sum_i \bigotimes_{j=1}^{n} \tilde{r}^{i,j}$$

This enhances the performance of those methods that rely on the Tensor Trains format. Similarly, $\tilde{e}_{\mathcal{B}}$ is expressible as the sum of Kronecker products of binary column vectors.

As an example, call F the binary column vector that indicates the state (F, F, F) of Figure 1, i.e., $F = (0,0,1)^T \otimes (0,0,1)^T \otimes (0,0,1,0)^T$, and B the one that indicates (F, F, B), i.e., $B = (0,0,1)^T \otimes (0,0,1)^T \otimes (0,0,0,1)^T$. Then, Y_∞ is the Time To Failure when the reward is defined as

$$\tilde{r} = (1,1,1)^T \otimes (1,1,1)^T \otimes (1,1,1,1)^T - F - B,$$

argument	description
inv	linear system solver
tta_variance	evaluate Equation (1)
momentk	evaluate Equation (2)
cond_etta	evaluate Equation (3)

Table 1: First argument of `eval_measure`.

and MTTF is $\mathcal{M}_1$. To evaluate $\pi_{\mathcal{B}}(\infty)$ and $\mathrm{MTTF}_{|\mathcal{B}}$ when $\mathcal{B} = \{(\mathrm{F}, \mathrm{F}, \mathrm{F})\}$, just define $e_{\mathcal{B}} = \mathrm{F}$; for $\mathcal{B} = \{(\mathrm{F}, \mathrm{F}, \mathrm{B})\}$, $e_{\mathcal{B}} = \mathrm{B}$.

5. TUTORIAL

In this section, a few details are provided on how to setup TAPAS, exploit auxiliary functions to define R, W, $\tilde{\pi}(0)$, $\tilde{r}$ and $\tilde{e}_{\mathcal{B}}$, and evaluate $\mathcal{M}_k$, $\pi_{\mathcal{B}}(\infty)$ and $\mathrm{MRTA}_{|\mathcal{B}}$. The model depicted in Figure 1 is taken as an illustrative example. A complete tutorial can be found in `examples/tutorial.m` inside the TAPAS package.

Assuming that the TT-Toolbox has been cloned in the same folder of TAPAS, the setup is done through the call of the `setup` functions, namely:

```
cd TT-Toolbox/
setup
cd ../tapas/
setup
```

R and W are defined as MATLAB cell arrays of matrices in Tensor Trains format, namely:

```
R = cell(1, n); W = cell(2, n);
W{1, 2} = tt_matrix([0,0,0; 0,0,1; 0,0,0]);
```

where, as an example, $W^{(\xi_1,2)}$ of Section 4 is defined. The `ktt_ej` function, that takes the array of $|\mathcal{S}^{(i)}|$ and an array of indices (each indicating the nonzero entry of the standard basis vector), can be exploited to define $\tilde{\pi}(0)$, as in

```
ss = [3, 3, 4]; pi0 = ktt_ej(sz, ones(1, n));
```

Once the result tolerance `tol` and the tolerance `ttol` exploited by `round` are fixed (e.g., to evaluate MTTF), $\tilde{r}$ can be defined through

```
F = [3 3 3]; B = [3 3 4];
r = round(ktt_ones(ss) - ktt_ej(ss, F)
          - ktt_ej(ss, B), ttol);
```

where `ktt_ones` is the column vector of all ones of the appropriate dimension. The absorbing states are represented by:

```
absorbing_states = [F; B];
```

The evaluation is finally obtained calling `eval_measure` as follows:

```
m = eval_measure('inv', pi0, r , R, W, 'debug',
    true, ..., 'algorithm', 'amen', 'ttol', ttol,
    ..., 'absorbing_states', absorbing_states);
```

where the measure is selected, among those described in Section 2, by specifying the first argument as in Table 1, and the algorithms, i.e., methods, are those listed in Table 2.

Tensor trains representations can be efficiently stored only if the TT-ranks remain bounded by small integers. However, every arithmetic operation would normally increase the ranks, so the tensors and tensor operators are repeatedly truncated to a given tolerance to ensure they are kept efficiently stored. This operation can be performed using the TT-SVD algorithm [13], which is guaranteed to recover a low-rank approximation that satisfies a prescribed accuracy, is quasi-optimal among all other approximations with the same TT-ranks, and has linear complexity in the number of tensor indices. The parameter `ttol` is the relative accuracy that is requested in these truncation stages. The exact amount of performed truncation operations depends on the chosen algorithms.

6. IMPLEMENTED METHODS

As stated in Section 2, at the core of TAPAS there is the solution of linear systems. Among the methods available in the literature, at the moment only those listed in Table 2 have been implemented [6]. They are briefly described, with emphasis on those that treat both matrices and vectors implicitly through Tensor Trains algebra (the explicit methods have been implemented only for verification purposes).

All the methods solve linear systems with the matrix Q or $Q-S$, with the final purpose of evaluating expressions of the form $\pi_0^T(Q-S)^{-j}r$. The algorithms can be grouped in two classes: the one solving linear systems such as $(Q-S)^j x = r$, and the ones computing solution to $x^T(Q-S)^j = \pi_0^T$. We call the latter *transposed* methods. Quite often, resorting to transposed methods can be beneficial. For instance, when dealing with Krylov subspace methods (see `gmres` below) the vectors $\pi_0^T(Q-S)^j$ —that form a basis for the Krylov projections subspace— never represent unreachable states, guaranteeing lower TT-ranks in practice and improving the convergence behavior. When several (unreachable) absorbing states are present, $Q-S$ can be singular[7]; this may pose theoretical and numerical problems for methods that do not belong to the latter class, and therefore we resort to consider $Q-S-\epsilon I$ instead, where ϵ is of the order of the truncation threshold `ttols`. This does not alter the solution more than the truncations performed using the TT-SVD algorithm, but guarantees better convergence properties in practice.

`full-from-tt` assembles explicitly $\tilde{Q}$, $\tilde{\pi}(0)$ and $\tilde{r}$, and solves the linear system through the MATLAB `linsolve` function. `spantree` instead, starting from $\tilde{Q}$, first finds the reachable states and assembles Q, $\pi(0)$ and r (notice that, trough a relabeling, the actual state-space and the potential one are related by $\mathcal{S} \subseteq \mathcal{S}^{(1)} \times \cdots \times \mathcal{S}^{(n)}$), and then solves the system through `linsolve`. These two methods are standard ones, and here are exploited to verify results from the other methods when addressing small models.

`dense-splitting` utilizes a regular splitting [22] of the fully assembled $\tilde{Q}$ decomposed as $Q - S = M - N$, and then solves the system $(Q-S)x = r$ through the standard iterative approach of iterating $x_{k+1} = M^{-1}Nx_k + M^{-1}r$.

This method, not published yet, should be considered as beta version; it is of interest because the convergence is guar-

[6]The code of methods can be found in `tapas/methods/inv_methodname.m`, where `methodname` spans the first column of Table 2.

[7]Most often the correction S is only designed to account for reachable absorbing states, which are easier to describe.

Method	Trans	Pub	Exp	TT	ES
spantree		✓	✓		
full-from-tt		✓	✓		
dense-splitting			✓		
tt-regular-splitting				✓	✓
amen(t)		✓		✓	✓
	✓			✓	✓
gmres(t)				✓	
	✓			✓	
tt-expsumst	✓			✓	✓

Table 2: Methods supported by TAPAS. Legenda: Trans=transposed, Pub=published in the context of Availability/Reliability models, Exp=for constructing explicitly the state-space, TT=for exploiting Tensor Trains, ES=Exponential Sums.

anteed and monotonic, i.e., the partial approximation obtained by x_k is always a guaranteed lower bound for the actual measure MTTF. `tt-regular-splitting` is the implicit version of `dense-splitting`, and is implemented in two variants: one solves the linear system as it is, while `tt-regular-splittingt` solves the transposed system. As experimentally verified, in some situations, solving the system with the transposed can be more effective in keeping the TT-rank bounded. Indeed, the fixed point iteration considers the action of powers of $M^{-1}N$ on the initial vectors, and implicitly represents only reachable states. A deeper analysis of this phenomenon is planned in the future.

`amen` and `dmrg` rely on the AMEn [23] and DMRG [24] solvers in the TT-Toolbox, respectively; they are well-known solvers in the Tensor Train format. The former originates from the physics community and the work on matrix product states representations [24, 25], whereas the latter is a more recent evolution, based on alternative minimization.

AMEn applied to the transposed system is also implemented in `ament`. At the moment, a first guess for the starting point in `amen` is obtained by solving first the linear system $\tilde{D}x = -\tilde{r}$, where $\tilde{D}$ is a diagonal matrix such that $\bigoplus_{i=1}^{n} R^{(i)} \leq \tilde{D}$ component-wise. The preconditioner exploits `ttexpsummldivide`, an approximation of the linear system solution based on the Exponential Sums [14]. This method is the one adoped in [4].

`tt-expsumst` solves the system exploiting the Exponential Sums and an *ad hoc* iteration step deduced from a (non-regular) splitting. This method, called KAES, is an (unpublished) advancement with respect to the one presented in [3]. At the moment is in beta version. The proof that KAES approaches the solution monotonically is in [3].

`gmres` and `gmrest` implement the GMRES iteration exploiting the Tensor-Train arithmetic for matrix-vector operation and for compressing the orthogonal basis. Using the exponential sums approximation of the inverse of $\bigoplus_{i=1}^{n} R^{(i)} \leq \tilde{D}$ as a preconditioner, they work well when the local synchronizations provide a good description of the stochastic process, but require many iterations in other situations.

7. EXAMPLE OF ANALYSIS

Consider an extended version of the example depicted in Figure 1 where there are m copies of A1, and then the system model comprises $n = m+2$ submodels. The parameters setting and other details can be found in `examples/tutorial_-large.m`; the aim of this section is just to show the potentialities of TAPAS. The potential state-space has $4 \cdot 3^{m+1}$ states; so, working with double precision, if $m = 15$ then each vector requires about 1 Gb of memory, if represented explicitly. Instead, with TAPAS it is possible to handle matrices and vectors implicitly; so the memory occupancy depends on the TT-ranks, that for the examples under analysis remains quite small, even if ξ_1 synchronizes all the automata. For small n, the results of the `amen` method were verified with `spantree`, but the analysis for $n \geq 13$ was not feasible on a PC equipped with an Intel i7-1165G7 CPU, 40 Gb of DDR4 RAM at 33MHz when selecting `spantree` or `full-from-tt`.

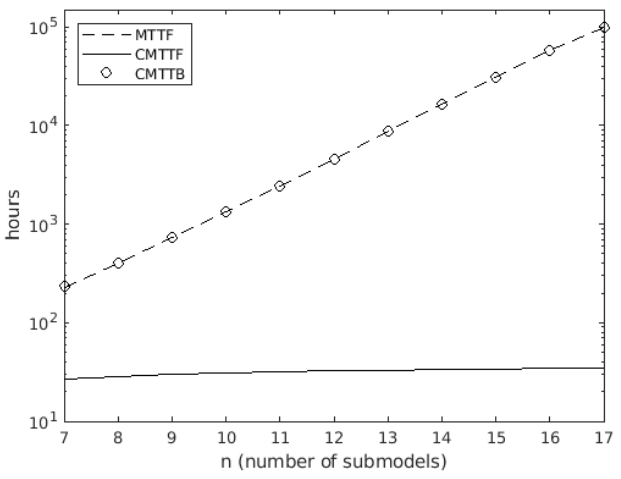

Figure 2: MTTF and $\text{MTTF}_{|\mathcal{B}}$ for $\mathcal{B} = \{\text{F}\}$, called here CMTTF, and $\mathcal{B} = \{\text{B}\}$, called CMTTB.

Figure 2 shows how MTTF and $\text{MTTF}_{|\mathcal{B}}$ vary at increasing of n. Notice that $\pi_{\text{B}}(\infty)$ varies between 0.82 and 0.98, thus from Figure 2 and the known relation

$$MTTF = \pi_{\text{F}}(\infty) \cdot \text{MTTF}_{|\text{F}} + \pi_{\text{B}}(\infty) \cdot \text{MTTF}_{|\text{B}}$$

it is possible to gain useful insights on the system behavior.

8. CONCLUSION AND FUTURE WORK

This paper presented TAPAS, a new tool for efficient evaluation of dependability and performability attributes of systems composed of a large number of interconnected components. Its features and usage have been described using a representative case study.

Useful extensions to the current implementation include to: i) amplify the set of implemented methods, with respect to those in Table 2; ii) exploit the tool to evaluate additional performability measures of interest, with focus on specific application domains; iii) fully adhere to the SAN formalism, resorting to generalized Kronecker algebra theory; iv) develop new features, e.g. to allow steady-state analysis, so to address availability-related measures, and to import the model description, as elaborated by other tools; v) integration of TAPAS in other tools, in addition to MATLAB, possibly open source ones, to promote wider usability.

References

[1] B. Plateau and W. J. Stewart. 2000. Stochastic automata networks. In *Computational Probability*, 113–151.

[2] B. R. Haverkort and K. S. Trivedi. 1993. Specification techniques for Markov reward models. *Discrete Event Dyn. Syst.*, 3, 219–247.

[3] G. Masetti, L. Robol, S. Chiaradonna, and F. Di Giandomenico. 2019. Stochastic evaluation of large interdependent composed models through Kronecker algebra and exponential sums. In *Application and Theory of Petri Nets and Concurrency*, 47–66.

[4] G. Masetti, S. Chiaradonna, L. Robol, and F. Di Giandomenico. 2021. Implicit reward structures for implicit reliability models. *Submitted to Trans. on Reliability.*

[5] A. Bondavalli, M. Nelli, L. Simoncini, and G. Mongardi. 2001. Hierarchical modelling of complex control systems: dependability analysis of a railway interlocking. *Journal of Computer Systems Science and Engineering*, 16, 4, 249–261.

[6] G. Ciardo and K. S. Trivedi. 1991. A decomposition approach for stochastic Petri net models. In *The* 4^{th} *Int. Workshop on Petri Nets and Perform. Models (PNPM 1991).* Melbourne, Victoria, Australia, 74–83.

[7] P. Lollini, A. Bondavalli, and F. Di Giandomenico. 2009. A decomposition-based modeling framework for complex systems. *IEEE Trans. Reliab.*, 58, 1, 20–33.

[8] S. Derisavi, P. Kemper, and W. H. Sanders. 2004. Symbolic state-space exploration and numerical analysis of state-sharing composed models. *Linear Algebra Applications, Special Issue on the Conference on the Numerical Solution of Markov Chains*, 386, 137–166.

[9] H. Sukhwani, A. Bobbio, and K. S. Trivedi. 2015. Largeness avoidance in availability modeling using hierarchical and fixed-point iterative techniques. *International Journal of Performability Engineering*, 11, 4, 305–319.

[10] B. Plateau, J.-M. Fourneau, and K.-H. Lee. 1989. PEPS: A package for solving complex Markov models of parallel systems. In *Modeling Techniques and Tools for Computer Performance Evaluation*, 291–305.

[11] S. Donatelli. 1993. Superposed stochastic automata: A class of stochastic Petri nets with parallel solution and distributed state space. *Performance Evaluation*, 18, 1, 21–36.

[12] P. Buchholz and P. Kemper. 2004. Kronecker based matrix representations for large Markov models. In *Validation of Stochastic Systems.* Volume 2925, 256–295.

[13] I. V. Oseledets. 2011. Tensor-train decomposition. *SIAM J. Sci. Comput.*, 33, 5, 2295–2317.

[14] D. Braess and W. Hackbusch. 2005. Approximation of $1/x$ by exponential sums in $[1, \infty)$. *IMA J. Numer. Anal.*, 25, 4, 685–697.

[15] The Mathworks, Inc. 2021. *MATLAB version 9.11.0. 1769968 (R2021b).* The Mathworks, Inc. Natick, Massachusetts.

[16] J. Babar, M. Beccuti, S. Donatelli, and A. Miner. 2010. GreatSPN enhanced with decision diagram data structures. *Applications and Theory of Petri Nets*, 308–317.

[17] T. Courtney, D. Daly, S. Derisavi, S. Gaonkar, M. Griffith, V. Lam, and W. H. Sanders. 2004. The Möbius modeling environment: recent developments. In *First Int. Conf. on Quant. Eval. of Syst. (QEST 2004).* Enschede, Netherlands, 328–329.

[18] P. Buchholz. 1999. Hierarchical structuring of superposed GSPNs. *IEEE Trans on Software Engineering*, 25, 2, 166–181.

[19] K. S. Trivedi. 2002. SHARPE 2002: Symbolic hierarchical automated reliability and performance evaluator. In 32^{nd} *Annu. IEEE/IFIP Int. Conf. on Dependable Syst. and Netw. (DSN 2002).* Washington, DC, USA, 544.

[20] W. H. Sanders and J. F. Meyer. 1991. A unified approach for specifying measures of performance, dependability and performability. In *Dependable Computing for Critical Applications, Vol. 4 of Dependable Computing and Fault-Tolerant Systems*, 215–237.

[21] H. Choi and K. S. Trivedi. 1993. Conditional MTTF and its computation in Markov reliability models. In *Annual Reliability and Maintainability Symposium 1993 Proceedings*, 56–63.

[22] A. Berman and R. J. Plemmons. 1994. *Nonnegative matrices in the mathematical sciences. Classics in Applied Mathematics.* Volume 9, 340.

[23] S. V. Dolgov and D. V. Savostyanov. 2014. Alternating minimal energy methods for linear systems in higher dimensions. *SIAM J. Sci. Comput.*, 36, 5, A2248–A2271.

[24] S. R. White. 1992. Density matrix formulation for quantum renormalization groups. *Phys. Rev. Lett.*, 69, 19, 2863 –2866.

[25] S. R. White. 1993. Density-matrix algorithms for quantum renormalization groups. *Physical review b*, 48, 14, 10345 –10356.