Abstract

PEPS is a tool package for modeling and solving models expressed in Stochastic Automata Networks (SAN). The SAN formalism defines a compact storage scheme for the transition matrix of the Markov chain and it uses tensor algebra to handle the basic vector matrix multiplications. This paper presents a short timeline of PEPS previous versions and the new features included in version 2007.

1. PEPS Timeline

PEPS project started in the late 80’s aiming a software tool to model and to compute numerical solutions for the Stochastic Automata Networks (SAN) formalism. The SAN formalism was proposed by Plateau [1] and its basic idea is to represent a whole system by a collection of subsystems with an independent behavior (local events) and occasional interdependencies (functional rates and synchronizing events). The framework proposed by Plateau defines a modular way to describe continuous and discrete-time Markovian models, but only continuous-time SAN are implemented in PEPS.

The first PEPS version was presented in [2] and implemented a simple vector-matrix multiplication, where the matrix columns were generated, column by column, in each iteration. The full matrix was never generated, only the tensor format descriptor was stored.

1.1. PEPS2000

In 2000 a new version was release, this version implemented a set of new algorithms proposed by Fernandes [3]. The main increasing technique implemented in this version was a vector-descriptor multiplication method, the Shuffle algorithm. With this method, each tensor product term multiplies a part of the vector, never generating any part of the full matrix. After all tensor multiplications, we have a complet vector-matrix multiplication. Still in PEPS2000 version a simple interface compiler, optimized manipulation of functional dependencies [3], projection methods to stationary solution and algebraic aggregation of automata were implemented.

1.2. PEPS2003

The PEPS2003 release brings a new interface compiler, sparse vector manipulation and just-in-time functions evaluation. The new interface compiler was more intuitive and compact. It implemented some basic replications features to describe identical automata and queueing structures. The implementation of a sparse vector format in PEPS allowed the computation exclusively within the reachable space state. This kind of format reduced considerably the storage space for rather sparse models. Another feature implemented in PEPS2003 is the just-in-time function evaluation. This method generates, for each function described in the SAN model, a C++ code. The C++ function codes are compiled and linked with the PEPS solution methods and they are called every time that a function is evaluated.

2. PEPS2007 Version

For this version a new software structure was adopted, splitting PEPS2007 in a set of independent modules. Each module implements a part of the compiling/solution procedures. This approach improves PEPS maintenance and development. With this approach, new methods can be developed and tested independently of the PEPS stable version. Specifically, it allows different versions development locally and/or logically distributed without a particular care of recent bugs corrections and improvements in other parts of the PEPS software.

The basic PEPS2007 modules are assembled in three groups: Interface, Data Structure, and Solution Methods.

Together with the previous modules implementing old version features, the following new features of PEPS2007 version are implemented as well.

2.1. New Compiler for SAN models

A new version of compiler was developed to facilitate the description of models with replicated automata, or states inside an automaton. This compiler is based on previous version interface and it kept the PEPS2003 full compatibility. In previous PEPS versions, only the
replication of identical structures, were possible. Now, we can replicate automata and states with different transitions, events or rates. The only restriction concerns the cardinality of each replicated structure, i.e., replicated automata must have the same number of states.

2.2. Semantic Aggregation

Semantic aggregation method implemented in PEPS is based on the method described by Benoit et al. [4]. According to this method, a group of fully identical automata, i.e., automata with the same number of states, transitions and rates, may be aggregate into a single automata where each state represents the number of grouped automata in each state. This method seems to have a narrow scope of application, since it applies only when the model integration functions consider the identical automata also indistinguishable. However, it represents quite impressive state space reductions.

2.3. Bounding Methods

Tensor algebra in SAN can be used to build large Markov Chains and store them in compact format. However, for really huge models, the model solution is still hard and, sometimes, computing exact performance indexes is feasible. Bounding algorithms [5] can be used in Markov Chains to obtain upper (or lower) bounds on performance. Bounding algorithms reduce the Markov chain transition matrix into a smaller monotone matrix. PEPS2007 implements an algorithm to generate a monotone upper bounding matrix directly from the SAN Markovian Descriptor.

2.4. Transient Analysis

PEPS2007 version brings a transient solution method based in the standard uniformization method [6]. Additionally, an extended module was developed to compute the Point Availability to large systems. Point Availability represents the probability that a system are in a defined set of $UP$ global states, where the system delivers a service. Two different point availability methods were implemented in PEPS. Both of them implement a stationary detection method, where computations are stopped if the system arrives in the stationarity before the maximum iteration number. The first method was described in [7] and it detects the stationarity over the probability vector, if the probability vector does not change between two iterations. The second method was proposed by Sericola in [8]. This method uses a control of the sequence of the vector $V_n$. The vector is obtained by $V_n = P^n1_U$, where $P$ is transition matrix and $1_U$ is a vector with the $UP$ states marked as 1 and $DOWN$ states marked as 0. The stationary detection method used in this algorithm searches the largest and the smallest values of $V_n$. The stationarity is reached when the difference between them is smaller than an error $\epsilon$.

2.5. PH-SAN

Ph-SAN is an extension of SAN compiler that includes pre-compiler directives to handle phase-type rates. The inclusion of a phase-type rate in a SAN model is quite complex because it has different behaviors according to where it appears. The inclusion techniques implemented in PEPS2007 were developed in [9] and it consists in translating the ph-events into series of common (exponential) events, i.e., this transformation technique converts a PH-SAN model in a equivalent SAN model with only exponential rates.

3. Work in Progress

PEPS software tool is a live project and currently there is on going work to add new vector-descriptor multiplication techniques to speedup both transient and stationary solutions, as well as a reachable state space manipulation using more sophisticated structures, namely MDD [10]. These new features may result in new versions of the PEPS tool and the interested researcher and practitioner may found a more thorough information about PEPS in the webpage http://www-id.imag.fr/Logiciels/peps/.

References


