Assessment of reliability indicators from automatically generated partial Markov chains

Pierre-Antoine Brameret

To cite this version:

Pierre-Antoine Brameret. Assessment of reliability indicators from automatically generated partial Markov chains. Other. École normale supérieure de Cachan - ENS Cachan, 2015. English. <NNT : 2015DENS0032>. <tel-01230869>

HAL Id: tel-01230869
https://tel.archives-ouvertes.fr/tel-01230869

Submitted on 19 Nov 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

ENSC-2015-N°591

PhD Thesis
of the ECOLE NORMALE SUPERIEURE DE CACHAN (FRANCE)

Presented by
Mr. Pierre-Antoine BRAMERET

to obtain the degree of

DOCTEUR DE L'ECOLE NORMALE SUPERIEURE DE CACHAN
Domaine : **Electronique, Electrotechnique, Automatique**

Assessment of Reliability Indicators from Automatically Generated Partial Markov Chains

Calcul des Indicateurs de Sûreté par la Génération Automatique
de Chaînes de Markov Partielles

PhD Thesis defended on 09.07.2015 in Cachan in front of the following committee:

Andrea Bobbio	Professor, Università del Piemonte Orientale, Italy	President
Christophe Berenguer	Professor, Grenoble-INP, France	Reviewer
Pierre-Etienne Labeau	Professor, Université Libre de Bruxelles, Belgique	Reviewer
Ana Bušić	Research Assistant, INRIA Ecole Normale Supérieure, France	Examiner
Antoine Rauzy	Professor, Ecole Centrale de Paris, France	Advisor
Jean-Marc Roussel	Associate Prof., Ecole Normale Supérieure de Cachan, France	Advisor

Laboratoire de Recherche en Production Automatisée (LURPA)
ENS Cachan
61, avenue du Président Wilson,
94235 Cachan CEDEX, France

Acknowledgments

A Ph.D. thesis is a personal work, in which advisors have a whole role to play. I thank Professor Antoine Rauzy and Associate Professor HDR Jean-Marc Roussel for all their work with me, their trust in my choices, their guidance, and their advices. I also want to thank Professor Jean-Jacques Lesage, who accepted to supervise me at the beginning of my Ph.D.

I would like to thank my reviewers Professor Christophe Berenguer and Professor Pierre-Etienne Labeau, as they accepted to review my thesis, in such short notice. Also, I would like to thank them for the useful content of their reviews. Thank you both Researcher Ana Bušić and Professor Andrea Bobbio for being part of my doctoral committee, as well as for your precious feedback on the thesis.

Here are special thanks to very helpful friends and colleagues, who helped me during the writing of the thesis. Jeremie Saives read my whole thesis and helped me fix erroneous spellings, grammar issues, and unclear developments. Cyril Lacroix, with who I shared the pressure and the difficulty of the writing. Their help was very important to my thesis.

Now I am asking for cheerfulness from all my colleagues, as it builds a solid and efficient background for work. I would like to thank the whole Lurpa and the AltaRica team, because we gave it a successful try together. Some special thanks for all the extra activities than we shared together, in the Civil, or in NoL@g. Thank you all, previous and current colleagues, and keep it that way.

Finally, I want to express the deepest thanks to Ophélie and to my parents and family, for their support and comfort, day after day, in this long adventure.

Cachan, July 2015

Pierre-Antoine

Contents

List of Figures

List of Tables

List of Algorithms

Introduction

Trustworthiness in systems is of paramount importance. Many safety analysis methods have been developed to analyze and calculate the risks associated to a system. Evaluation of the risks has two dimensions, which are the severity and the frequency. The severity is analyzed with qualitative methods, such as the Failure Mode and Effects Analysis, whereas quantitative methods are used to analyze the frequency.

The most efficient quantitative method, the Fault Tree Analysis, is usually too low-level: the gap between the specifications of the system under study and the fault tree of the system distorts the results. Therefore, Model-Based Safety Assessment (MBSA) are developed to offer higher-level languages. Such techniques enable more systematical approaches, re-usability of the models, and hierarchical structures of the models. Different formalisms embody different tradeoffs between the computability of the models and the gap between the model and the system under study. The most accurate models require stochastic simulations, and are, as a consequence, costly to calculate.

Markov chains offer a reasonable tradeoff between computability and modeling concepts. With Markov chains, it is possible to model reconfigurations, repairs, failures on demand, common cause failures, etc., but it is difficult to model the aging of components, and probability distributions are usually exponential to make the calculations efficient (homogeneous Markov chains). Markov chains are usually used to model the different states, or configurations, of a system, and transitions between these states model events, such as failures or repairs. However, they suffer from a major drawback: the state space explosion that is the consequence of the numerous combinations of the states of each component.

The objective of this thesis is to propose a method to tackle the state space explosion of the Markov chains while calculating accurately the safety indicators of the system under study. This idea is based on the observation that a repairable system is, most of the time, in its nominal state. From the nominal state, only states with few failed components are reachable before the system is repaired. Hence, only a small proportion of the states of the Markov chain will gather most of the probability, and these states are close to the nominal state.

To tackle the state space explosion, a Markov chain of the system under study is partially generated from a higher-level model. A variation of Dijkstra's algorithm is used to select the most probable states of the chain *during* its generation. The selection is based on a so-called *relevance factor*, which gives a heuristic estimation of the weight of the states in the Markov chain. Bounds on the safety indicators can be obtained with the partial chain.

The partial generation of Markov chains gives very good results. The reliability indicators can be obtained with satisfactory accuracy with only a tiny fraction of the complete chain. The generation of the Markov chain is very efficient and almost linear in time regarding the number of transitions. In our case, the chosen modeling language from which Markov chains are generated is AltaRica 3.0. The proposed method can also be used with other high-level modeling languages, such as Boolean logic Driven Markov Process (BDMP) or Petri nets.

The manuscript is organized as follows. Chapter 1 introduces an overview of the AltaRica 3.0 project in the scope of the safety assessment, followed by a review of the available methods to compute safety indicators with very large Markov chains. The main contributions of this thesis are presented at the end of the chapter.

In Chapter 2, the partial generation method is presented. First, the soundness of the partial generation method is developed through the analysis of Markov chains, their transient solutions, and the interpretations of Markov chains for large systems. Then, the core of the partial generation method is presented. The partial generation relies on Dijkstra's algorithm combined with a relevance factor to estimate which states of the Markov chain will be relevant to the assessment of the safety indicators of the system under study. Finally, an alternative algorithm is developed, which calculates the bounds on the safety indicators of the system.

Chapter 3 focuses on the strengths and weaknesses of the partial generation. To do so, case studies of different sizes and taken from the literature are used to analyze the method on practical experiments.

In Chapter 4, the implementation of the partial generation method is discussed. In particular, the computational complexity of the method can only be calculated according to the chosen implementation.

1. Context and Related Work

In this chapter, the context of the thesis is presented. First, an overview of safety engineering and of probabilistic safety assessments is given. This part focuses on some of the available modeling languages to assess safety models, namely Markov chains. Then, the related work analysis focuses on the approximated methods to calculate transient solutions of large Markov chains. Finally, the main contributions of this thesis are presented.

1.1. Context

Confidence in systems is of paramount importance. Hazards in the environment of the system may lead to catastrophic consequences for the system or its users. The risks of such consequences must be analyzed.

This activity is today an engineering discipline named *Safety engineering.* The role of safety engineers is to assure that a safety-critical system behaves as specified, even when some of its components fail. As soon as possible, potential risks must be analyzed, to predict both their frequency, and severity. In other words, safety engineers answer the three questions:

- What can go wrong?
- How frequently is it? What is the likelihood of it going wrong?
- What are the consequences?

Failure Mode and Effects Analysis (FMEA) [Stamatis 2003] is a methodology to analyze the risks. To do so, components, assemblies, and subsystems are reviewed to identify failure modes, their causes (hazards) and effects (risks). FMEA are mostly used to identify all the initiating faults of a system.

HAZard and OPerability study (HAZOP) is another analysis of the risks initiated by hazards. HAZOP focuses on confidence intervals in the system, and analyze the subsequent risks. Actions may be taken to reduce the severity of the risks.

Both these qualitative methodologies aim to reduce the severity of the risks. However, the frequency of the risks must be analyzed. For instance, a terrible risk, such as the fusion of the nuclear core of a nuclear power plant, is very severe but its very low frequency, such as "once every ten thousand years", makes it acceptable.

Probabilistic Safety Analysis (PSA) determines the frequency, or probability, of the risks. This probability is very difficult to estimate due to numerous factors and their complex interconnections. Safety engineers have developed specific quantitative analysis methods to solve these problems. Both qualitative and quantitative approaches share the goal of analyzing the risks.

Fault Tree Analysis (FTA) [Dugan et al. 1992] [Vesely et al. 2002] is a top-down, deductive analytical method. In FTA, initiating primary events, such as component failures, human errors, and external events, are traced through Boolean logic gates to an undesired risk (top event). FTA may be qualitative or quantitative. When failure and event probabilities are unknown, qualitative FTA may be used to compute minimal cut sets, which are the minimal sequences of events leading to the unwanted event. Quantitative FTA is used to compute top event probability, and is based on Boolean interpretations of the fault tree (see [Valiant 1979] for instance).

FTA is very efficient (see [Rauzy 1993]) and different industrial tools were developed for the safety analysis of industrial case studies. Qualitative and quantitative analyses are usually used as complementary approaches (e.g. in the standard for civil airborne systems ARP 4761 [SAE 1996]).

However, due to the mathematical theory used for FTA models, the dynamic behaviors in these models are not representative of the dynamic behaviors of real systems. This limitation is well-known to safety engineers. The choice of the modeling language is a tradeoff between the complexity of the calculations and the concepts that can be modeled. Figure 1.1 shows some modeling languages and the tradeoff they embody. Currently, the ideal tool does not exists, i.e. a tool which would compute fine-grained model with no more computations than those a fault tree of the same size would require.

Several approaches have been developed to provide different tradeoffs. Some authors proposed an extension of Fault Trees to take into account temporal aspects [Walker and Papadopoulos 2009] [Merle et al. 2011]. Other authors used higher-level modeling languages in which sequential aspects are included. Such languages are:

- Stochastic Automata Networks (SAN) [Fernandes et al. 1998]
- Boolean logic Driven Markov Processes (BDMP) [Bouissou 2002; Bouissou and Bon 2003]
- AltaRica 3.0 [Rauzy 2008] [Prosvirnova 2014]

These models are directly or indirectly based on state machines. Qualitative analysis is based on the search of specific paths in this state machine. Some quantitative methods are also based on the evaluation of these paths. Other methods are based on Markov chains generated from these state machines. The major drawback of methods based on Markov chains is the state space explosion. For instance, the oil production system detailed in Section 3.2.3 has only 15 basic components, and its state machine has nearly $47,000$ states and $5,210,000$ transitions.

Because of the size of the Markov chains, safety indicators cannot always be calculated with

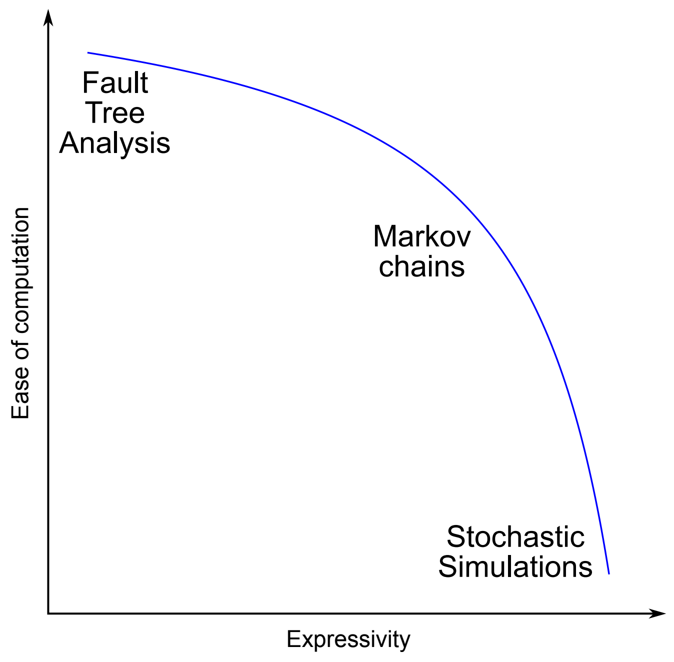


Figure 1.1.: Pareto front representing the tradeoff between the ease of modeling and the efficiency of the calculations.

this modeling language. In this thesis, we studied the possibility to obtain the probabilistic indicators of a safety system, such as its availability, from high-level modeling languages, without completely generating the Markov chain describing its dynamic behavior.

Usually, system designers and stakeholders specify bounds of the error made on the probabilistic evaluations of the safety indicators. For instance, for the unavailability, it is rarely below 10% (as seen in [Bon and Collet 1994; Collet and Renault 1997]), given that it is difficult to obtain accurate reliability data of basic components. It is not always possible to achieve less than 10% error, as seen in the PSA of the Paluel nuclear power plant, where the probability of the fusion of the core is evaluated with an error factor about 3.5 (see [Dubreuil Chambardel et al. 1991] [Brisbois et al. 1990]).

The error made by the partial generation is analyzed in Figure 2.9, in Section 2.3. The idea is that the error made with the partial generation of Markov chains stays below the error of the input data.

In this thesis, we focused on AltaRica models to benefit from tools developed in our team. However, it is important to note that the proposed method can be used with other high-level modeling languages such as BDMP, or Petri nets.

The AltaRica 3.0 project

AltaRica 3.0 is a high-level formalism dedicated to safety analyses. The idea is to write safety models in high-level description language to keep them close to the functional and physical architecture of the system at hand. Such a high-level description formalism reduces the gap between the specifications of the system under study and the associated safety model.

The semantics of AltaRica 3.0 is defined in terms of Guarded Transition Systems, as detailed in [Rauzy 2008]. Prior to most assessments, AltaRica 3.0 models are flattened into Guarded Transition Systems (GTS). GTS acts as a pivot language, for which assessment tools are developed. As illustrated Figure 1.2, which gives an overview of the AltaRica 3.0 project (see [Prosvirnova et al. 2013]), GTS can be calculated with different low-level languages.

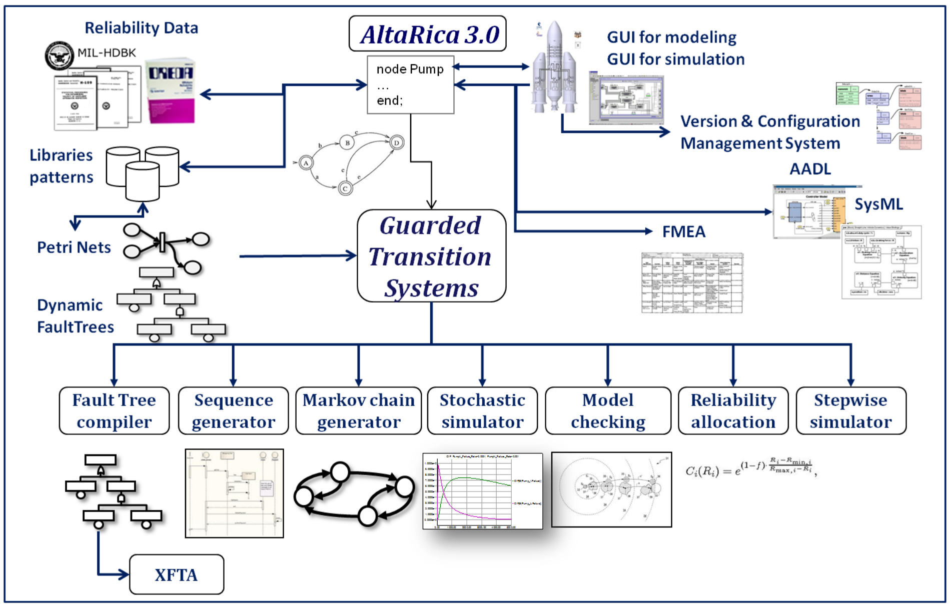


Figure 1.2.: The AltaRica 3.0 project.

The AltaRica project provides different tools to the safety engineers which embody different tradeoffs between the size of the models and the expressiveness of the models. The main low-level languages used for quantitative analyses are:

- Static fault trees (see [Rauzy 2002]) are very efficient, but are limited to static approximations of the systems.
- Stochastic simulations are very accurate and can, by definition, simulate any behavior. However, safety properties are difficult to assess, because of the high number of sequences that must be calculated.
- Markov chains are limited to models where failures and repairs are exponentially distributed, but it can model higher-level concepts than fault trees (such as reconfigurations). The main drawback of Markov chains is the number of states of the chain, which is prone to the combinatorial explosion. Markov chains embody a good tradeoff between computation costs and expressiveness.

All the results presented in this thesis were obtained with the help of tools that were developed within the AltaRica 3.0 project. The “Markov chain generator” is my personal contribution to the project. It is mostly used with two other tools:

- The tool developed by Tatiana Prosvirnova to compile AltaRica models to Guarded Transition Systems [Prosvirnova 2014],
- A tool developed by Antoine Rauzy to calculate transient solutions of Markov chains.

1.2. Related Work

The objective of this section is to provide insight related to the calculation of safety indicators of systems modeled with large Markov chains. The main problem of such Markov chains is the state space explosion, which makes the calculation impossible for complex systems. Establishing a full review of the literature on this topic is quite difficult, because Markov chains are widespread in many areas of science. We review here a few ideas that are related to the calculation of large Markov chains.

Markov chains is a very good modeling language, which is based on the modeling of the states of a system. Further definitions and calculation methods are presented in Section 2.1. Markov chains are constrained by the Markov hypothesis, which states that the future of the system only depends on the current situation of the system. However, even with the Markov hypothesis, the very nature of the exponential distributions associated to transitions between states (in the case of homogeneous Markov chains) makes their strength, as it makes their mathematical analysis possible and efficient. Consequently, they have been thoroughly used and studied since their creation in [Markov 1906], in various scientific fields.

Our review is based on three families of approaches. The first approaches, shown in Section 1.2.1, deal with various calculation techniques, using approximate calculation or reduction of the state space. These methods generally uses the full Markov chain, which is then reduced.

The second family of approaches overcome the inherent state space explosion by avoiding the construction of the chain. These approaches, shown in Section 1.2.2, use the calculation of sequences of events to evaluate the safety indicators of a system. Path-based approaches are close to the minimal cutsets approach for FTA. Paths can be generated from a higher-level language, which may avoid the generation of the complete Markov chain. However, the number of paths is prone to the combinatorial explosion.

The third family of approaches overcome the inherent state space explosion by aggregating numerous states into very little states. These aggregation techniques are presented in Section 1.2.3. The full chain may not be required to build aggregates, as some of these methods can be applied during the generation of the chain from a higher level language. This makes these approaches particularly interesting.

1.2.1. Calculation and Reduction Techniques of Large Markov Chains

Approximate calculation techniques and reduction techniques are tightly linked to the numerical calculation techniques introduced in Section 2.1.4. This is due to the solution of the

Markov chain (see Section 2.1.2, Equation (2.2)), which is a series involving the powers of the transition matrix. Hence, approximate calculation techniques that *also* take into account the large size of the Markov chains are developed. Some authors developed exact methods which produce a smaller Markov chain that is *equivalent* to the complete chain, and others, approximate methods.

Lal et al. considered the inversion of transition matrices and optimized calculations of steady state probabilities by partitioning the transition matrix (see [Lal and Bhat 1988]). It is not strictly speaking a reduction of the model to a smaller model, but it is yet a reduction of the complexity toward the calculation of the Markov chain.

Pribadi et al. introduced a method in [Pribadi et al. 2001] to reduce Markov chains in size while guaranteeing the exact probabilistic assessment of the performance indicators. It is efficient to reduce the chain toward its calculation but this relies on ergodicity of the chain to be reduced, which is not always verified, in particular for non-repairable systems.

The following methods are based on the concept of *reward*. A reward is an arbitrary quantity that is associated to the system in each state of the Markov chain. The expected value of the reward is computed knowing the expected probabilities of the states of the Markov chain. The reward can be the production rate of the system, its availability, and so on. Zhao et al. proposed a theoretical framework to study state-space truncation, by censoring the Markov chain to keep only the states which give non-null reward. As shown in [Zhao and Liu 1996], the equivalence between the censored Markov chain and the initial Markov chain is computationally expensive to calculate. Fourneau et al. proposed a method to approximate the censored Markov chain, which gives bounds on the error made by applying such censoring (see e.g. [Fourneau et al. 2007]).

Other approaches with the censored Markov chains aimed at approximating the censored chain without knowledge of the full Markov chain. Dayar et al. introduced the DPY algorithm to calculate such approximation (see [Dayar et al. 2006]). Bušić et al. further studied the approximations of censored Markov chains, and improved the bounds proposed by Dayar et al. using more knowledge on the full Markov chain (see [Bušić et al. 2012]).

Another way to evaluate large Markov chains is to calculate them approximately. Mercier developed in [Mercier 2007, 2008] approximate bounds to quickly calculate transient and steady-state probabilities. Unfortunately, the method uses the inversion of a matrix which is similar to the transition matrix, so the method is not scalable.

Carrasco proposed in [Carrasco 2003] approximate bounds with error control to calculate transient rewards of a Markov chain. This method is interesting because it uses a smaller Markov chain to approximate the bigger one, and is based on sequences to determine the size of the small chain. It is an optimized algorithm to calculate rewarded Markov chains. However, the algorithm which generates the smaller chain uses matrix product involving the generator matrix of the full chain, so the method does not avoid the combinatorial explosion of full the Markov chain.

Another approach has been proposed by Plateau et al. for the assessment of Stochastic Automata Network (SAN). SAN are a high level formalism to describe finite state automata (see e.g. [Fourneau and Plateau 1991] [Fernandes et al. 1998]). The idea is to express the Markov chain in terms of a generalized tensor product, using the modularity of SAN models. This approach makes it possible to reduce the size of the chain, but it does not reduce its number of states.

For all these efficient methods (with the exception of the DPY algorithm), the input data is only a Markov chain without complementary information about the represented system. In this thesis, the method uses a higher-level formalism to *partially* construct the state space of the failure/repair behavior of the system. We directly obtain a partial Markov chain, which does not need to be reduced.

1.2.2. Path-based Approaches

Path-based approaches calculate the availability of a system through the calculation of the probabilities of paths in the Markov chain. These methods avoid the evaluation of the whole Markov chain and propose an evaluation of the error introduced by such approximation.

Path-based approaches have been developed by Electricité de France (EDF) for the risk assessment of nuclear power plants. The idea is close to what is done in Fault Tree Analyses. Sequences of failures and repairs are considered from the initial state to the failure states of the system. A probability is calculated for these sequences, and their contribution is added to the unreliability of the system. When the probability of a sequence is below a given threshold, the sequence – and the possible sub-sequences – is discarded, and the error is augmented with this probability. At the end of the process, the error is used to assert the confidence in the results.

To obtain representative results, it is necessary to address the following difficulties: the computation of the probability of a sequence with respect to the events in the sequence, and the loops introduced by repairs. Three methods were developed to tackle these difficulties in different situations:

- Bon and Collet proposed in [Bon and Collet 1994] a first method to compute the reliability of completely repairable systems.
- Collet and Renault proposed in [Collet and Renault 1997] a specific method to compute the reliability of unrepairable systems.
- Bouissou and Lefebvre described in [Bouissou and Lefebvre 2002] an extension of [Bon and Collet 1994] to compute the asymptotic availability of completely repairable systems.

Reliability of completely repairable systems [Bon and Collet 1994]

The method called SRI is developed to compute a lower bound of the reliability of a system through the exploration of sequences of events. The idea is to calculate an exponential distribution which would approximate the whole system. The reliability $R(t)$ is approximated as follows:

$$R(t) \geq e^{-\Lambda_{eq} t}$$

The algorithm focuses on the probability α of arriving in a failure state before returning in the perfect state.

$$\Lambda_{eq} = \lambda \times \alpha$$

where λ is the exit rate from the initial state. To do so, the algorithm calculates the sequences which starts from the initial state E_0 and goes to failure states E_1 without transiting through another element of E_1. This gives the probability $P(E_0, E_1)$ to go from E_0 to the set E_1. Finally, α is a quantity that satisfies $\alpha \geq P(E_0, E_1)$.

To compute those quantities, an algorithm is proposed to explore the sequences, in depth-first order. The state space of the system is only partially explored, and the Markov chain is only used as a support of the exploration. Hence, loops in the sequences are handled.

The algorithm needs the complete repairability of the system to work correctly, in order to ensure the regeneration of the Markov process. This method gives closer approximations when the system is highly available.

With this algorithm, it is also possible to compute other quantities of interest for completely repairable systems, as the set E_1 can be changed to a set of objectives. However, there are strong requirements on the indicators that can be calculated.

Reliability of unrepairable systems [Collet and Renault 1997]

This method is based on Laplace-transform inverses. They are used to calculate the probability of sequences of failures. This requires the traversal of all the paths from the initial state to failed states. The unreliability of the system is obtained by summing the probabilities of the paths.

To avoid the complete exploration, sequences may be truncated if their probability is below a threshold (which avoids the calculation of the sub-sequences). However, the calculation of the probability of a sequence is computationally expensive (non linear with respect to the number of events along the path). Moreover, if the truncation is needed to reduce the number of sequences, this calculation must be done after each newly discovered event. The unreliability of repairable systems cannot be obtained without truncation of the sequences, because of the inherent loops induced by the failures and repairs.

This method does not require the full Markov chain to be generated, but must keep track of the already explored sequences. As a consequence, it can be used directly with higher-level language and tools. Even if this method does not require Markov chains, it is based on exponential distributions of the events, which is typical of Markov chains.

Asymptotic availability of quickly repairable systems [Bouissou and Lefebvre 2002]

In [Bouissou and Lefebvre 2002], an extension of [Bon and Collet 1994] is developed. The development of this method is tightly related to the publication of the Boolean logic Driven Markov Processes (BDMP) [Bouissou 2002; Bouissou and Bon 2003]. The method is implemented in the software FIGSEQ (for FIGaro SEQuences), which enables efficient calculation of the BDMPs.

The idea of the method is that, for highly repairable systems, after a failure of the system, all the repairs on components are made before another failure of the system. Hence, the asymptotic availability of a system $\bar{A}(\infty)$ is expressed as a ratio of the expected time to recover from failure E_R and the total length of a cycle of failure-repair E_C:

$$\bar{A}(\infty) \approx \frac{E_R}{E_C}, \qquad E_C = MTTF + E_R$$

where $MTTF$ is the Mean Time To Failure of the system.

The $MTTF$ of the system is approximated with the SRI method, and $MTTF \geq \frac{1}{\Lambda_{eq}}$.

The calculation of E_R is done in two stages. Each stage uses the principle of the SRI method. First, in the failure analysis stage, the sequences S leading to failure states are calculated. Besides, the $MTTF$ is also calculated. Second, in the repair analysis stage, the failure sequences are extended. For each sequence $s \in S$, SRI is used again to explore the sequences of events $S(s)$ that recover the system from failure.

As in SRI, sequences can be truncated, in both stages of the method, which leads to intricate calculations of bounds on E_R. These bounds are used to calculate the approximate value of $\bar{A}(\infty)$. To calculate the higher bound of E_R, the system engineer must provide an upper bound of the mean time to repair the system. The obtained approximation of $\bar{A}(\infty)$ cannot be bounded.

The Markov chain is only generated to trace the state space visited by the sequences. This technique avoids the interpretation of the complete Markov chain. The sequences can be calculated from a higher-level language.

The truncation provides satisfactory results on highly available systems. Inaccurate results are obtained when the system is not available enough. This is due to the high number of looped sequences compared to the number of direct sequences. These looped sequences are not negligible for system with low availability, and they must be truncated because there are too many of them.

Conclusions on path-based techniques

Path-based methods proved to be very efficient in the calculation of the safety indicators for unrepairable or highly repairable systems. They are, however, limited by the huge number of sequences that must be considered, because of the very nature of failure-and-repair cycles.

In this thesis, as loops are inherently part of repairable systems, our first idea was to select some states of a Markov chain that would be able to represent the most probable evolutions of the system. Instead of calculating the sequences that could lead to undesired states, the partial Markov chain is used to calculate the probability of each of these undesired states. Looped sequences are naturally handled by the calculation of the Markov chain. The truncation of the Markov chain would be done in an analogous way: an indicator is associated to each state, representing its relevance in the final Markov chain, and the exploration is stopped if this indicator is below a given threshold.

1.2.3. Aggregation Techniques

Aggregation techniques focus on the reduction of the size of the Markov chain by aggregating multiple states into fewer states. States are grouped by their level of interaction. When some states are strongly connected together, but are less connected to the others, they are aggregated into a single group, without changing the overall dynamic of the Markov chain. However, transitions to groups and from groups must be calculated, in order to respect the overall dynamic of the chain.

Regarding aggregation techniques, we chose to focus on the work of two teams: on the one hand, Courtois and Semal, on the other hand, Muntz, de Souza e Silva and Goyal.

Courtois and Semal developed important theorems to calculate the equivalent transitions in aggregated Markov chains (see e.g. [Courtois 1977; Courtois and Semal 1984]). However, these results imply once again the inversion of a matrix which size is prohibitive.

Muntz, de Souza e Silva and Goyal proposed in [Muntz et al. 1989] a method to compute the steady-state unavailability of a repairable system through an approximate aggregation of the Markov chain. Bounds on the error are also provided. The main benefit of the method is that it does not require the complete Markov chain to produce the approximate chain. The method is based on the fact that the probability of a safe repairable system is concentrated in the states where there are at most a few failed components, and that there are numerous improbable states where a high number of components are failed. Consequently, the states where the probability is concentrated are kept, and the improbable states are aggregated into fewer states.

The method works as follows. States where few components are failed are kept detailed, to keep a good accuracy for the most probable states. The other states are aggregated, by grouping states that have the same number of failed components. For instance, only states where more than five components are failed are aggregated.

However, the method relies on the ergodicity of the chain, and makes hypotheses on the model: the number of failed components must be known in each state of the system, and repairs of each component must be done one after another. Further considerations on the complexity of the method are presented in Section 2.4.3, when concepts about computational complexity of calculation of Markov chains have been presented.

This method, with more knowledge on the initial model (which must be a safety model with the known number of failures in each state), gives a good tradeoff between the computational cost and the obtained bounds.

Courtois and Semal developed in [Courtois and Semal 1995] another method which computes steady state probabilities of repairable system by aggregating states according to the number of failed components of the system. In this method, all the states are aggregated according to the number of failed components they yield. Under these conditions, the steady state probabilities can be calculated piecewise, one aggregate state at a time. This leads to acceptable computation costs (inversion of small matrices), but the computation of the bounds may require a high number of iterations to converge. In this thesis, this fact is illustrated on a modified example taken from [Muntz et al. 1989], where bounds on the steady state probability of aggregates are calculated. The bounds are far from one another for the aggregates representing the highest numbers of failures, on this example. It is not always possible to deduce the unavailability from the number of failed components. This is especially true for the taken example, as it is an extension of the fault tolerant database that is presented in Section 3.2.1. Hence, the probability of the aggregates should not be used to approximate the unavailability of the system.

1.3. Contributions to the Quantitative Safety Analyses of Large and Complex Systems with Markov Chains

The partial generation of Markov chains is aimed at fighting the inherent combinatorial explosion of Markov chains. The path-based calculation approaches and the partial generation of Markov chains share a common objective. This objective is to compute the safety indicators of repairable Markovian systems whose Markov chain would be too large to be generated and calculated.

The partial generation method is based on the observation that a repairable system is, most of the time, in its nominal state. This leads to the idea of the selection of the most probable states of a system. This selection should be done during the generation of the Markov chain, to avoid the construction of a very large chain that would be reduced afterward.

The main contributions of this thesis are to provide partial generation of Markov chains from

AltaRica models, using a variation of the very efficient Dijkstra's algorithm. The selection of the most probable states is based on a relevance factor associated to each state of the system. These concepts were introduced in [Brameret et al. 2012, 2013]. The generation of the partial chain is very efficient, as the cost of the partial generation is comparable to the cost of the calculation of the partial Markov chain.

The error introduced by the partial generation cannot be calculated with the partial chain. Bounds on the indicators of the system are introduced in [Brameret et al. 2015]. The bounds are obtained by introducing a new state in the Markov chain, at the end of the exploration of the chain. This state is called the sink and is, by definition, an absorbing state. The sink gathers all the states that are not explored at the end of the exploration. Bounds on the indicators are computed knowing the probability to be in the sink. Such bounds provide a measure of the error caused by the partial generation. Hence, the error introduced by the partial generation is known only *after* the Markov chain is calculated.

Although the sink is very simple and provides very conservative bounds, it is not computationally efficient to improve it. The sink can be seen as a tradeoff between more knowledge on the obtained indicators of the system and the accuracy of the partial generation.

Software tools are developed in the context of the AltaRica 3.0 project. The partial generation method is tested on several examples with these tools. They also provide means to analyze the strengths and weaknesses of the partial generation with respect to the experimental conditions: parameters of the system under study, chosen mission time, size of the partial chain, and so on.

2. Core Algorithm, Generation of the Most Useful Part of a Markov Chain

In this chapter are introduced the fundamentals about the partial generation method. As the partial generation methods takes root in the Markov chains calculation, it is natural to introduce Markov chains first. The core idea behind the method is that, in a safe system, the probability is concentrated in the nominal state, and in a few of the states reachable from the nominal state. The goal of the partial exploration is to predict which states will be useful in the assessment of the probabilistic indicators of a system, and which ones will not. Obviously, this should be done without calculation nor construction of the whole state space of the system.

This chapter is organized as follows. Section 2.1 introduces Markov chains and their calculation. Section 2.2 shows an alternative interpretation of the Markov chains, which provides another perspective on the Markov chains. Section 2.3 presents the core of the method, the generation of a partial Markov chain based on Dijkstra's algorithm and on a relevance factor. In Section 2.4, a variation of the method is introduced to calculate an upper bound on the error caused by the partial generation of the Markov chain.

2.1. Markov Chains: Definitions, and Calculation

Markov chains were introduced by Andrey Markov in [Markov 1906]. Since then, they are used in many fields to model and calculate systems of various nature. His work was translated and republished in [Markov 1971].

2.1.1. Introduction

To avoid confusing definitions taken from different publications, we chose to work only with definitions taken from William J. Stewart, *Introduction to the numerical solution of Markov chains* [Stewart 1994]. Numerous elements from this section are taken from this reference book.

It is often possible to represent the behavior of a system by describing all the different states that the system can occupy and by indicating how it moves from one state to another in time. If the transition rates between states are constant and if the time spent in any state is exponentially distributed, the system may be represented by a Markov process. The system being modeled by the process is assumed to occupy one and only one of these states at any moment in time. The evolution of the system is represented by the transitions of the Markov process from one state to another. These transitions are assumed to occur instantaneously.

The fundamental property of a Markovian system, referred to as the *Markov property*, is that the future evolution of the system depends only on the current state of the system and not on its past history. This is often seen as the *memoryless* property.

The term *Markov chain* is employed when the state space is discrete. The information that is most often sought from such a model is the probability of being in a given state or subset of states at a certain time after the system becomes operational. Often this time is taken to be sufficiently long that all influence of the initial starting state has been erased. The probabilities thus obtained are referred to as the stationary, or *steady-state* probabilities. Probabilities at a particular time t are called *transient probabilities.* The transient or steady-state probabilities are used to predict the behavior of the system.

For these reasons, Markov chains are a common approach to System Safety Assessment. They are one of the possible tools that are proposed in the ARP 4761 standard [SAE 1996]. This standard gives guidelines and methods for conducting the safety assessment process on civil airborne systems and equipment.

Markov chains are a versatile tool, and can model a wide variety of behaviors. However, in an industrial context, Markov chains are not written by hand, because their large size makes this task prohibitive and error-prone. This is why Markov chains are usually obtained from higher level modeling languages. In this document, we use AltaRica 3.0.

When the number of states is small, it is relatively easy to obtain transient and stationary solutions quickly and accurately. From these, the behavior of the system is accurately predicted. When the models become more detailed, the calculation of these solutions becomes much more difficult.

2.1.2. Definitions

In this section, only definitions required for the good understanding of this thesis are given. For a discrete-time Markov chain (DTMC), we observe the state of a system at a discrete set of times. If the Markov chain may change state at any point of time, we say that the chain is a continuous-time Markov chain (CTMC).

Discrete-Time Markov Chain

In an homogeneous DTMC, the transition probabilities are constant, and the time spent in each state is geometrically distributed. In the remainder of the thesis, DTMC denotes an homogeneous DTMC.

Let p_{ij} be the *transition probability* from State i to State j, which is the probability to go from State i to State j after each time step. The square matrix P, formed by placing p_{ij} in row i and column j for all states i and j, is called the *transition matrix* or *chain matrix.*

$$P = [p_{ij}]_{\forall i,j}$$

All the elements of the matrix P satisfy the following two properties.

$$\begin{cases} \forall i, j & 0 \leq p_{ij} \leq 1 \\ \forall i & \sum\limits_{\forall j} p_{ij} = 1 \end{cases}$$

Let n be a given time step. Let $x_i^{(n)}$ be the probability to be in State i at time step n, and $X^{(n)}$ be the row vector formed by all the $x_i^{(n)}$. From the Chapman-Kolmogorov equation, $X^{(n)}$ and $X^{(0)}$ satisfy:

$$X^{(n)} = X^{(0)} \cdot P^n$$

As $X^{(n)}$ is the vector of state probabilities:

$$\forall n, \quad \sum_{\forall i} x_i^{(n)} = 1$$

Continuous-Time Markov Chain

In a CTMC, the Markov chain can change state at any point in time. The time spent in each state is exponentially distributed if the CTMC is homogeneous.

Because of their continuous-time characteristics, the probabilities of changing state depend on the time. Let $p_{ij}(s,t)$ the probability of being in State j at time t, given that the state is i at time s (with $t \geq s$). If the CTMC is homogeneous, $p_{ij}(s,t)$ does not depend on s nor on t, but only on the length of the time interval $\tau = t - s$. Hence,

$$\forall s \ \forall \tau, \quad p_{ij}(s, s+\tau) = p_{ij}(\tau)$$

In the remainder of this thesis, a “Markov chain” denotes an homogeneous continuous-time Markov chain.

To work more easily with homogeneous CTMC, the notion of transition rate is introduced. Let q_{ij} be the transition rate from State i to State j. By definition,

$$q_{ij} = \frac{\mathrm{d}p_{ij}(\tau)}{\mathrm{d}\tau} = \lim_{\tau \to 0} \frac{p_{ij}(\tau)}{\tau}$$

The matrix Q, formed by the q_{ij}, is called the *transition rate matrix.* Because of the conservation of the probabilities, the elements of Q satisfy the following property:

$$q_{ii}(t) = -\sum_{j \neq i} q_{ij}(t)$$

Let $\pi_i(t)$ be the probability that the system is in state i at time t, and $\pi(t)$ the row vector formed by all the $\pi_i(t)$. The Chapman-Kolmogorov equation for CTMC is used to obtain the following set of ordinary differential equations:

$$\forall j, \quad \frac{\mathrm{d}\pi_j(t)}{\mathrm{d}t} = \sum_{\forall k} \pi_k(t) \cdot q_{kj}$$

which can be more simply written as:

$$\frac{\mathrm{d}\pi(t)}{\mathrm{d}t} = \pi(t) \cdot Q \tag{2.1}$$

To solve such system, it is important to recall that:

$$\forall t, \quad \sum_{\forall i} \pi_i(t) = 1 \qquad ; \qquad \pi(t=0) \quad \text{is known}$$

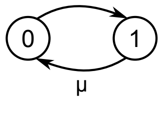
It follows that the solution $\pi(t)$ is given by:

$$\pi(t) = \pi(0) \cdot e^{Qt} \quad \overset{\text{def}}{=} \pi(0) \cdot \left(I + \sum_{n=1}^{\infty} \frac{Q^n t^n}{n!} \right) \tag{2.2}$$

The challenge of the numerical solution of Markov chains is to calculate such series. This challenge is further addressed in Section 2.1.4.

2.1.3. Illustration, the Case of a Repairable System

Consider a simple repairable system, whose availability is to be determined, and which can be in two states: the working state and the failure state. When the system is working, it can fail, and when the system is failed, it can be repaired. The failure rate λ is the expected number of failures per time interval (say an hour). The repair rate μ is the expected number of repairs per time interval. To model this system with a Markov chain, we suppose that the system is *memoryless*. In other words, the probability of a failure (or repair) only depends on the *current state* of the system, and does not depend on the time that the system spent in the current state (these rates do not depend on the overall age of the system).



$$\begin{cases} \dfrac{\mathrm{d}P_0(t)}{\mathrm{d}t} = -\lambda \cdot P_0(t) + \mu \cdot P_1(t) \\ \dfrac{\mathrm{d}P_1(t)}{\mathrm{d}t} = \lambda \cdot P_0(t) - \mu \cdot P_1(t) \\ P_0(t) + P_1(t) = 1 \end{cases} \qquad \begin{cases} P_0(0) = 1 \\ P_1(0) = 0 \end{cases}$$

Figure 2.1.: A small Markov chain, the corresponding set of ordinary differential equations, and the initial distribution.

The homogeneous CTMC for this elementary system has two states and two transitions (see Figure 2.1). State 0 is the working state, and State 1 is the failure state. The probability $P_0(t)$ is the probability to be in State 0 over time ($P_1(t)$ for State 1). As State 0 is the working state, the availability of the system is $P_0(t)$, and its unavailability is $P_1(t)$. This system is supposed to be initially working, that is to say $P_0(0) = 1$ and $P_1(0) = 0$. For this simple system, the transient solution is:

$$\begin{cases} P_0(t) = \dfrac{\mu}{\lambda+\mu} + \dfrac{\lambda}{\lambda+\mu} e^{-(\lambda+\mu)\cdot t} \\ P_1(t) = \dfrac{\lambda}{\lambda+\mu} - \dfrac{\lambda}{\lambda+\mu} e^{-(\lambda+\mu)\cdot t} \end{cases} \tag{2.3}$$

The asymptotic probabilities are easily computed:

$$\begin{cases} P_0(\infty) = \dfrac{\mu}{\lambda+\mu} \\ P_1(\infty) = \dfrac{\lambda}{\lambda+\mu} \end{cases}$$

Hence, the steady state unavailability of the system is $\frac{\lambda}{\lambda+\mu}$, and the steady state availability is $\frac{\mu}{\lambda+\mu}$.

A numerical representation of the functions from Equation (2.3) is given in Figure 2.2. For practical reasons, λ and μ were chosen as follows:

$$\lambda = 1.0 \times 10^{-3} \text{ hours}^{-1} \qquad \mu = 1.0 \times 10^{-2} \text{ h}^{-1}$$

Figure 2.2 shows the basic equilibrium between the availability and the unavailability of a system, and their overall evolution with time.

Let us focus on the transient evolution of the solution. The solution of the homogeneous ordinary first order differential equation $\frac{\mathrm{d}P(t)}{\mathrm{d}t} + \frac{P(t)}{\tau} = 0$ is known to be $P(t) = C \cdot \exp(-t/\tau)$, where C is a constant determined by the value of P at time 0. The constant τ is the time constant for the evolution of solution, and characterizes the time taken by the function P to stabilize. Consider now the expressions of $P_0(t)$ and $P_1(t)$ in the set of equations (2.3):

$$\tau = \frac{1}{\lambda+\mu} \tag{2.4}$$

The exponential decay leads to the stabilization of the solutions:

$$\begin{cases} t = \tau, & P_0(\tau) = \dfrac{\mu}{\lambda+\mu} + 0.37 \cdot \dfrac{\lambda}{\lambda+\mu} & P_1(\tau) = \dfrac{\lambda}{\lambda+\mu} - 0.37 \cdot \dfrac{\lambda}{\lambda+\mu} \\ t = 3\cdot\tau, & P_0(3\cdot\tau) = \dfrac{\mu}{\lambda+\mu} + 0.050 \cdot \dfrac{\lambda}{\lambda+\mu} & P_1(3\cdot\tau) = \dfrac{\lambda}{\lambda+\mu} - 0.050 \cdot \dfrac{\lambda}{\lambda+\mu} \\ t = 5\cdot\tau, & P_0(5\cdot\tau) = \dfrac{\mu}{\lambda+\mu} + 0.0067 \cdot \dfrac{\lambda}{\lambda+\mu} & P_1(5\cdot\tau) = \dfrac{\lambda}{\lambda+\mu} - 0.0067 \cdot \dfrac{\lambda}{\lambda+\mu} \\ t = 7\cdot\tau, & P_0(7\cdot\tau) \approx \dfrac{\mu}{\lambda+\mu} & P_1(7\cdot\tau) \approx \dfrac{\lambda}{\lambda+\mu} \end{cases}$$

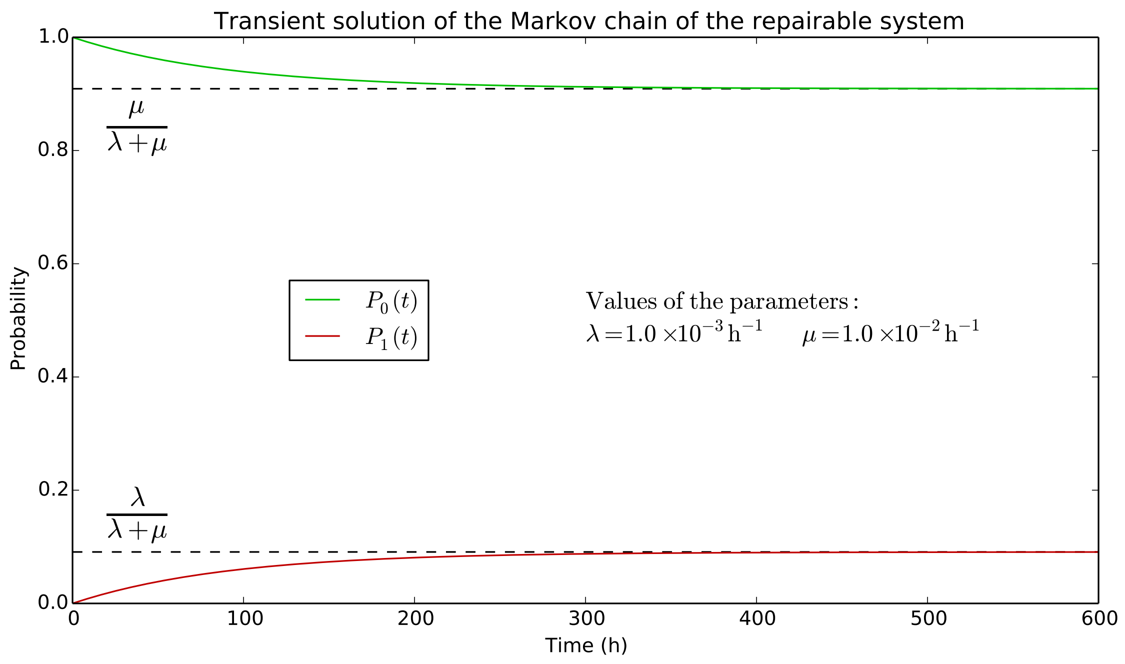


Figure 2.2.: The transient solution of the Markov chain given in Figure 2.1.

τ is an important constant that represents the evolutions of the system over time. The system is transient while t is below $5 \cdot \tau$. For $t \gg \tau$, the numerical solutions of the system are stable.

2.1.4. Transient Solutions of Markov Chains

The most intuitive way to calculate the transient probabilities of the states of a Markov chain is to solve the set of differential equations (see Equation (2.1)). Several methods have been developed, which can be sorted in two families: the analytical or the numerical approaches. However, for Markov chains with hundreds of states, the analytic approach is not practical, as it involves the search of the eigenvalues and vectors of the transition rate matrix.

Numerous methods exist to numerically calculate the transient solution of a Markov chain (see [Stewart 1994]). Only three strategies are presented here.

Some methods are based on the solution of the set of linear ordinary differential equations expressed as a series (see Equation (2.2)). This method computes the approximate value of the series for the given time t, so it is not suitable to compute the transient probabilities. Moreover, this direct solution is rather hard to compute, due to the rounding errors when computing the series (see [Moler and Van Loan 1978, 2003]).

Uniformization techniques introduced by [Grassmann 1977] use a DTMC as a support for the calculations. The transition matrix of the DTMC has the same size as the generator matrix of the CTMC. The formula to compute the solution is a very similar series involving the newly constructed transition matrix, which computes the solution at a given time t. However, the number of useful terms is usually less than a dozen in order to obtain accurate results. If the time t is too long, it may be required to divide the time in smaller time-steps, which increases the computation time. According to [Reibman and Trivedi 1988] and [Stewart 1994], uniformization may be irrelevant for stiff chains (chains where there are very low and very high transitions rates). It is, however, algorithmically efficient and parallelizable (see [Dingle et al. 2004]).

Iterative methods uses a discrete time approximation to calculate the exponentiation of the transition rate matrix Q. Time is sampled, using a small enough time-step dt, which leads to the computation of the transient probabilities. Some of these methods are introduced in [Stewart 1994] and tested on different examples that are related to safety assessment in [Rauzy 2004]. We will focus on the full exponentiation matrix (EXP) and the Forward Euler Method (FEM) inspired from [Euler 1768]. The discrete time approximation is based on the solution seen in Equation (2.2), by introducing the time-step dt:

$$\pi(t+dt) = \pi(0) \cdot e^{(t+dt)\cdot Q} = \pi(0) \cdot e^{t\cdot Q} \cdot e^{dt \cdot Q} = \pi(t) \cdot e^{dt\cdot Q}$$

With $t = n \cdot dt$:

$$\pi(t) = \pi(0) \cdot \left(e^{dt\cdot Q}\right)^n$$

Both EXP and FEM rely on the definition of the matrix exponential of the transition rate matrix Q times the time-step dt:

$$e^{dt\cdot Q} = \sum_{k=0}^{+\infty} \frac{(dt \cdot Q)^k}{k!} \tag{2.5}$$

The EXP method calculates $e^{dt\cdot Q}$ until $\frac{(dt\cdot Q)^k}{k!}$ is low enough. The FEM method keeps only the first two terms of the series: $e^{dt\cdot Q} \approx I + dt \cdot Q$. The time step dt must be low enough so that the probabilities calculated in one step stay below 1: $dt < \frac{1}{\max_i(-q_{ii})}$ (as q_{ii} are negative). It can be noted that $e^{dt\cdot Q}$ is calculated only once in order to obtain the values of the transient probabilities from time 0 to time t with time-step dt.

It has been shown in [Rauzy 2004] that FEM gives good results with respect to the accuracy of the transient solution. The accuracy can be adjusted through $dt = \frac{\rho}{\max_i(-q_{ii})}$ where $\rho \in]0..1]$. The quantity ρ is a tradeoff between the quality of the computations (when $\rho \to 0$) and the speed (when $\rho = 1$).

Let $|E|$ be the number of non-null transitions in the transition rate matrix, $|V|$ the number of states, T the time at which the Markov chain is solved, and dt the chosen time-step. FEM runs in worst-case $\mathcal{O}(|E| \times \frac{T}{dt})$ time because it does at most $\frac{T}{dt}$ iterations involving the multiplication of the sparse transition rate matrix containing only $|E|$ elements. Data structures in FEM use $\mathcal{O}(|E| + |V|)$ space, as it only stores the sparse matrix $I + dt \cdot Q \approx e^{dt\cdot Q}$ and two distribution vectors.

2.2. Understanding Markov Chains of Large and Safe Systems

2.2.1. Physical Interpretations of a Markov Chain

From a physical point of view, a Markov chain is similar to a network of water tanks connected with pipes which have different diameters. Indeed, the quantity of water that flows from Tank i to Tank j is proportional to the diameter of the pipe and to the pressure in the upstream tank. This pressure is proportional to the height of the water in the tank.

The level of water in a tank is analogous to the probability of a state of a Markov chain, and the diameters of the pipes is analogous to the transition rates of the transitions. Of course, the analogy is incomplete as water flows are driven by gravity, while probability flows are driven by the following ordinary equation:

$$\frac{\mathrm{d}P_i(t)}{\mathrm{d}t} = -(\text{outgoing rate}) \times P_i(t) + \sum_{j \in \text{parents}} (\text{ingoing rate}) \times P_j(t)$$

where $P_i(t)$ is the probability to be in State i at time t.

However, because this point of view illustrates how the probability evolves between states, it is very useful to understand one of the core principles of the method. The selection of the states is based on the flow of the probability from the initial state to all the other states (see Section 2.3.2). This selection uses the competition between outgoing states of a given state. With water tanks, the problem is formulated as follows. Knowing that the initial tank is full and not replenished, the goal is to *estimate* the level of the water in all the other tanks, without calculation of the Markov chain, in order to keep only the fullest tanks.

The idea is to focus on one full Tank i until it is empty, and see where the water goes. Let α_{ij} be the quantity of water that goes to Tank j. It is easy to see that α_{ij} is proportional to the diameter of the pipe q_{ij} between i and j. However, the outgoing pipes are concurrent, as they all empty the same water tank. So the part that goes to j is a ratio between the diameter of the pipe divided by the sum of the diameters of the pipes that goes out of Tank i:

$$\forall j, \quad \alpha_{ij} = \frac{q_{ij}}{\sum_{k \neq i} q_{ik}} \tag{2.6}$$

This idea gives the same result with the probabilities. Considering that the Markov chain is currently in State i, let t be the time at which a transition will occur, and α_{ij} be the probability that the State j is reached at time t. Because of the set of Equation (2.1), α_{ij} is proportional to q_{ij}. Moreover, as a transition occurs at time t, $\sum_{j \neq i} \alpha_{ij} = 1$. Then, the same equation is obtained.

Equation (2.6) is important, as it shows the competition between states. The probability to go from State i to State j is driven by q_{ij} *with respect to* the other outgoing transitions $q_{ik}, \forall k \neq i$. Consequently, if a repair and a failure may happen in the same state, as the repair rate is usually very high compared to the failure rate, the repair is very more likely taken.

2.2.2. Probability Distributions in Large Systems

For safe systems, repairs are far quicker than failures. Under these conditions, we suppose that $\lambda \ll \mu$ (that is to say $\frac{\lambda}{\mu} < \frac{1}{100}$). The transient solution of a modified version of the 2-state Markov chain from Figure 2.1 is plotted in Figure 2.3. The chosen parameters give $\frac{\lambda}{\mu} = \frac{1}{1000}$. Only $P_1(t)$, the unavailability, is plotted.

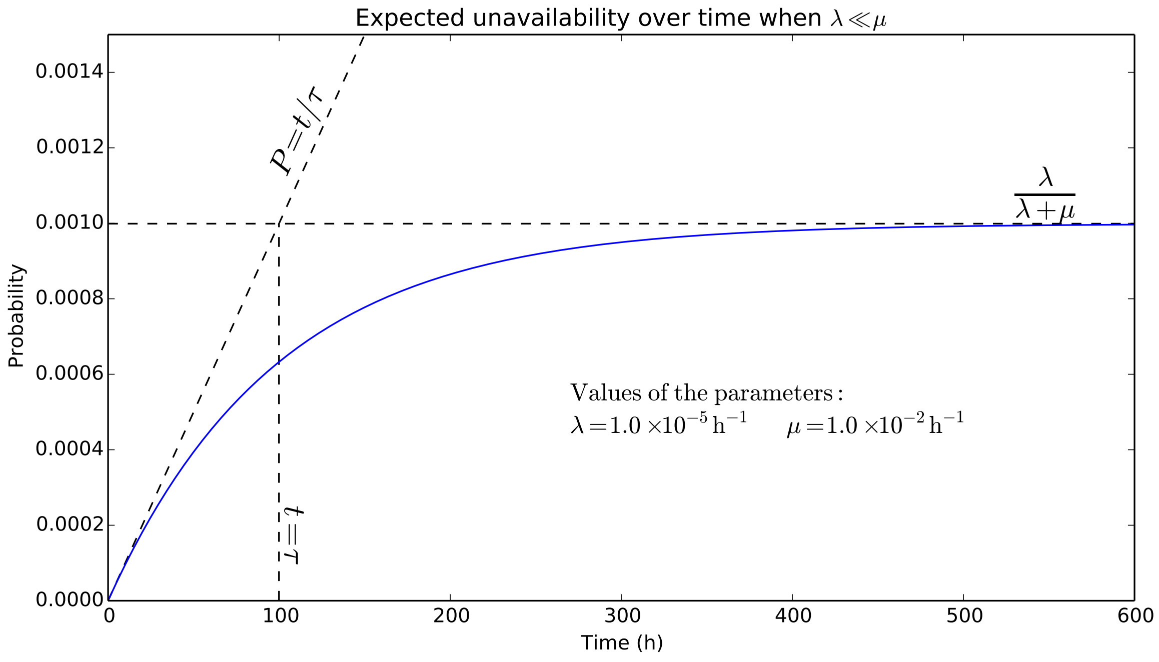


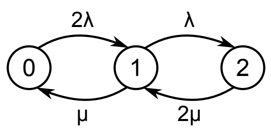
Figure 2.3.: The transient solution of the simple Markov chain from Figure 2.1 for a safe system ($\mu = 1000\lambda$).

Since $\lambda \ll \mu$, the asymptotic unavailability $\frac{\lambda}{\lambda+\mu}$ is roughly $\frac{\lambda}{\mu}$, and the time constant $\tau = \frac{1}{\lambda+\mu}$ becomes $\tau \approx \frac{1}{\mu}$. These constants are very important as they give *rough estimates* about a system, and can be extended to larger systems as follows.

Figure 2.4 shows a Markov chain which extends the previous one. Here, a system with two identical and independent components is modeled. State 0 is the initial state, where both components are working. State 1 is a state where one of the two components is failed, and the other one is working. In state 2, both components are failed.

This system can be solved analytically, and the following expressions are obtained:

$$\begin{cases} P_0(t) = \dfrac{\mu^2}{(\lambda+\mu)^2} + \dfrac{2\mu\lambda}{(\lambda+\mu)^2}e^{-(\lambda+\mu)t} + \dfrac{\lambda^2}{(\lambda+\mu)^2}e^{-2(\lambda+\mu)t} \\ P_1(t) = \dfrac{2\mu\lambda}{(\lambda+\mu)^2} + \dfrac{2\lambda(\lambda-\mu)}{(\lambda+\mu)^2}e^{-(\lambda+\mu)t} + \dfrac{-2\lambda^2}{(\lambda+\mu)^2}e^{-2(\lambda+\mu)t} \\ P_2(t) = \dfrac{\lambda^2}{(\lambda+\mu)^2} + \dfrac{-2\lambda^2}{(\lambda+\mu)^2}e^{-(\lambda+\mu)t} + \dfrac{\lambda^2}{(\lambda+\mu)^2}e^{-2(\lambda+\mu)t} \end{cases} \tag{2.7}$$



$$\begin{cases} \dfrac{\mathrm{d}P_0(t)}{\mathrm{d}t} = -2 \cdot \lambda \cdot P_0(t) + \mu \cdot P_1(t) \\ \dfrac{\mathrm{d}P_1(t)}{\mathrm{d}t} = 2 \cdot \lambda \cdot P_0(t) - (\lambda + \mu) \cdot P_1(t) + 2 \cdot \mu \cdot P_2(t) \\ \dfrac{\mathrm{d}P_2(t)}{\mathrm{d}t} = \lambda \cdot P_1(t) - 2 \cdot \mu \cdot P_2(t) \\ P_0(t) + P_1(t) + P_2(t) = 1 \end{cases} \qquad \begin{cases} P_0(0) = 1 \\ P_1(0) = 0 \\ P_2(0) = 0 \end{cases}$$

Figure 2.4.: A Markov chain with 3 states, its set of ordinary differential equations, and its initial distribution.

The asymptotic probabilities are immediately calculated, and can be expressed in another form to show that the solutions for the system with two components can be deduced from the solutions from the simpler system shown in Figure 2.1:

$$\begin{cases} P_0(\infty) = \dfrac{\mu^2}{(\lambda+\mu)^2} & = & \left(\dfrac{\mu}{\lambda+\mu}\right)\left(\dfrac{\mu}{\lambda+\mu}\right) \\ P_1(\infty) = \dfrac{2\mu\lambda}{(\lambda+\mu)^2} & = & 2\left(\dfrac{\lambda}{\lambda+\mu}\right)\left(\dfrac{\mu}{\lambda+\mu}\right) \\ P_2(\infty) = \dfrac{\lambda^2}{(\lambda+\mu)^2} & = & \left(\dfrac{\lambda}{\lambda+\mu}\right)\left(\dfrac{\lambda}{\lambda+\mu}\right) \end{cases} \tag{2.8}$$

From this, it is possible to give a rough estimate of the asymptotic probabilities of each state of the Markov chain for a system including more components:

$$k \cdot \left(\frac{\mu}{\lambda+\mu}\right)^a \cdot \left(\frac{\lambda}{\lambda+\mu}\right)^b$$

where a is the number of working components, b the number of failed component, and k the number of possible combinations of a working components in $(a+b)$ components. This equation is of course only valid for a system with $(a+b)$ identical and independent binary components.

In the case of the system with two components, where $\lambda \ll \mu$, $\frac{1}{\lambda+\mu} \approx \frac{1}{\mu}$ and these expressions become:

$$\begin{cases} P_0(\infty) \approx 1 - \epsilon \\ P_1(\infty) \approx 2 \cdot \dfrac{\lambda}{\mu} \\ P_2(\infty) \approx \left(\dfrac{\lambda}{\mu}\right)^2 \end{cases} \tag{2.9}$$

where ϵ is positive and small. ϵ can be expressed either to verify $\sum P_i(t) = 1$, which gives $\epsilon = 2\frac{\lambda}{\mu} + \left(\frac{\lambda}{\mu}\right)^2$, or as the series expansion of $\frac{1}{1+\frac{\lambda}{\mu}} = 1 - \frac{\lambda}{\mu} + \left(\frac{\lambda}{\mu}\right)^2 + o\left(\left(\frac{\lambda}{\mu}\right)^2\right)$ with the small o notation, which gives $\epsilon = 2\frac{\lambda}{\mu} - 3\left(\frac{\lambda}{\mu}\right)^2$.

The set of approximate asymptotic solutions (2.9) is very important, as it shows the rough approximation of the probability distributions in a system with redundant components. The probability to be in a state where n components are failed in the same time is in $\left(\frac{\lambda}{\mu}\right)^n$, which is $\lesssim 0.001^n$ for safe systems. In this respect, a *notion of distance* from the nominal state is emerging. The nominal state has roughly the probability 1 (minus a little something), and the states that are distant from the nominal states through n stages of failures are very improbable. This explains why the probability is mostly concentrated around the nominal state for safety systems.

Now consider the possible evolutions from State 1. The probability to be in State 1 can flow in the two other states. The probability rate to go back in state 0 is μ (through a repair), and the flow to go in 2 is λ (through a failure). When $\mu \gg \lambda$, most of the probability flows to 0, and very little probability flows to 2 (see Equation (2.6)). In this sense, 0 and 2 *are competing* to obtain probability from 1. There will be $\frac{\mu}{\lambda+\mu} \approx 1$ of the probability in 0, and $\frac{\lambda}{\lambda+\mu} \approx \frac{\lambda}{\mu}$ of the probability in 2.

An interesting aspect of the numerical transient solutions of the Markov chains is related to the time constant of the system τ. The time constant for each component is around $\frac{1}{\mu}$. The time at which each component is in its steady state is roughly $5 \cdot \frac{1}{\mu}$. When the system is built upon different components, the transient probability of each state is influenced by the evolution of each component. As a consequence, the fastest evolution of each transient probability is comparable to $\frac{1}{\max \mu}$. However, the steady state of the system is reached when all the components are stabilized. This happens when all the components reach their steady state. As a consequence, the system is steady state when the time is comparable to $\frac{1}{\min \mu}$ (see Equation (2.4)). This may lead to a stiff Markov chain if the system is composed of both quickly and slowly repairable components.

When a Markov chain models a repairable and safe system, the most probable states will be the states close to the nominal state. This is because the probability from the nominal states flows slowly out of it due to low failure rates, and goes back rapidly to the nominal state through repairs with high repair rates. The states close to the nominal state will gather all the probability, while remote states will have very low probabilities. Hence, the performance indicators of the system are mainly influenced by the states close to the nominal state. The goal of the method is to select only those states.

2.3. Partial Generation: the Method

Modeling a system with Markov chains consists in enumerating the different states in which the system can be, and calculating the transition rates between these states. Writing a Markov chain *by hand* quickly becomes unmanageable due to the number of states and transitions (the oil production system detailed in Section 3.2.3 has 15 basic components, but nearly $47,000$ states and $5,210,000$ transitions). The goal of the partial generation method is to generate the Markov chain from a higher modeling language, and keep only the most probable states of the chain before the chain is completely generated. This is possible as the Markov chains are obtained from a higher level modeling language.

In practice, an AltaRica 3.0 model is used to generate a Guarded Transition System (GTS), which holds the semantics of the model. GTS are detailed in Appendix A. A GTS $\langle V, E, T, A, \iota\rangle$ is animated to partially generate the states and transitions of the system as a reachability graph $\Gamma = \langle\Sigma, \Theta\rangle$, where Σ is the set of nodes of the system, and Θ the set of edges. The reachability graph is finally transformed into a Markov chain. This final transformation is straightforward, as nodes of the reachability graph are states of the Markov chain, and the edges are the transitions.

The exploration and partial generation *must be efficient enough.* If the partial generation is not efficient enough, the calculation of the transient solution of the whole chain may be faster than the partial generation and the partial generation would be useless.

In order to keep the most probable states, *a sort among states has to be done.* This sort must be done before the chain is fully generated, and must be done efficiently.

How to Sort States?

The idea is to keep the states that are the closest to the nominal state[1], as shown in Figure 2.5. To do so, a *relevance factor* is associated to the nominal state. The algorithm computes the relevance factor of the other states from the nominal state. The relevance factor is then used to sort the states.

As seen in Section 2.2, the relevance factor of a state must take into account the "depth" from the nominal state (in terms of the number of consecutive failures), and the competition between states (and avoid branches of the Markov chain which only have improbable states).

The calculation of the relevance factor for the states is based on a specific use of Dijkstra's algorithm developed in [Dijkstra 1959]. The main benefits of Dijkstra's algorithm are the efficiency of the algorithm (states and transitions are only visited once) and the fact that it can be interrupted. Both aspects are important, because the computation of the partial Markov chain must not cost more than the calculation of the full chain.

[1]The "nominal state" is an expert knowledge and is given as ι, the initial state of the GTS $\langle V, E, T, A, \iota\rangle$.

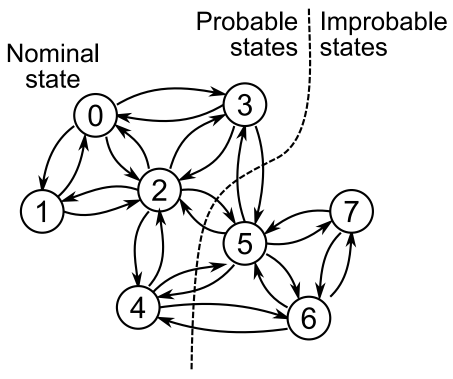


Figure 2.5.: Strategy used to select states in a Markov chain. States are indexed according to their relevance factor.

In the remainder of this section, first we present Dijkstra's algorithm, then we detail the relevance factor. Finally, the main steps from the Guarded Transition System to the indicators of the safety assessment are presented.

2.3.1. Dijkstra's algorithm

Dijkstra's algorithm finds the shortest paths in a graph, i.e. it minimizes the distance $d(\sigma)$ of each node $\sigma \in \Sigma$ of the graph from a chosen source node $\sigma_0 \in \Sigma$. This distance is the sum of the length of the transitions l over the shortest possible path. The length l must always be positive.

Dijkstra's algorithm organizes nodes of the graph being explored in two sets: the set Σ of states which are already explored (i.e. nodes whose distance is minimized), and the set C of candidates (i.e. nodes which are candidates for the next exploration round).

The core idea of Dijkstra's algorithm is to pursue the exploration of the graph with the candidate σ_{min} that has the smallest distance from the source node. As l is always positive, it is not possible to find any shorter path to σ_{min} in the future rounds of exploration. Hence, σ_{min} can be added to Σ. Then, the successors of σ_{min} are examined. Let τ be such a successor. If $\tau \in \Sigma$, $d(\tau)$ is already minimized, and τ is skipped. If $\tau \in C$, $d(\tau)$ is set to the minimum of the current value $d(\tau)$ or $d(\sigma) + l(\sigma \to \tau)$, where $l(\sigma \to \tau)$ is the length l associated to the edge $\sigma \to \tau$. Otherwise, τ is added to C and $d(\tau)$ is set to $d(\sigma) + l(\sigma \to \tau)$. Algorithm 2.1 shows these operations. Initially, there is only the source node in the set C, and $d(\sigma_0) = 0$.

Dijkstra's algorithm visits only once every edge of the graph and finds the minimum distance between the source node to all the other nodes. The worst-case run-time of the algorithm heavily relies on the data structures used to implement the sets Σ and C. Theoretical minimum complexity is achieved with Fibonacci heaps [Fredman and Tarjan 1984] [Brodal et al. 2012], which gives worst-case $\mathcal{O}(|\Theta| + |\Sigma| \log |\Sigma|)$ time, where Θ is the set of edges, and Σ the set of explored states. Dijkstra's algorithm is a good candidate for the partial exploration, as

Algorithm 2.1: Dijkstra's algorithm

```
1  while $C \neq \emptyset$ do
2      let $\sigma$ be the candidate with the smallest $d$ in $C$
3      remove $\sigma$ from $C$, add it to $\Sigma$
4      foreach successor $\tau$ of $\sigma$ do
5          if $\tau \in C$ and $d(\tau) > d(\sigma) + l(\sigma \to \tau)$ then
6              $d(\tau)$ is set to $d(\sigma) + l(\sigma \to \tau)$
7          else if $\tau \notin C$ and $\tau \notin \Sigma$ then
8              $\tau$ is added to C
9              $d(\tau)$ is set to $d(\sigma) + l(\sigma \to \tau)$
10         transition $\sigma \to \tau$ is recorded
```

it does not cost more than the calculation of the transient solution of the resulting Markov chain. Details of the chosen implementation are given in Section 4.2.1.

Dijkstra's algorithm can be interrupted before the complete exploration of the graph, and the resulting Σ will contain all the closest nodes to the initial node. Moreover, the distance is minimized for nodes in Σ. The distances of the remaining nodes in C are not minimized yet, as there might be undiscovered shorter path leading to them.

Adapting Dijkstra's algorithm

Dijkstra's algorithm finds the *minimized distance* for all the nodes in the graph, summing the lengths of the edges along the shortest paths. These concepts can be abstracted, and the fundamental idea of Dijkstra's is to know the *order* in which the exploration is done, to *optimize* a *quantity* $q(\sigma)$ associated to each node σ of a graph. This quantity is calculated through a *cost function* $\text{cost}(\sigma, \epsilon)$ associated to the edges ϵ of the graph. The cost function must be monotonic, i.e. there must exist a partial order $\preceq$ between the quantities $q(\sigma)$ and $q(\tau)$ for all nodes τ that are reachable from node σ:

$$\forall \sigma \quad \forall (\epsilon : \sigma \to \tau), \quad q(\tau) \preceq q(\sigma)$$

The classical Dijkstra's algorithm is based on the following cost function:

$$q(\tau) = \text{cost}(\sigma, \epsilon) = q(\sigma) + c, \qquad 0 \leq c$$

Now, consider the following cost function:

$$q(\tau) = \text{cost}(\sigma, \epsilon) = q(\sigma) \times c, \qquad 0 \leq c \leq 1$$

If q is positive for the initial state, q is also positive for the other nodes, as $k \geq 0$. The cost function is monotonic, as q is positive and $q(\tau) = \text{cost}(\sigma, \epsilon) \leq q(\sigma)$. The quantity associated to the nodes will be maximized by Dijkstra's algorithm.

The relevance factor we propose is similar to the latter quantity.

2.3.2. The Relevance Factor

As the relevance factor is used to select the states that are kept in the final Markov chain, it is a key concept of the method. Here we develop the chosen relevance factor.

The relevance factor is based on access probabilities of remote states from the initial state. It uses the concepts shown in Section 2.2, and uses the Equation (2.6), which takes into account the competition between outgoing transitions of a given state:

$$\frac{\lambda_{\sigma\to\tau}}{\sum_k \lambda_{\sigma\to k}}$$

Figure 2.6 shows a small Markov chain with transition rates. The relevance factor R should take into account the competing transitions, so $R(D)$ should be $R(A) \times \frac{4}{5} \times \frac{1}{5}$. The relevance factor should be maximized for the nodes, so that the most influential nodes are kept.

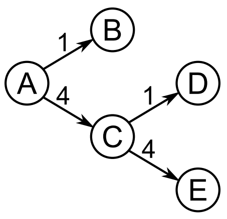


Figure 2.6.: A small Markov chain with transition rates labeling the transitions.

The relevance factor $R(\tau)$ of state j is defined as follows:

$$R(\tau) \overset{\text{def}}{=} \max_{\text{parents } \sigma} \left(R(\sigma) \times \frac{\lambda_{\sigma\to\tau}}{\sum_k \lambda_{\sigma\to k}} \right) \tag{2.10}$$

where σ are states from which state τ is reachable (with rate $\lambda_{\sigma\to\tau}$), and transitions $\sigma \to k$ are the other transitions going out of each the state σ.

Dijkstra's algorithm maximizes the relevance factor $R(\tau)$ of the state τ for each node:

$$R(\tau) = \max_{\text{paths}} \left(\prod_{\text{path}} \left(\frac{\lambda_{\sigma\to\tau}}{\sum_k \lambda_{\sigma\to k}} \right) \right) \times R(\iota)$$

where $R(\iota)$ is the relevance factor of the initial node ι, $R(\iota) \overset{def}{=} 1.0$.

The chosen relevance factor can be analyzed as the factor which favors the most influential parents through the most influential paths. This is in accordance with the interpretations of the Markov chains for safety systems (see Section 2.2). This formula leads to the selection of most of the most probable states of the Markov chain. Correlations between the value of the relevance factor of the nodes and the probability of the corresponding states are done in the experiments (e.g. for the oil production system in Section 3.2.3).

2.3.3. From Guarded Transition Systems to Markov chains and Safety Indicators

The exploration and construction of the final Markov chain is done from a model specified in a higher level modeling language, namely the Guarded Transition Systems (GTS), as explained Section 1.1. It is done in three steps:

- First, the Reachability Graph (RG) $\Gamma = \langle \Sigma, \Theta \rangle$ of the GTS is partially computed using Dijkstra's algorithm and the relevance factor defined in Equation 2.10, where Σ is the set of nodes (i.e. states of the system) and Θ is the set of edges between nodes (i.e. the transitions between states). The reachability graph is the minimal Kripke structure representing the behavior of the GTS.
- Second, the RG is translated into a Markov chain.
- Finally, the performance indicators are calculated from the Markov chain.

Partial Construction of the Reachability Graph

Let $\langle V, E, T, A, \iota \rangle$ be a GTS and $\Gamma = \langle \Sigma, \Theta \rangle$ be its RG. A transition rate is associated with each transition of Θ using the corresponding event $e \in E$ of the GTS.

Algorithm 2.2 shows the partial generation of the reachability graph. The initial node ι is the initial assignment of variables of the given GTS $\langle V, E, T, A, \iota \rangle$. The GTS model is animated to calculate the fireable transitions from ι. For each of these transitions $(e : G \to P)$, where e is the event associated to the transition, G is the guard of the transition, and P is its action, the reachable nodes are calculated. This process is repeated with the candidate that as the highest relevance factor, until the end of the exploration. The end of the exploration is either when the candidate set C is empty (i.e. there are no more nodes to explore) or when a threshold $\mathbb{S}$ on the size of the model $|\Gamma_S|$ is reached.

The usual thresholds are: the number of states $|\Sigma|$, the number of transitions $|\Theta|$, or the size of the data structure holding Γ. As the worst-case time of the calculation of the transient solution of the Markov chain is in $\mathcal{O}(|\Theta|)$, and because of the chosen implementation (see Section 4.2.4), the threshold on the number of transitions is used. This threshold is denoted ξ.

The complexity of the algorithm is detailed in Section 4.2. Our data structures achieve worst-case $\mathcal{O}(|\Theta| \times \log_2 |\Sigma|)$ time and $\mathcal{O}(|\Theta| + |\Sigma| \times |V|)$ memory, where V is the set of variables of the GTS. The relevance factor is calculated in line (9) of Algorithm 2.2. According to its definition in Equation (2.10), $R(\tau) = R(\sigma) \times \left(\frac{\lambda_{\sigma\to\tau}}{\sum_k \lambda_{\sigma\to k}}\right)$, the sum $\sum_k \lambda_{\sigma\to k}$ should be recalculated for each transition, which would alter the time complexity of the algorithm. In practice, this is avoided (see Section 4.2.1 for more details on the implementation).

Algorithm 2.3 shows what happens after the exploration is finished: the remaining candidates are discarded, and transitions from states $\sigma \in \Sigma$ to states $\tau \in C$ are deleted. The partial reachability graph is (Σ, Θ).

Algorithm 2.2: Algorithm for the construction of a partial reachability graph of size at most $\mathbb{S}$.

```
   Input: A GTS $\langle V, E, T, A, \iota \rangle$
   Input: A function $l(e : \sigma \to \tau)$ that calculates the length of a transition
   Input: A threshold $\mathbb{S}$ on the size of the reachability graph
   Output: $\Gamma_S = (\Sigma, \Theta)$ the partial reachability graph
   Local: $C$ the set of candidate states
   Local: $R(\sigma)$ the relevance factor of each state $\sigma$ from the initial state $\iota$
1  begin
       // Initialization
2      $C \leftarrow \{\iota\}$, $\Sigma \leftarrow \emptyset$, $\Theta \leftarrow \emptyset$, $R(\iota) \leftarrow 1.0$
       // Construction of the state space
3      while $C \neq \emptyset$ and $|\Gamma_S| \leq \mathbb{S}$ do
           // Selection of the best candidate
4          let $\sigma$ be the candidate with the minimum value $R(\sigma)$
5          $C \leftarrow C \setminus \{\sigma\}$
6          $\Sigma \leftarrow \Sigma \cup \{\sigma\}$
           // Calculation of its successors
7          foreach fireable transition $(e : G \to P)$ of $T$ do
8              let $\tau = A(P(\sigma))$
9              let $R = R(\sigma) \times \left(\frac{\lambda_{\sigma \to \tau}}{\sum_k \lambda_{\sigma \to k}}\right)$
10             if $\tau \in C$ and $R(\tau) > d$ then
11                 $R(\tau) \leftarrow R$
12             else if $\tau \notin C$ and $\tau \notin \Sigma$ then
13                 $R(\tau) \leftarrow R$
14                 $C \leftarrow C \cup \{\tau\}$
15             $\Theta \leftarrow \Theta \cup \{(e : \sigma \to \tau)\}$
```

Algorithm 2.3: Algorithm for the construction of a partial reachability graph of size at most $\mathbb{S}$, where candidates are discarded.

```
1  begin
       [contents of Algorithm 2.2]
       // Removal of discarded candidates
16     foreach transition $(e : \sigma \to \tau)$ in $\Theta$ such that $\tau \in C$ do
17         remove $(e : \sigma \to \tau)$ from $\Theta$
```

From the Reachability Graph to a Markov chain

This translation is straightforward. The nodes of Σ of the RG are directly translated into states of the Markov chain. The edges of Θ are translated into transitions. In practice, the edges are labeled with an event e in the GTS, and the transition rates are retrieved from the GTS model.

From the Markov Chain to the Safety Indicators

The values of the desired indicators (unavailability, production rate, ...) are associated to each state of the Markov chain. This follows the concept of rewarded Markov chain. The transient solution of the Markov chain is computed over time and the values of the desired indicators are computed by summing for each state their value multiplied by their probability:

$$r_i, \quad \mathcal{R}(t) = \sum_i r_i \times \pi_i(t)$$

The obtained quantity is the mean value at a given time of the safety indicators of interest.

2.4. Quality of the Exploration, Bounds on the Reliability Indicators

When the full generation is possible, the quality of the partial generation is easily analyzed. Experiments were done and results are presented in Chapter 3 (e.g. with the computing modules in Section 3.2.2).

However, when the full generation is not possible ($|\Theta| > 100$ million transitions for the complete chain), it is not possible to measure or predict the accuracy of the method. A measure can be introduced before the calculation of the Markov chain. It uses a special state, the sink state, which is added to the partial Markov chain. Its efficiency is discussed here, and is illustrated on examples in Chapter 3 (e.g. with the emergency power supply in Section 3.2.4).

2.4.1. The Sink

As discussed in Section 2.2, Markov chains can be interpreted as a graph of probability buckets. In Algorithm 2.3, when partially exploring the state space, the remaining candidates are simply discarded, as well as the transitions leading to them. This means that the pipes leading to these states are deleted, and the probability reflows in the partial chain.

Instead of cutting these transitions, they can be redirected to a single state. This special state is, by definition, an absorbing state, and is called the sink. This process is illustrated in Figure 2.7. When the exploration is stopped, the sink ω is added to Σ and transitions leading

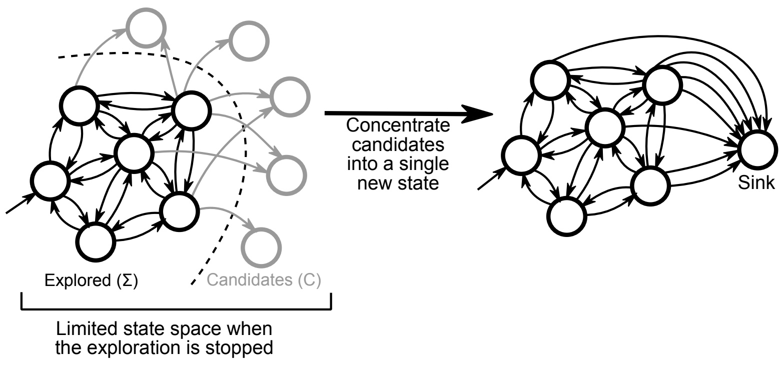


Figure 2.7.: The sink is the aggregation of the remaining candidates.

Algorithm 2.4: Algorithm for the construction of a partial reachability graph of size at most $\mathbb{S}$, where candidates are gathered into the sink.

```
1  begin
       [contents of Algorithm 2.2]
       // Redirections of the transitions toward the sink
16     create a sink state $\omega$ and add it to $\Sigma$ if $C$ is not empty
17     foreach transition $(e : \sigma \to \tau)$ in $\Theta$ such that $\tau \in C$ do
18         remove $(e : \sigma \to \tau)$ from $\Theta$
19         add $(e : \sigma \to \omega)$ to $\Theta$
```

to the remaining candidates are redirected to ω. Algorithm 2.4, which provides another ending to Algorithm 2.2, is used to obtain the Markov chain with sink.

Formally, the sink is the aggregation of all the remaining candidates at the end of the exploration. Consequently, it accumulates all the states that could not be reached, and all the probability that is not taken into account because of the partial exploration. Because of the sink, the calculated probability of being in any state of the explored chain is lower than the probability of being in the same state in the complete chain. Consequently, the partial exploration with a sink provides a conservative Markov chain representing the system.

Moreover, the probability to be in the sink ω is the absolute error made because of the partial exploration. The probability to be in the sink obviously depends on the mission time and on the system under study.

2.4.2. Using the Sink

In practice, the absolute error $\epsilon(t)$, which is the probability to be in the sink over time $P_\omega(t)$, must be compared to the indicator of interest $Q(t)$. Usual indicators are the unavailability, the unreliability, or the production rate of the system. Let Q_{min} be the absolute minimum value of Q in the unexplored part of the state space (respectively Q_{max} the maximum value). These quantities are expert knowledge on the system. For instance, when Q is the unavailability of the system, Q_{min} is 0 and Q_{max} is 1. Bounds on the measured indicators are defined as follows:

$$Q_{lb}(t) = Q(t) + Q_{min} \times \epsilon(t)$$
$$Q_{ub}(t) = Q(t) + Q_{max} \times \epsilon(t)$$

These bounds are useful in the following situations:

- When $Q(t)$ must be lower (or higher) than a specified value: $\forall t \quad Q_{ub}(t) < Q_{\text{target}}$.
- When the confidence in the value of $Q(t)$ must be known.

The relative error $\eta(t)$ is defined as follows:

$$\eta(t) = \frac{Q_{ub}(t) - Q_{lb}(t)}{Q_{lb}(t)} = \epsilon(t) \times \frac{Q_{max} - Q_{min}}{Q_{lb}(t)} \tag{2.11}$$

Choosing Q_{lb} as the divisor in the previous formula makes η easier to use. When $\eta(t)$ is lower than 1, the bounds have the same order of magnitude, and $(-\log(\eta))$ gives roughly the number of expected correct significant figures. When $\eta(t)$ is higher than 1, the bounds are not close enough, and the approximate value for Q could vary from more than one order of magnitude if the full chain was calculated.

It is important to remark that, as the sink state does not have any outgoing transition:

$$\epsilon(t) \xrightarrow[t \to +\infty]{} 1$$

which means that the partial generation with sink *will* become irrelevant if the chosen mission time is too long.

The efficiency of the sink and the value of the relative error depends on the mission time and on the system that is modeled. In Chapter 3, the sink and the relative error are analyzed on several examples (e.g. the oil production system in Section 3.2.3).

2.4.3. Towards a Better Sink

The proposed sink can be more formally proven by seeing it as an aggregate using arguments taken from [Courtois 1977] [Muntz et al. 1989]. It is tempting to modify and improve the sink.

A better sink would give a less crude approximation of the bounds on the error while not being computationally expensive to compute.

The first idea is to provide some outgoing transitions from the sink. However, which states would be the output of such transitions, and what would be the transition rates? To be efficient, the calculation of the back transitions should not take more than $\mathcal{O}(|\Theta|)$ time (see Section 2.3.1 and 4.2.1). Otherwise, it would be simpler to reduce the error by exploring further the state space.

Aggregation techniques, as seen in Section 1.2.3, are not good candidates. It has been shown in [Muntz et al. 1989] that an aggregation would be possible. However, the strategy to calculate the backward transitions from the aggregates to the detailed part of the chain is not well proved nor efficient.

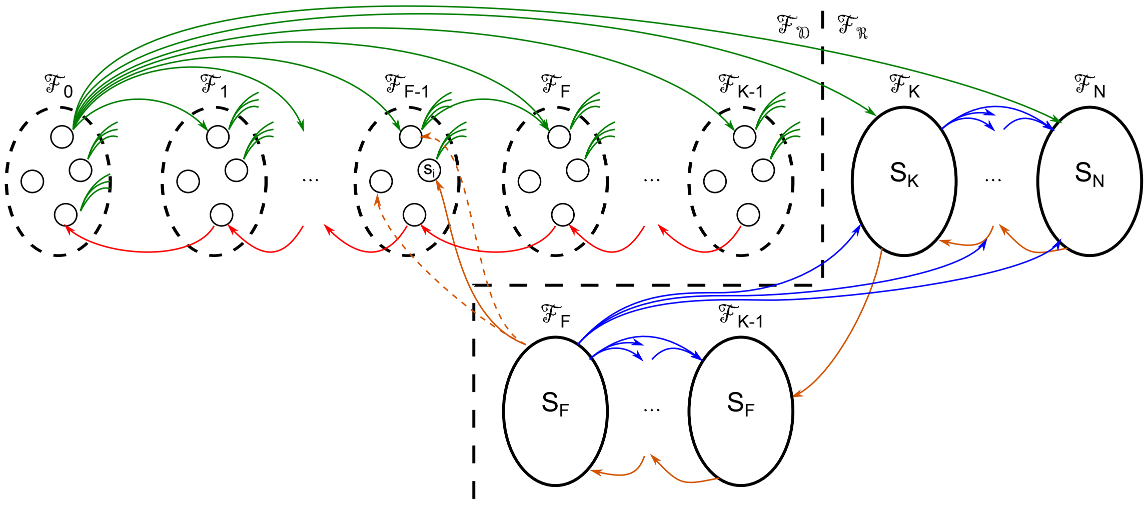


Figure 2.8.: Aggregation as seen in [Muntz et al. 1989], with duplication of part of the state space.

Figure 2.8 illustrates the aggregation as seen in [Muntz et al. 1989]. It separates the chain in two sets: $\mathcal{F}_\mathcal{D}$ and $\mathcal{F}_\mathcal{R}$. The detailed part of the chain $\mathcal{F}_\mathcal{D}$ is very similar to the one proposed in this thesis. $\mathcal{F}_\mathcal{R}$ is the reduced part, where states are aggregated according to the number of failed components. The aggregated part $\mathcal{F}_\mathcal{R}$ differs from the sink in that it is split in different aggregates instead of only one. An aggregate state S_J regroups the states of $\mathcal{F}_J$. The aggregation is done for states that have more than K failed component (K is arbitrary). The transitions from the detailed part to the aggregates are the transitions that would have gone to the state which has been aggregated. This is similar. However, transitions between the aggregates and returning transitions from the aggregates to the detailed part are a problem.

To bound the indicator of interest, the potential transitions going from the closer aggregate $\mathcal{F}_K$ to the detailed part are all tested one by one. This leads to the upper and lower bounds of the indicator of interest. This is not computationally efficient, as it is not possible to predict the number of potential transitions going from the aggregates to the detailed part of the chain. To reduce the number of returning transitions, part of the state space is duplicated and aggregated (states $\mathcal{F}_F$ to $\mathcal{F}_{K-1}$). Of course, lower F give shorter run-times but worse approximations. Still, the number of returning transition is not bound, neither is the complexity of the method.

To improve our partial exploration method, one would expect that the maximum error could be set *before* the exploration is done. This would guarantee the error *a priori*. In the next subsection, it is shown that an approximation with predictable error cannot be obtained within reasonable time. Better bounds can be obtained by going further in the exploration.

There is no predictable error

Using Algorithm 2.4 with the sink, the absolute error $\epsilon(t)$ can be measured at any time, after the partial Markov chain has been calculated. If the error goes above a target error ϵ_{max}, the whole exploration must be done again with a larger threshold $\mathbb{S}$, starting from the GTS model. This process is iterative and finishes either when $\forall t \in [0..T], \epsilon(t) < \epsilon_{max}$ or when the computer cannot calculate the model anymore. There is no mean to predict the optimal value of $\mathbb{S}$.

It would be desirable to obtain a predictable degree of approximation that will be induced by the method. In other words, it would be desirable to obtain a mean to calculate a threshold $\mathbb{S}$ on the size of the RG $|\Gamma_S|$ such that the absolute error is below a target ϵ_{max}. Moreover, it would be interesting for the calculation of this threshold to be less expensive and more predictable. In other words, obtaining $\mathbb{S}$ should be done in polynomial time with respect to the size of the input GTS model. Here is shown a counterexample underlining that it is not possible.

Let FT be a Fault Tree whose top event is to be calculated and approximated. On the one hand, the top event of FT can be reduced to a Boolean expression over the variables of the model. Calculating the probability of the top event is a Boolean satisfiability problem (SAT-problem), which worst-case time is not polynomial with respect to the number of basic events of the tree, and is known to be #P-hard (see [Valiant 1979]). Even the approximation of a #P-hard problem is hard (see [Papadimitriou 1994]). This means that no bounds of the probability of the top event can be obtained in polynomial time.

On the other hand, a fault tree can be easily encoded as a GTS:

- Each basic event is encoded by means of a Boolean state variable and a transition that changes the state variable from false to true,
- Each gate is encoded by a flow variable which is updated in the assertion according to the value of its fan-ins.

If it was possible to predict the threshold $\mathbb{S}$ in polynomial time over the variables of the GTS (which are the basic events of FT), such that the absolute error is below the target ϵ_{max}, it would be possible to obtain a predictable degree of approximation on the probability of the top event of the fault tree in polynomial time. □

There is only one way to reduce the absolute error: going further in the exploration by raising $\mathbb{S}$. The problem should be seen differently: $\mathbb{S}$ can be raised until the models cannot fit in memory anymore. So $\mathbb{S}$ should depend on the memory resources at hand. When trying to obtain an accurate $Q(t)$, calculation should be done with the highest possible $\mathbb{S}$, and $\eta(t)$ should be used to determine if the model was correctly approximated or not.

Summary

Algorithm 2.3 is used to generate a RG from a given GTS without sink. It gives, in practice, the best results, as shown in Chapter 3. However, it can only be used as a heuristic method, because it gives an approximation that cannot be measured.

Algorithm 2.4 is used to generate a RG from a given GTS with a sink. The sink is used to measure the absolute error, after the final Markov chain is calculated. Bounds from the absolute error can be obtained, but they may not be very close to each other. It is not possible to obtain better bounds while keeping all the calculations in polynomial time.

Probabilistic Safety Assessments (PSA) and Reliability Availability Maintainability and Safety assessments (RAMS) usually targets 10% error (see for instance [Bon and Collet 1994] [Collet and Renault 1997]). Actual PSA for nuclear power plants may be less accurate, as in [Dubreuil Chambardel et al. 1991], where the core damage risk for the French nuclear power plant of Paluel was evaluated in 1989 to 10^{-5} per reactor per year, with an error factor of 3.5 (this error factor means that the probability is between 3×10^{-6} and 3×10^{-5}). This raises the question of the sources of inaccuracy in the PSA.

Figure 2.9 analyzes the different tools, from the model to the indicators of interest. It is important to note that the input data are rarely known correctly for components that fail once in a million hours. The partial generation induces more error, but that error may be measured. Because the input data is so little known, it is important to keep in mind that the method consists in heuristics that can be used as decision aids. The partial generation method for Markov chain is similar to minimal cutsets with cutoffs for fault tree analyses.

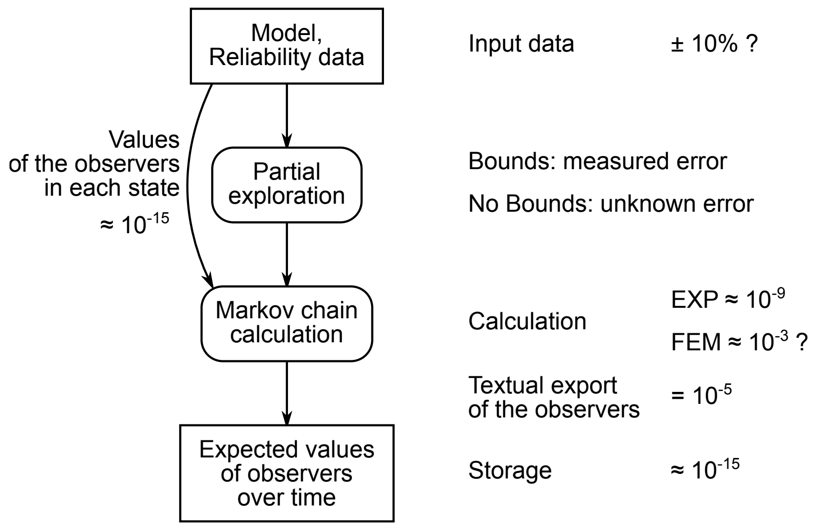


Figure 2.9.: Analysis of the accuracy loss alongside the toolchain.

3. Experiments

The intent of this chapter is to illustrate and analyze the partial generation method. The analysis is twofold. First, the presented examples illustrate the practicability of the method on some realistic examples. Second, the examples are used to analyze the partial generation method in several different cases.

Besides the presentation of each example, the following studies are performed:

- Numerical values for the performance indicators of interest are given for several sizes of the partial models, with or without sink. Such data are the basis of the studies, and shows the results obtained with the partial generation.
- The evolution of the relative error made on the quantity of interest, with respect to the number of transitions of the partial model, is shown.
- One or more of the following correlations are analyzed:
  - The correlation between the relevance factor calculated during the exploration and the actual probability calculated with the Markov chain, to compare the selection criterion to the reality of the chain.
  - The correlation between the probabilities calculated for each state, with the small partial model and with a larger model, to compare the behavior of the states whether they are in the small model or in the large model.
  - The correlation between the probabilities of the most probable states of the limited model and of the full model. This correlation shows how the limited model globally behave compared to the full model, and how many more probable states could have been missed.
  - The correlation between the probabilities of states in a partial chain without and with sink. The influence of the sink on the overall behavior of the chain is observed with this correlation.

  These correlations are given as plots, and support a visual analyze, but should not be quantified, as they only give a partial view on the method.

The tool-chain used to do these computations is detailed in Section 4.1. The algorithm used to compute the probabilities of states of the Markov chains is FEM, with the accuracy parameter ρ set to 0.25 (see Section 2.1.4).

The main strength of the method is to provide an efficient and accurate approximation for the performance indicators of a system. However, the accuracy of the partial generation heavily depends on the system.

The most influential characteristic of a system which determines the accuracy of the partial generation is the concentration of the probability near the nominal state of the system. There are experimental conditions that may degrade the accuracy of the method:

- The bad choice of the nominal state.
- The time at which the relative error is to be evaluated (mission time).
- The proneness of a system to have its probability scattered through numerous states.
- The need for a bounded evaluation of the performance indicators. The relative error is calculated with the sink state ω, which drains the probability from the rest of the chain.

The partial method generates a chain without sink that can be used as decision aids. It is used to model safe systems, from which high productivity or high availability is expected. Such systems are expected to be most of the time in their nominal state.

The remainder of this chapter is organized as follows. First, we discuss how to choose an example. Second, the examples are presented and analyzed. Finally, the examples are taken as a whole to provide feedback on the partial generation method.

3.1. Targeting the Examples

The objective of this section is to target which examples are useful to *analyze* the method. The idea is to determine what a useful example would be, by choosing some criteria.

The criteria are summarized as follows:

- **Balanced examples.** The example should not have obvious state space reductions, such as symmetries. Design flaws should be avoided, such as weak component choices, weak redundancy strategies, wide common cause failures, ...
- **Modeling features from high level language.** The examples should show the use of interesting modeling features. The solving method should be adapted to the problem. For instance, it would be useless to use Markov chains to model a non-repairable system made of binary and independent components, as a fault tree would give results more efficiently. Such interesting features are: warm redundancies, repairs, common cause failures, failures on demand, limited number of repairers, modularity, ...
- **Realism.** The examples should not be *artificial.* The idea is too choose real systems that could be subject to real safety engineering problems. The performances of the method are illustrated on actual use-case. The goal is to avoid tailor-made examples which would be successful with the partial generation.
- **Model size.** The examples must be large enough to illustrate the method in realistic conditions, and its scalability.

Some examples are taken from the literature, which inspires trustworthiness in the tests. It enforces the consistency of our results when the results are the same with both methods, and may enable further comparison.

A single example cannot meet all the criteria. For instance, it is hard to find an example which is small enough so that the accuracy of the method can be shown (with help of the full Markov chain), but large enough so that the self estimated accuracy with the bounds is possible and realistic.

Table 3.1 shows the different examples that were chosen and their main characteristics. The computing modules is the only non-repairable system. The “nominal state” column tells whether the system has a state in which the system is most of the time, i.e. a state which probability is very close to 1. Non-repairable systems do not have a nominal state.

The oil production system is used to study both the unavailability and the production rate. This example is singular in the sense that the production rate is an indicator which has a very high order of magnitude compared to the unavailability of the other systems. Both the emergency power supply and distributed database are very large systems.

Table 3.1.: The different examples and their characteristics.

Experiment	Repairable	Complete chain	Computed indicator Name	Computed indicator Expected value	Nominal state
E1: Fault Tolerant Database	✓	✓	Unavailability	10^{-3}	✓
E2: Computing Modules		✓	Unreliability	1	
E3: Oil Production System	✓	✓	Production rate	200	✓
			Unavailability	10^{-1}	
E4: Emergency Power Supply	✓		Unavailability	10^{-7}	✓
E5: Distributed Database	✓		Unavailability	10^{-5}	✓

3.2. Studied Systems

The different examples are developed and studied in this section. The method is analyzed based on these individual examples. Smaller examples are introduced first.

Section 3.2.1 shows a fault tolerant database. Section 3.2.2 shows two redundant computing modules. Section 3.2.3 shows an oil production system, which production rate is calculated. Section 3.2.4 shows an emergency power supply. Section 3.2.5 shows a redundant database system which is distributed over several clusters of hard drive disks.

3.2.1. Fault Tolerant Database

The fault tolerant database system models a database which can be accessed by a user through two redundant paths. It is taken from [Muntz et al. 1989] and was inspired by [Goyal et al. 1987]. It was used by Muntz et al. to illustrate the creation and aggregation of a Markov chain from a higher-level modeling language. Numerical parameters are taken from [Muntz et al. 1989] for comparison purposes.

The state space of the system has few states and few transitions, can be fully generated, and used as a reference.

Description of the System

The fault tolerant database is depicted in Figure 3.1. The role of the system is to provide an access to the database DB from a Front End. The system is composed of two processing subsystems. Each processing subsystem has switch a SWx, a memory Mx and two processors PxA and PxB. Components may fail and are repaired according to the rates given in Table 3.2. The system is available when the front end and the database are working, and when there is at least one working processing subsystem. A subsystem is working when one of the processors is working, as well as the memory and the switch. If a processor fails it may contaminate the database with probability 0.01.

Repairs are prioritized. If all components had failed, the front-end and the database would be repaired first, followed by the switches and memory units, followed by the processors. To simplify the model, multiple components with the same priority are being repaired simultaneously.

Components cannot fail when the system is down.

The mission time was chosen to 100 hours. This might appear a very short mission time, but, as explained in Section 2.2.2, the time constant for this system is roughly $\frac{1}{\mu} \approx 1$ hour. At mission time, the system reached its steady-state.

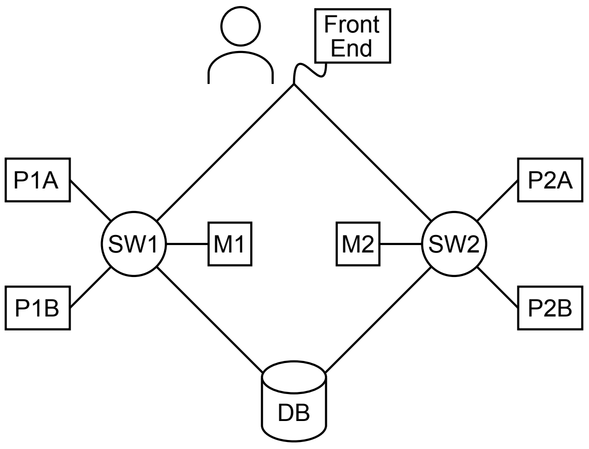


Figure 3.1.: The fault tolerant database.

Table 3.2.: Failure and repair rates for the fault-tolerant database, taken from [Muntz et al. 1989].

Component	Failure rates	Repair rates
Database	1/2400 h^{-1}	1 h^{-1}
Front End	1/2400 h^{-1}	1 h^{-1}
Switches	1/2400 h^{-1}	1 h^{-1}
Memories	1/2400 h^{-1}	1 h^{-1}
Processors	1/120 h^{-1}	1 h^{-1}

This rather small system is interesting as it involves several modeling concepts: the common cause failure, the deactivation of the components once the system is down, and the prioritized repairs. Moreover, values of the unavailability can be compared between the two approaches.

Results and Analysis

The complete Markov chain for this system has 419 states and $1,449$ transitions. In [Muntz et al. 1989], the Markov chain that was obtained for this model has 226 states. The steady-state unavailability was calculated to 0.0011647 with such Markov chain. The difference in the number of states probably comes from the different input modeling languages, which handles the repairs differently.

The results obtained with the partial generation method are shown in Table 3.3. Table 3.4 gives the size of the partial Markov chains and the calculation times. For this system, the Markov chain without sink is accurate, even with very few states, as $\xi = 250$ (84 transitions and 31 states) is sufficient to obtain less than 2% error ($\eta(t)$ is calculated with respect to the value of the unavailability obtained with the complete Markov chain). The Markov chain with sink needs more states to be accurate, but gives bounds on the unavailability. The Markov chain with sink with $\xi = 500$ (506 transitions and 82 states) gives satisfactory results.

In the case where $\xi = 100$ transitions, the large difference between the bounds for this particular example can be explained by the "long" mission time compared to the time constant of the system. The time constant of the system is about 1 hour, which means that the system reaches its steady-state in less than 5 hours.

Table 3.3.: Measures of the unavailability Q of the fault tolerant database for several thresholds on the size of the model, with given mission time of $T = 100$h.

	Partial chain without sink		Partial chain with sink		
Threshold ξ	Q	$\eta(T)$[a]	Q_{lb}	Q_{ub}	$\eta(T)$
$1,449$	0.0011638	-	-	-	-
$1,000$	0.0011638	8.5925×10^{-6}	0.0011638	0.0011642	0.00034001
500	0.0011629	0.00079051	0.0011624	0.001552	0.33512
250	0.0011455	0.015707	0.0011367	0.0089996	6.9174
100	0.00080476	0.30851	0.00071357	0.1159	161.42

[a] without sink, Q for the highest threshold is used as reference.

The unavailability obtained with the complete Markov chain (1.1638×10^{-3}) can be compared to the unavailability obtained in [Muntz et al. 1989] (1.1647×10^{-3}). The two numbers are very close (0.077%). The difference can be explained by several factors, but the most probable source of difference is the number of significant figures taken for the failure rates.

Table 3.4.: Sizes of the models and computation times for several thresholds on the size of the model of the fault tolerant database, with given mission time of $T = 100$kh.

		Partial chain without sink			Partial chain with sink	
Threshold ξ	Fraction	States	Transitions	Times[a] (s)	Transitions	Times[a] (s)
$1,449$	-	419	$1,449$	< 1	-	-
$1,000$	2/3	228	794	< 1	$1,059$	< 1
500	1/3	81	248	< 1	506	< 1
250	1/6	31	84	< 1	298	< 1
100	1/14	11	20	< 1	124	< 1

[a] (generation of the chain) + (calculation of the chain).

With $\frac{1}{2400} \approx 4.17 \times 10^{-3}$, the unavailability is calculated to 1.1647×10^{-3}. This highlights the danger of giving two many significant figures in the results.

In this thesis, we chose to display only 5 significant figures, which is probably already too much (see Figure 2.9). However, numbers of states and transitions are written with what appears more than 5 significant figures, because these numbers are integers. As integers are accurate up to 1, all digits are shown.

The calculation of each model took less than 1 second. As computation times may be very inaccurate due to the experimental conditions (see Section 4.3.5), the computation times can be displayed with 2 significant figures. Times that are below 10 seconds cannot be considered accurate.

As the relevance factor is used to select the states, it is interesting to compare its value to the actual probability of the states.

Figure 3.2 shows the complete Markov chain of the system, with the 419 states and $1,449$ transitions. On the left, the Markov chain is colored according to the relevance factor calculated by Algorithm 2.2. On the right, the Markov chain is colored according to the probability calculated with the Markov chain at mission time $T = 100$ hours. Color scales are logarithmic. Darker states have the highest relevance factor (or the highest probability), and whiter states have the lowest. The whiter states are the least reachable, according to the relevance factor. In these Markov chains, failure states are represented by diamond and working states by circles. For practical reasons, transitions representing the failure and repair of the same component are grouped together and are represented by a single non oriented line. Transitions that represent a failure are plain arrow, whereas transitions that represent a repair are dashed arrows.

It is interesting to see that the two chains look alike, which means that the relevance factor is able to capture the overall behavior of the Markov chain.

Consider the chain on the right. As the scale is logarithmic, the blue states are already

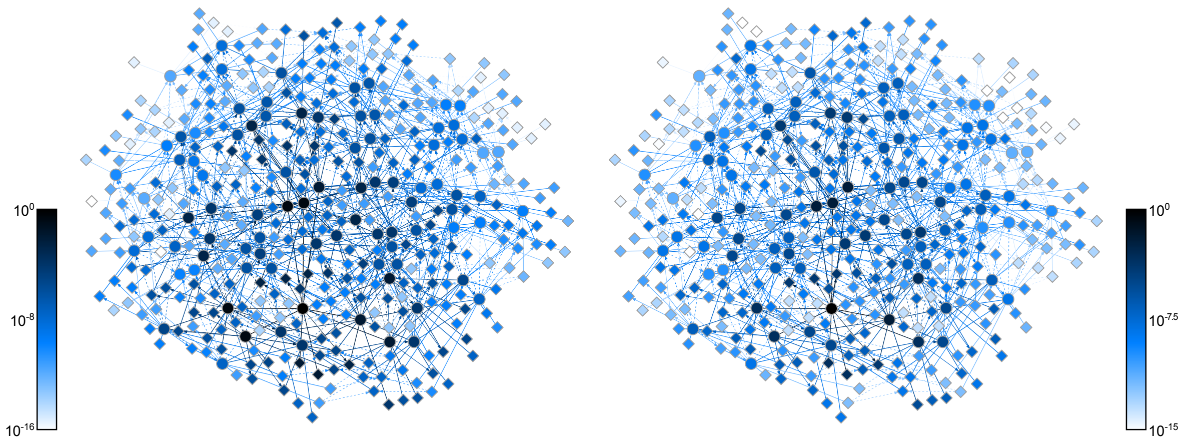


Figure 3.2.: Visual comparison of the relevance factor (left) and the probability at mission time (right), for the 419 states of the fault tolerant database.

improbable, as their probability is around 10^{-7}. The view of the chain highlights that the probability is concentrated only in a few states for a repairable system, as there are few dark states.

The chain on the left shows the same few dark states, and the same repartition of the colors, even if the scales are not exactly the same. The view of the relevance factor for this system shows that the relevance factor may be used to select the most probable states of the system.

In Figure 3.3, the numerical correlation between the relevance factor and the probability in the complete chain is shown. Each mark corresponds to a state of the Markov chain. The x-coordinate of the mark is the relevance factor and the y-coordinate is the probability. A linear fit is plotted in the logarithmic space to show that the probability is not exactly related to the relevance factor, but the overall correlation is good. This shows that the relevance factor is a good indicator of the probability of the states in the complete Markov chain, for the fault tolerant database system.

Figure 3.4 shows the effects of the partial generation without sink on the probability of the states of the Markov chain. On this plot, each mark is a state of the partial Markov chain constructed up to $\xi = 1,000$ transitions. The x-coordinate of the marks is the probability of the state in the partial chain, while the y-coordinate is the probability of the same state in the complete Markov chain. This shows that the probability of most states is unchanged when being part of the smaller Markov chain. Hence, their local behavior is not altered. The relative error on the unavailability calculated with this chain is roughly 10^{-6}.

However, some states have their probability lowered, and some other states have their probability raised. These states are on the border, because their probability is very low (around 10^{-10}), so there are only reachable through a long sequence of failures. This shows that the selection of some states have border effects which leads to unpredictable variations of the probabilities of the states.

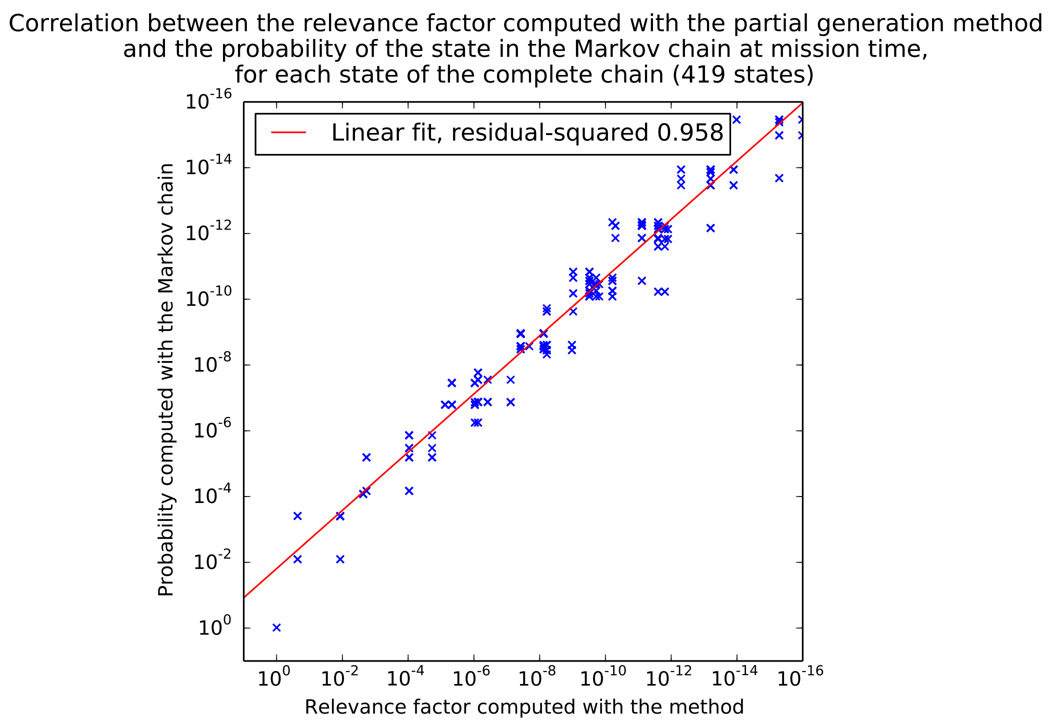


Figure 3.3.: Correlation between the relevance factor computed during the generation of the chain and the probability calculated with the Markov chain for each state of the complete chain for the fault tolerant database, with mission time $T = 100$ hours.

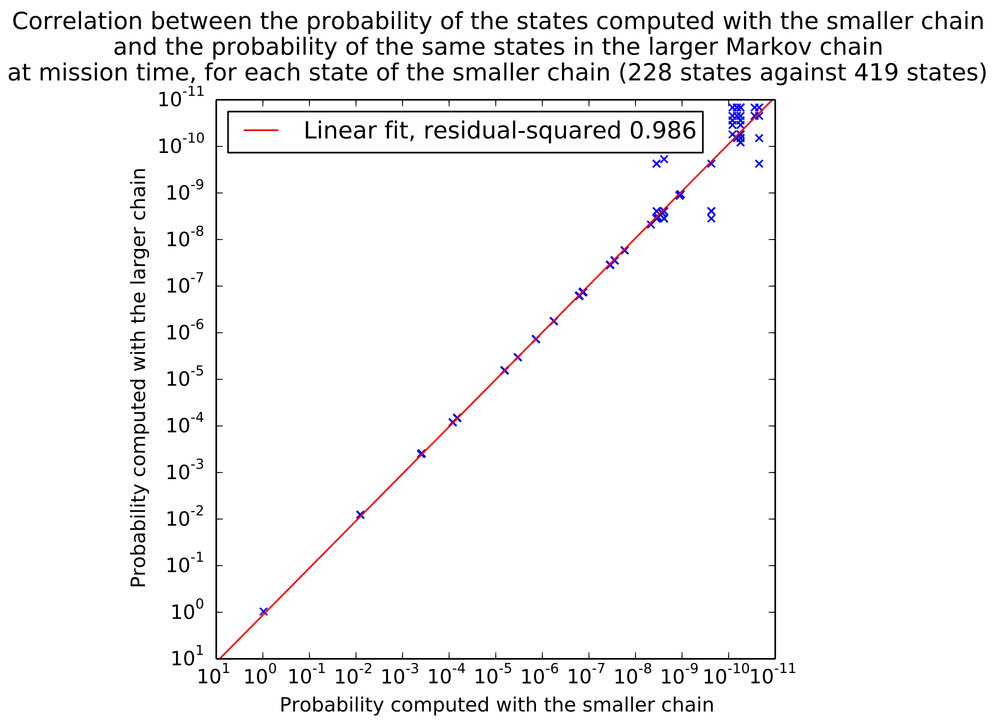


Figure 3.4.: Correlation between the probability of each state of the partial model (threshold $\xi = 1,000$ transitions) whether they are in the partial chain or in the full chain, for the fault tolerant database, with mission time $T = 100$ hours.

3.2.2. Computing Modules

The computing modules is a non-repairable redundant computing architecture. This example comes from Malhorta et al. [Malhotra and Trivedi 1995]. It was used in [Montani et al. 2006] to compare three safety tools which assess the unreliability of a system, based on different approaches: DBNet [Montani et al. 2006], DRPFTproc [Bobbio and Raiteri 2004] and Galileo [Dugan et al. 2000]. The safety model, technical data, and mission times we use here were described in [Montani et al. 2006]. This example was also shown in [Brameret et al. 2013, 2015] to illustrate the efficiency of the partial Markov chain generation without sink.

The system is small enough to obtain the complete Markov chain, which can be used as a reference.

Description of the System

The system pictured Figure 3.5 is a non-repairable multiprocessor computing system made of two computing modules CMx. Each module consists in a process Px, a memory Mx, a primary hard drive Dx1, and a secondary hard drive Dx2. A spare memory M3 can be used as a replacement of M1 or M2, but it cannot replace both. A bus connects the modules and the spare memory. The power supply PS is used by both processors.

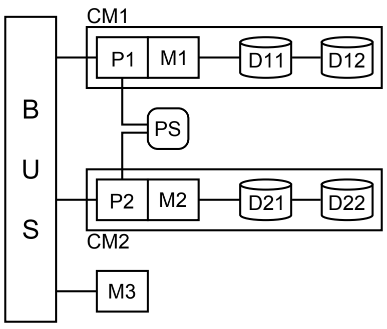


Figure 3.5.: A multiprocessor computing system.

The system is available if one of the two modules is available. Each module is available if the processor, a memory and one of the disks are available. If both modules are available, both modules do the requested computations.

Table 3.5 gives reliability data of the components. The disks and the memory are warm spares, which deteriorate at a slower rate when unused. Such "dormant" failure rate is calculated by multiplying the failure rate by the dormancy factor. The components that have no dormancy factor are components which are never dormant. The power supply provides obviously a cut set of length 1, which drains the availability of the whole system, but its reliability data are kept as is for comparison purposes.

This system is interesting as it is *non-repairable*, unbalanced by the power supply which is frequently failed, and it provides a shared spare memory, and warm spare components. The

Table 3.5.: Failures Rates and Dormancy Factors for the Computing System

Component	Failure rates (h^{-1})	Dormancy factors
BUS	2.0×10^{-9}	-
P1, P2	5.0×10^{-7}	-
PS	6.0×10^{-6}	-
Disks	8.0×10^{-5}	0.5
M1, M2, M3	3.0×10^{-8}	0.5

values of the unreliability can be compared with several methods. The high failure rate of the power supply should ease the partial generation, as it drives the overall failure of the system.

Results and Analysis

The Markov chain of this rather small system can be completely generated and calculated. It is composed of $17,152$ transitions and $3,328$ states.

Table 3.6 shows the transient unreliability of the system for several mission times up to $5,000$ hours, calculated with different methods. The results are presented as they are presented in [Montani et al. 2006], with 5 to 6 significant figures. The numbers are identical up to 6 figures for three methods, which inspires confidence in the complete generation method.

Table 3.6.: Unreliability of the computing system, assessed with different tools.

Time (hours)	DBNet	DRPFTproc	Galileo	Complete Markov chain
1,000	0.0060086	0.0060088	0.0060088	0.00600877
2,000	0.0122452	0.0122455	0.0122455	0.0122456
3,000	0.0191820	0.0191832	0.0191832	0.0191833
4,000	0.0273523	0.0273548	0.0273548	0.0273548
5,000	0.0372379	0.0372413	0.0372413	0.0372413

Table 3.7 shows the calculation of the unreliability with mission time $T = 50,000$ hours, for different sizes of partial chains. At this time, the system is close to complete failure. For the chains without sink, the relative error is very low, even with small Markov chains. A chain which is fifty times smaller than the complete chain is large enough to approximate the unreliability of the system with 2 significant figures. The chains with sink give a worse approximation, but they do give bounds on the unreliability. Moreover, these bounds are acceptable even if the mission time is long and the system almost always broken at this time. For this particular system, these good results might be explained by the predominant failure of the power supply, and by the non-repairability of the system.

Table 3.7.: Measures of the unreliability Q of the computing modules for several thresholds on the size of the model, with given mission time of $T = 50$kh.

	Partial chain without sink		Partial chain with sink		
Threshold ξ	Q	$\eta(T)$[a]	Q_{lb}	Q_{ub}	$\eta(T)$
$17,152$	0.93497	-	-	-	-
$8,576$	0.93497	0	0.93497	0.93497	5.2271×10^{-9}
$3,430$	0.93497	0	0.93497	0.93497	7.9438×10^{-6}
$1,715$	0.93496	1.7113×10^{-5}	0.9347	0.93498	0.00029888
857	0.93458	0.00041819	0.93034	0.93513	0.0051461
343	0.92935	0.0060087	0.90059	0.93693	0.040347

[a] without sink, Q for the highest threshold is used as reference.

To compute the relative error η, the quantities that are compared are usually subtracted one from another, which may lead to the numerical problem of cancellation. Cancellation is the consequence of numerically subtracting two very close numbers, and is due to the finite accuracy of the numerical representation of numbers. Let a and b be two close numbers written up to 6 significant figures, $\frac{a-b}{b}$ gives either 0 when the numbers are equals or a number greater than 10^{-5}. When looking at Figure 2.9, which shows the different sources of approximation from the AltaRica model to the storage of the performance indicators, the results of the evaluation of the Markov chain are written in a textual format with 6 significant figures.

The relative error η calculated for the Markov chains without sink uses the subtraction of previously calculated unreliabilities. This explains why 0 is obtained for the relative errors of some partial chains without sink. However, η for the Markov chains with sink uses the probability to be in the sink (see Section 2.4.2, Equation (2.11)). This explains why relative error lower than 10^{-5} can be obtained.

Table 3.8 shows the sizes of the partial Markov chains without or with sink, as well as the computation times. The calculation times are below one second for this small system.

Figure 3.6 shows the correlation between the relevance factor of the states and their probability in the complete Markov chain, with mission time $T = 5,000$ hours. Each mark corresponds to a state of the Markov chain. The x-coordinate of the mark is the relevance factor and the y-coordinate is the probability at time T. According to the plot, some of the states that the relevance factor indicated as relevant are not probable. This is the case for states with relevance factor around 10^{-5} which have their probability from 10^{-5} to 10^{-18}.

There are at least two possible explanations for that, considering the fact that the system did not reach a steady state at $T = 5,000$. First, as the evolution of the system is not finished, states with low probability and high relevance factor may be states that are not probable *yet*, but that may become more probable. Second, as the system is not repairable, the relevance

Table 3.8.: Sizes of the models of the computing modules and computation times for several thresholds on the size of the model, with given mission time of $T = 50$kh.

		Partial chain without sink			Partial chain with sink	
Threshold ξ	Fraction	States	Transitions	Times[a] (s)	Transitions	Times[a] (s)
$17,152$	-	$3,328$	$17,152$	$1+1$	-	-
$8,576$	$1/2$	$1,425$	$5,819$	$1+0$	$8,589$	$1+0$
$3,430$	$1/5$	550	$1,858$	< 1	$3,437$	< 1
$1,715$	$1/10$	274	797	< 1	$1,719$	< 1
857	$1/20$	145	345	< 1	876	< 1
343	$1/50$	55	105	< 1	344	< 1

[a] (generation of the chain) + (calculation of the chain).

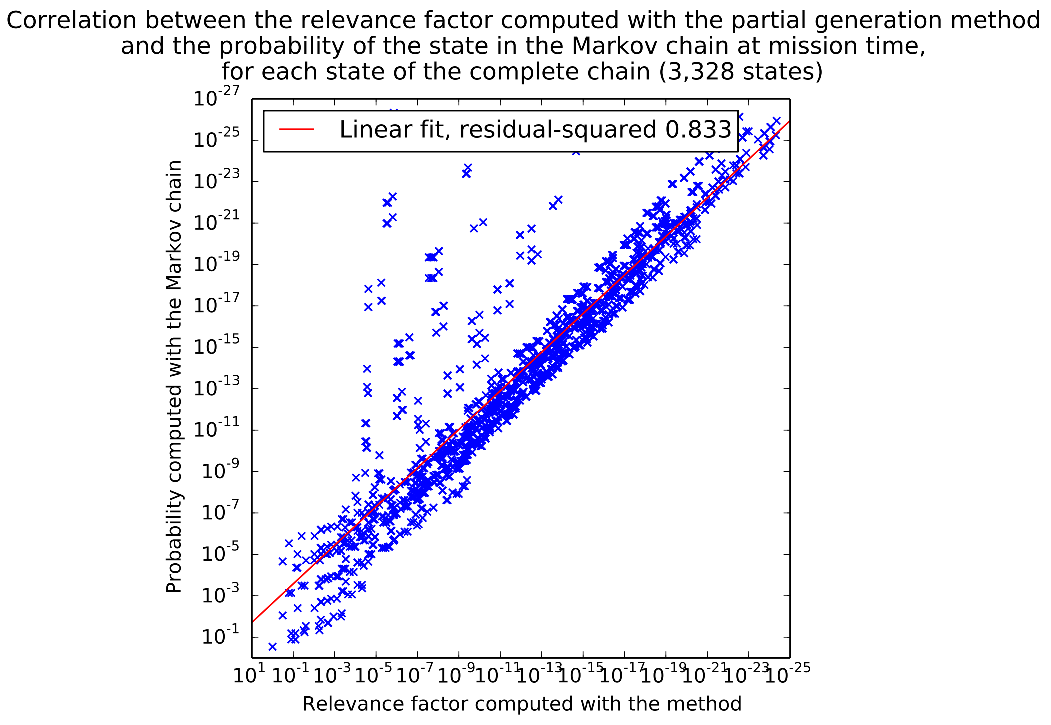


Figure 3.6.: Correlation between the relevance factor computed during the generation of the chain and the probability calculated with the Markov chain for each state of the full chain for the computing modules, with mission time $T = 5,000$ hours.

factor did not correctly take into account the behavior of those states.

Figure 3.7 shows the same correlation, but with a longer mission time $T = 50,000$ hours. With time, the probability of some states decreases, while the probability of some other states increases. This leads to the stretch of the plot. Some of the states which probability increased are most likely natural absorbing states of the system. If the time was infinite, all the probability would be in these natural absorbing states.

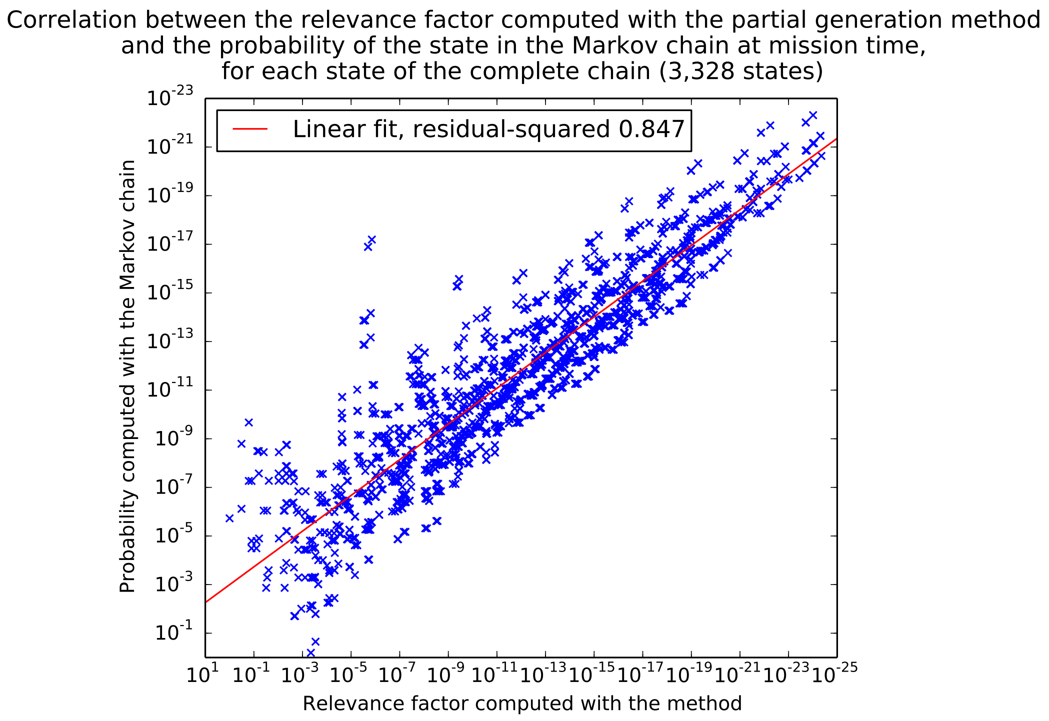


Figure 3.7.: Correlation between the relevance factor computed during the generation of the chain and the probability calculated with the Markov chain for each state of the full chain for the computing modules, with mission time $T = 50,000$ hours.

Figure 3.8 compares the probabilities of the states, whether they are part of the smaller or the complete chain. The small chain was generated with the threshold set to $\frac{1}{20}$ of the total number of transitions, $\xi = 857$. The chains are calculated with mission time $T = 5,000$ hours. Each mark on the plot is a state of the system that is part of the small chain. The x-coordinate is the probability that is computed with the small chain, and the y-coordinate is the probability computed with the complete chain for the same state. This shows that the small chain does not approximate the complete chain accurately, as the probability of the states in the complete chain cannot be deduced from the probability in the small chain. However, the relative error on the unreliability $\eta(5,000)$ is computed to $0,00142$ with the small chain, which is very low. This is the expected behavior for such a non-repairable system. The absorbing states in the larger chain are the states in which all the components of the system are failed.

On this particular example, which is not repairable, the partial generation method works

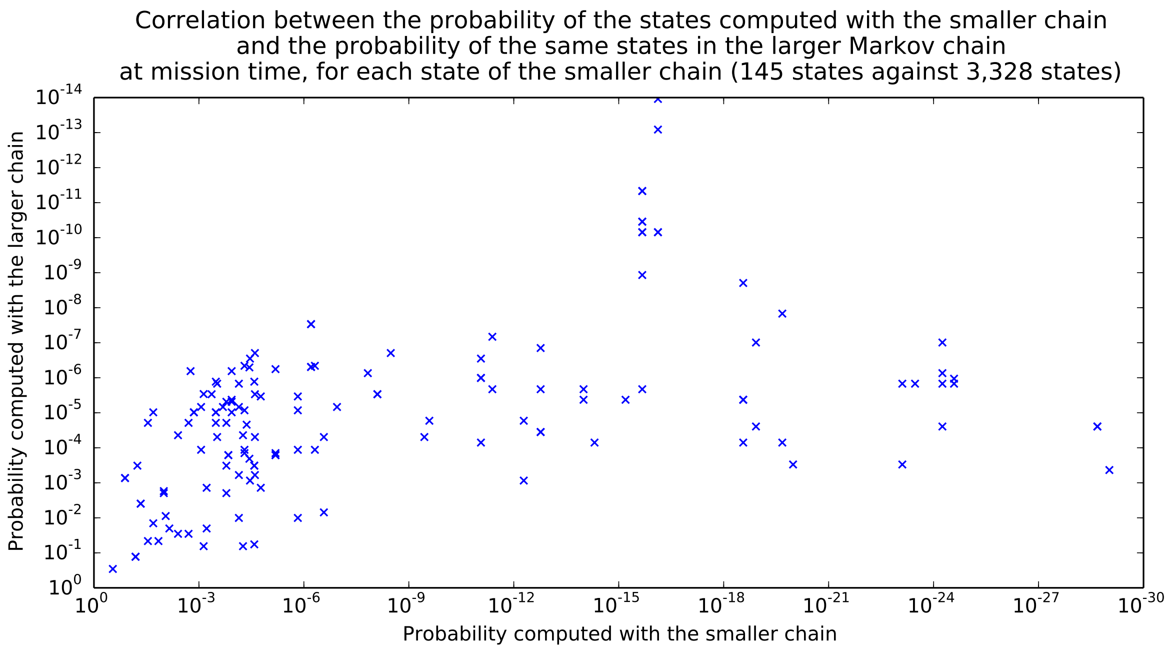


Figure 3.8.: Correlation between the probability of each state of the partial model (threshold $\xi = 857$ transitions) whether they are in the partial chain or in the full chain, for the computing modules, with mission time $T = 5,000$ hours.

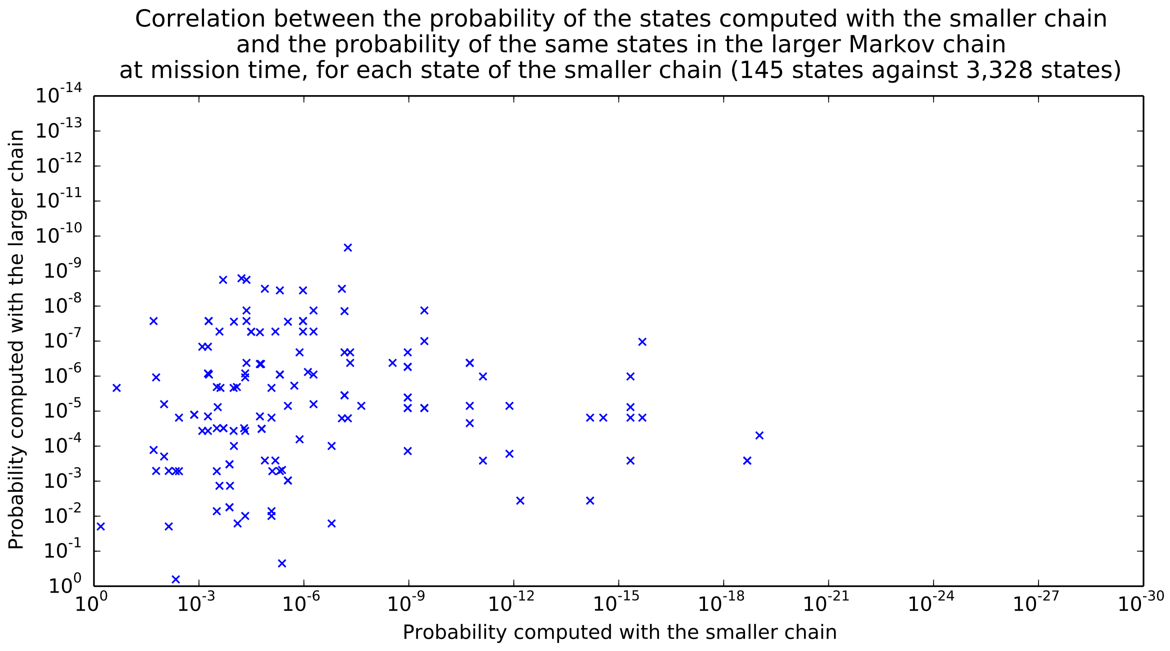


Figure 3.9.: Correlation between the probability of each state of the partial model (threshold $\xi = 857$ transitions) whether they are in the partial chain or in the full chain, for the computing modules, with mission time $T = 50,000$ hours.

well, but the probabilities of the individual states of the small chain are not related to the probabilities of the same states in the complete chain. However, the overall behavior of the system is well approximated, even if there is no nominal state in this system, i.e. even if there is no state which probability is close to 1. This is due to the high number of failure states in the complete chain, because they are successfully reduced to fewer failure states in the small chain. In other words, the partial generation reduced branches of states to a few states, which gives good results because all the states in the branches are failure states. The relevance factor proved efficient to analyze the depth of the branches and the competition between branches.

This result can be generalized to other non-repairable systems, and to other reliability or performance indicators. However, the indicator must be monotonous with respect to the failures of the system. For instance, when calculating the reliability of the system, there must not be a failure from a failure state leading to a working state (coherent systems).

Figure 3.9 shows the same comparison, but mission time is increased to $50,000$ hours. This plot shows that there are less extreme marks, but no correlation can be done between the small and the complete chain. According to Table 3.7, the small chain is very accurate ($\eta(50,000) \approx 0.00042$).

3.2.3. Oil Production System

The oil production system is a repairable system dedicated to the treatment of petroleum oil directly from wells and stores the output in tanks. The goal of the model is to calculate the production rate of the system.

Production systems are slightly different from safety systems. Typical production systems are repairable, with rather high failure and repair rates. Their components are frequently broken and rapidly repaired, but have a rather "low" availability.

The parameters of the production systems (architecture, maintenance policy, failure and repair rates, ...) are optimized with respect to some quantities, which are usually the costs and the productivity. Optimization methods, such as genetic algorithms, usually require a high number of tests. The partial generation method enables engineers to quickly calculate an approximate result for the productivity, so that multiple tests can be done. It is possible to calculate the performance indicators more accurately in the following parts of the design.

Description of the System

The system depicted in Figure 3.10 is a part of an oil extraction installation taken from [Rauzy 2004]. This system was designed to concentrate most of the modeling difficulties for the assessment of production availability. The system extracts oil from wells W1 and W2, refines it, and stores it finally in tanks T1 and T2. The system has the following components:

- W1 and W2 are the source wells. When they are failed, the production is stopped.

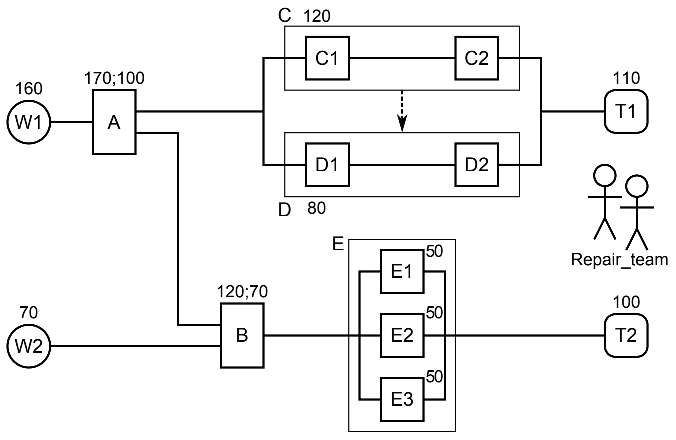


Figure 3.10.: The oil production system, with production rates of the components (in barrels per unit of time).

- T1 and T2 are the destination tanks. They cannot fail, but their flow rate limits the production of the whole system.
- A and B are treatment units. They have two failure modes: a *degraded* mode in which their production is reduced, and a *severe* failure mode in which their production is stopped. A severe failure can happen when the units are working correctly or when they are in degraded mode.
- Treatment unit A splits its production: the flow goes mainly to tank T1 through lines C and D, and the remainder goes to B, if needed.
- Treatment unit B has two possible inputs: mainly from W2, and possibly from A, if needed. It outputs to T2 through line E.
- Components Ci, Di, and Ei are treatment units. Failure of these components stop their production. They cannot fail when unused.
- Ci are in series, Di are in series, and line D is in cold redundancy with line C. As soon as line C is available, line D is stopped.
- Ei are in hot redundancy. As their production rate is low, they are used whenever it is necessary.
- There are two repairers. A component is not productive during a repair. A repairer is immediately allocated to a failed component. If more than two components are failed when a repairer is freed, he chooses a component at random.

The goal of the model is to predict the average flow rate in the tanks, and the average flow output from the wells, for a long mission time (15 years $\approx 130,000$ hours). For practical reasons, only the production rate for the whole system is presented. The production rate is actually calculated in the model as the sum of the production rates of the two tanks (which equals the production rates of the two wells). Reliability data and flow rates are given in Table 3.9 and in Table 3.10.

Table 3.9.: Reliability data for the components of the oil production system.

Component	λ (h^{-1})	μ (h^{-1})	γ	λ_d (h^{-1})	μ_d (h^{-1})
W1, W2	2.0×10^{-5}	5.0×10^{-3}			
A, B	3.0×10^{-5}	4.0×10^{-3}		9.0×10^{-3}	2.0×10^{-2}
C1, C2	3.0×10^{-3}	4.0×10^{-2}	2.0×10^{-2}		
D1, D2	8.0×10^{-3}	4.0×10^{-2}	2.0×10^{-2}		
E1, E2, E3	4.0×10^{-3}	5.0×10^{-2}			

Repair team	2 members

Table 3.10.: Flow rates (in barrels per time unit) of the components of the oil production system.

	Flow rates	
Component	normal mode	degraded mode
W1, W2	160, 70	
T1, T2	110, 100	
A, B	170, 120	100, 70
C1, C2	120	
D1, D2	80	
E1, E2, E3	50	

Optimizations can be made on several parameters, such as the number of repairers, the flow rates of some key elements (A, B, ...) or the failure rates of the components. As the goal of this example is to study the partial generation method on a production system, such optimizations are not studied.

A "safer" version of this system is also studied, as the availability of the components proved to have an important impact on the accuracy of the partial generation method. Flow rates are the same. Table 3.11 gives the modified reliability data.

Table 3.11.: Reliability data for the components of the "safer" oil production system.

Component	λ (h^{-1})	μ (h^{-1})	γ	λ_d (h^{-1})	μ_d (h^{-1})
W1, W2	6.0×10^{-6}	8.0×10^{-3}			
A, B	2.0×10^{-5}	4.0×10^{-2}		9.0×10^{-3}	2.0×10^{-2}
C1, C2	2.0×10^{-4}	7.0×10^{-1}	1.0×10^{-2}		
D1, D2	5.0×10^{-4}	7.0×10^{-1}	2.0×10^{-2}		
E1, E2, E3	4.0×10^{-3}	5.0×10^{-1}			

Repair team	2 members

Results and Analysis

The Markov chain for the oil production system can be completely generated and calculated, so it can be used as a reference. The complete Markov chain has $46,958$ states and $5,209,692$ transitions.

Tables 3.12 and 3.13 show the evaluation of the production rate, with mission time $T = 130,000$ hours, for several sizes of partial model, without sink, or with sink. The production rate is evaluated, as well as the relative error.

Table 3.12 shows that the production rate is well approximated without sink, even if the relative error does not always decreases when the number of transitions rises. This table also shows that the production rate is poorly approximated using the sink, and the results are worthless as the bounds are too far from each other.

Table 3.13 shows the number of transitions and the number of states of the partial models and the calculation times. The number of transitions is always higher than the threshold in the model with sink. This is due to remaining candidates that shares the same distance once the threshold has been reached (see Section 4.2.5), as they are explored too. The number of transitions for the model without sink is lower than the threshold, as there are many transitions leading from the explored part of the reachability graph to candidates. These transitions are removed (see Algorithm 2.3).

Table 3.12.: Measures of the production rate Q for several thresholds on the size of the model, with given mission time of $T = 130$kh, for the oil production system.

	Partial chain without sink		Partial chain with sink		
Threshold ξ	Q	$\eta(T)$[a]	Q_{lb}	Q_{ub}	$\eta(T)$
$5,209,692$	130.97	-	-	-	-
$1,736,564$	130.97	7.6355×10^{-6}	74.976	164.82	1.1983
$520,969$	116.42	0.11104	9.543×10^{-5}	210	$2.2006e + 06$
$173,656$	131.39	0.0032146	1.3095×10^{-25}	210	$1.6036e + 27$

[a] without sink, Q for the highest threshold is used as reference.

Table 3.13.: Sizes of the models and computation times for several thresholds on the size of the model, with given mission time of $T = 130$kh, for the oil production system.

		Partial chain without sink			Partial chain with sink	
Threshold ξ	Fraction	States	Transitions	Times[a] (s)	Transitions	Times[a] (s)
$5,209,692$	-	$46,958$	$5,209,692$	$1,600 + 220$	-	-
$1,736,564$	1/3	$14,554$	$1,465,230$	$830 + 67$	$1,747,958$	$830 + 1,300$
$520,969$	1/10	$4,405$	$390,350$	$290 + 360$	$532,137$	$290 + 470$
$173,656$	1/30	$1,297$	$98,750$	$89 + 9$	$177,565$	$89 + 170$

[a] (generation of the chain) + (calculation of the chain).

The number of removed transition compared to the threshold is an indicator of the progress of the exploration. If there are lots of removed transitions (such as $\approx 75,000$ against $\approx 170,000$, which is $\approx 45\%$), the exploration is stopped but the model was blooming. Indeed, there are lots of transitions leading to unknown candidates, which means that the edge of the known model is fully connected to new candidates. If there are fewer removed transitions (such as $\approx 250,000$ against $\approx 1,700,000$, which is $\approx 15\%$), the model tends to be more completely explored, as there are less connections to the unexplored part of the state space.

The calculation time for the Markov chain without sink obtain with $\xi = 520,969$ transitions is 360 seconds, which is unexpected. The expected time is around 25 seconds. As detailed in Section 4.3.5, the computation times can fluctuate due to the conditions in which calculations are done. The experiments are done only once and should be repeated multiple times to obtain more accurate computation times.

Tables 3.15 and 3.16 show the evaluation of the production rate for the safer version of the system, with mission time $T = 10,000$ hours. For this variation of the oil production system, the evaluated performance indicator is the unavailability of the system (the system is unavailable when the production is completely stopped). To be able to compare the two versions of the model, the unavailability and the production rate of the two version of the system are given in Table 3.14.

Table 3.14.: Comparison of the indicators for both versions of the oil production system at their respective mission time.

	Version	
Indicator	Normal	Safer
Production rate	130.97	206.18
Unavailability	9.9216×10^{-2}	2.5641×10^{-4}

Table 3.15 shows that the unavailability is well approximated without sink. These results illustrate the sensitivity of the partial generation method to the quality of the components. This can be explained by the fact that the probability is more concentrated around the nominal state for the safer version of the system. Hence, fewer states are necessary to obtain the same accuracy. Moreover, as the transitions leading to the sink have lower rates and because the mission time is reduced, the results given by the partial chain with sink are good.

Table 3.16 shows that the times to generate the chains are the same, with the normal or with the safer version. However, the calculation times of the Markov chains of the safer version are higher than the calculation times of the normal version, even if the mission time was reduced. This is due to the time-step of the calculation method which depends on the highest transition rates of the chain (see Section 2.1.4). Indeed, the highest transition rates are the repair rates, which have been raised for the safer version.

Table 3.15.: Measures of the unavailability Q for several thresholds on the size of the model, with given mission time of $T = 10,000$ hours, for the "safe" version of the oil production system.

	Partial chain without sink		Partial chain with sink		
Threshold ξ	Q	$\eta(T)$[a]	Q_{lb}	Q_{ub}	$\eta(T)$
$5,209,692$	0.00025641	-	-	-	-
$1,736,564$	0.00025641	0	0.00025641	0.00025641	3.2724×10^{-5}
$520,969$	0.00025641	3.5101×10^{-5}	0.0002564	0.00025926	0.011174
$173,656$	0.00026603	0.037538	0.00025629	0.00032801	0.27985

[a] without sink, Q for the highest threshold is used as reference.

Table 3.16.: Sizes of the models and computation times for several thresholds on the size of the model, with given mission time of $T = 10,000$ hours, for the "safe" version of the oil production system.

		Partial chain without sink			Partial chain with sink	
Threshold ξ	Fraction	States	Transitions	Times[a] (s)	Transitions	Times[a] (s)
$5,209,692$	-	$46,958$	$5,209,692$	$1,600 + 770$	-	-
$1,736,564$	1/3	$15,333$	$1,438,588$	$810 + 580$	$1,778,133$	$810 + 1,100$
$520,969$	1/10	$4,757$	$346,316$	$280 + 76$	$541,723$	$280 + 380$
$173,656$	1/30	$1,829$	$99,090$	$93 + 76$	$173,763$	$93 + 160$

[a] (generation of the chain) + (calculation of the chain).

Figure 3.11 shows the evolution of the relative error η of the production rate of the oil production system obtained with Markov chains with sink, as defined in Section 2.4.2, Equation (2.11), with respect to the threshold used for the number of transitions ξ. The maximum production rate Q_{max} is 210. For each mark on the plot, a model was partially generated up to ξ transitions and the production rate was calculated at the given mission time $T = 130,000$ hours. The dashed line $\eta = 1.0$ indicates that the partial model is trustworthy when there are more than $2,000,000$ transitions in the Markov chain. It is interesting and comforting to note that the error is always decreasing when the threshold is raised.

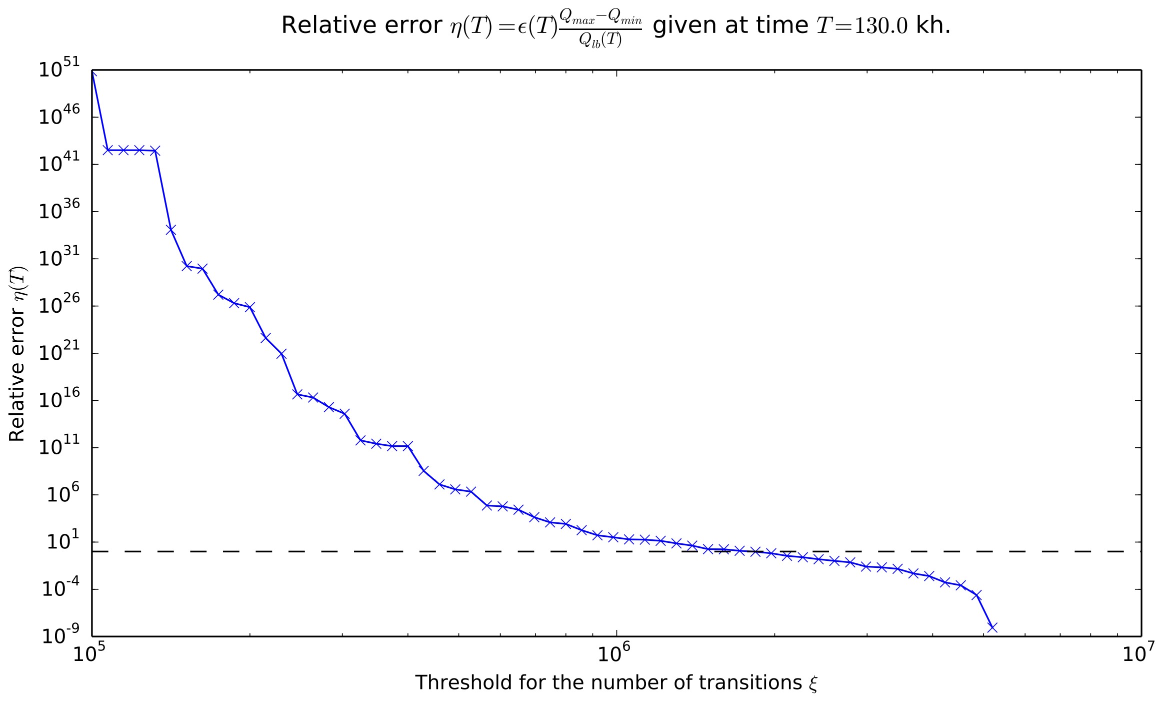


Figure 3.11.: The evolution of the relative error for the production rate of the oil production system at time T with respect to the threshold on the number of transitions.

Figure 3.12 shows the evolution of the relative error η of the production rate with respect to the threshold used for the number of transitions ξ, for the normal oil production system, for two mission times: $T = 10,000$, and $T = 130,000$ hours. For a mission time of $10,000$ hours, the model is trustworthy when there are more than $800,000$ transitions in the Markov chain. For a mission time of $130,000$ hours, the model is trustworthy when there are more than $2,000,000$ transitions. This result was predictable, as the error is calculated with the probability to be in the sink, which is an absorbing state. Mission times that are too long *will* lead to bad results with the sink. This shows the heavy influence of the sink state in the Markov chain, as it drains the probability from the rest of the chain. It is interesting to note that the mission time has more influence when the partial Markov chain is small.

Figure 3.13 and Figure 3.14 show results for the safer version of the system, at mission time $T = 10,000$ hours. Figure 3.13 shows the evolution of the relative error η of the unavailability

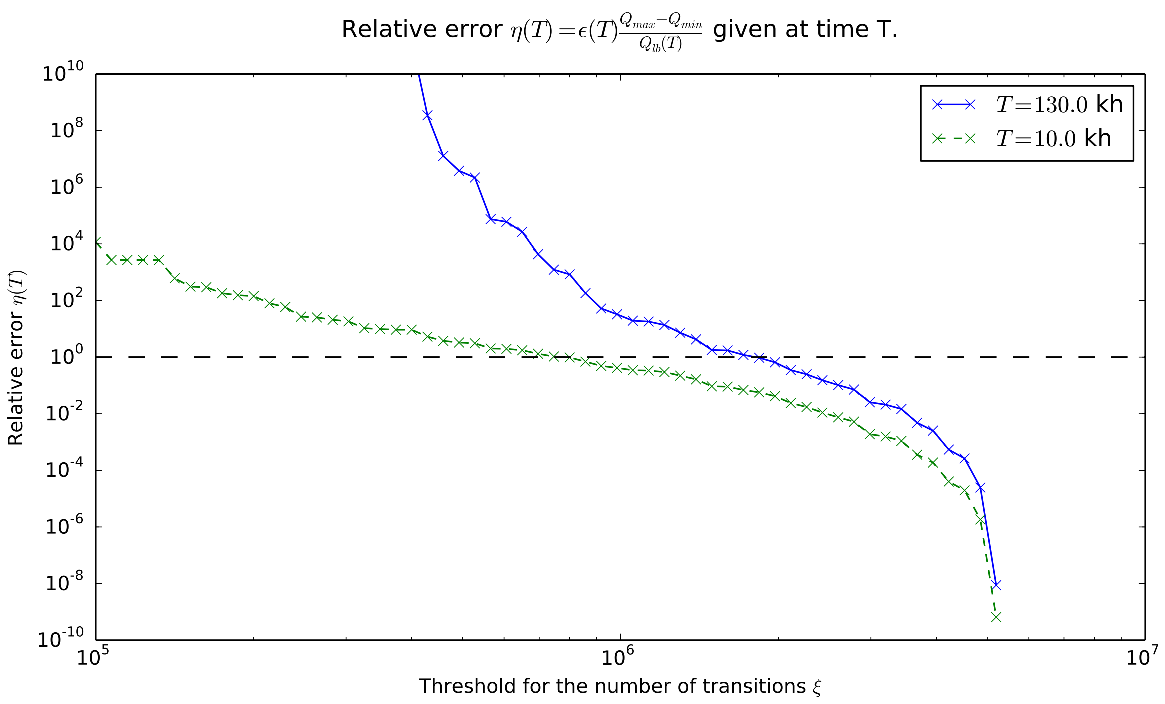


Figure 3.12.: Comparison of the evolution of the relative error for the production rate of the oil production system with respect to the threshold, for different mission times.

of the system with respect to the threshold used for the number of transitions ξ. Figure 3.14 shows the evolution of the relative error η of the production rate of the system with respect to the threshold used for the number of transitions ξ, for both versions of the system.

In Figure 3.13, it is shown that the unavailability is very well approximated for the safer system. In Figure 3.14, it is shown that the production rate is approximated with much more accuracy with the safer system. The safety parameters of the system have heavy influence on the quality of the partial generation. For safer systems, the probability is more concentrated around the nominal state and the partial generation keeps more of the states where the probability is scattered. This leads to a better evaluation of the performance indicators.

To further analyze the method, the relevance factor $R()$ used during the generation of the Markov chain (see Equation (2.10) and Algorithm 2.2) is compared, for each state of the Markov chain, to the probability that is calculated with the full Markov chain. Figure 3.15 shows this correlation for the oil production system at mission time $T = 130,000$ hours. Each mark is a state of the chain for which the x-coordinate is the relevance factor and the y-coordinate is the probability computed with the Markov chain. Note the log scales. Overall, the plot shows that a correlation exists between the relevance factor and the probability, because when the relevance factor is low, the probability is also low. This comforts the ability of the partial generation to “estimate” where the probability will be in the final Markov chain. However, this prediction is not very accurate, because some states with the same probability (e.g. 10^{-12}) may have very different relevance factor (10^{-21} to 10^{-13}). When the probability

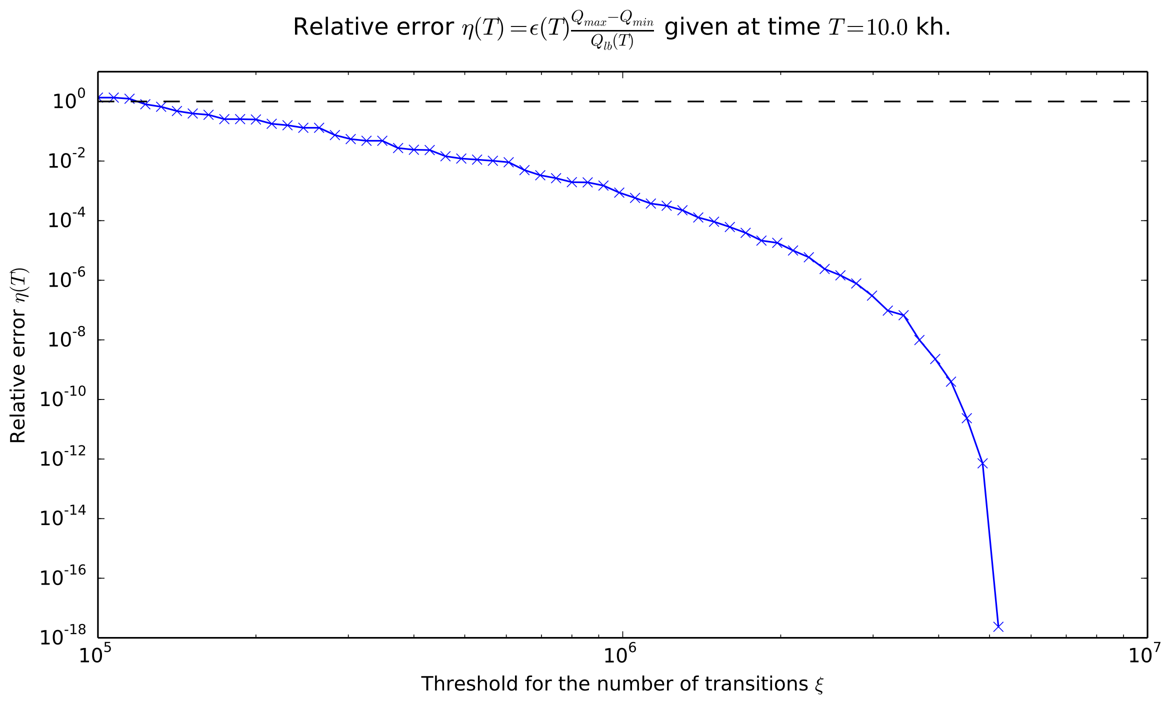


Figure 3.13.: The evolution of the relative error for the unavailability with respect to the threshold on the number of transitions, for the safer version of the system.

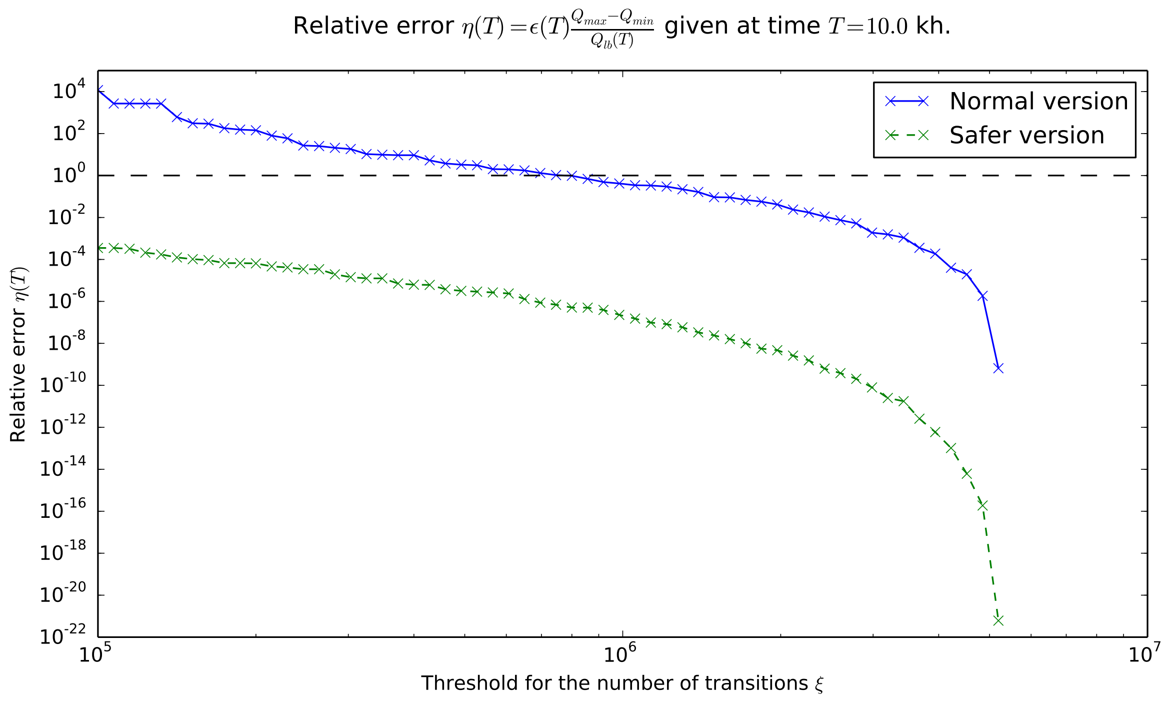


Figure 3.14.: The evolution of the relative error for the production rate with respect to the threshold on the number of transitions, for both versions of the system.

is scattered over a lot of states, this may lead to inaccurate results, because the most probable states may not be selected.

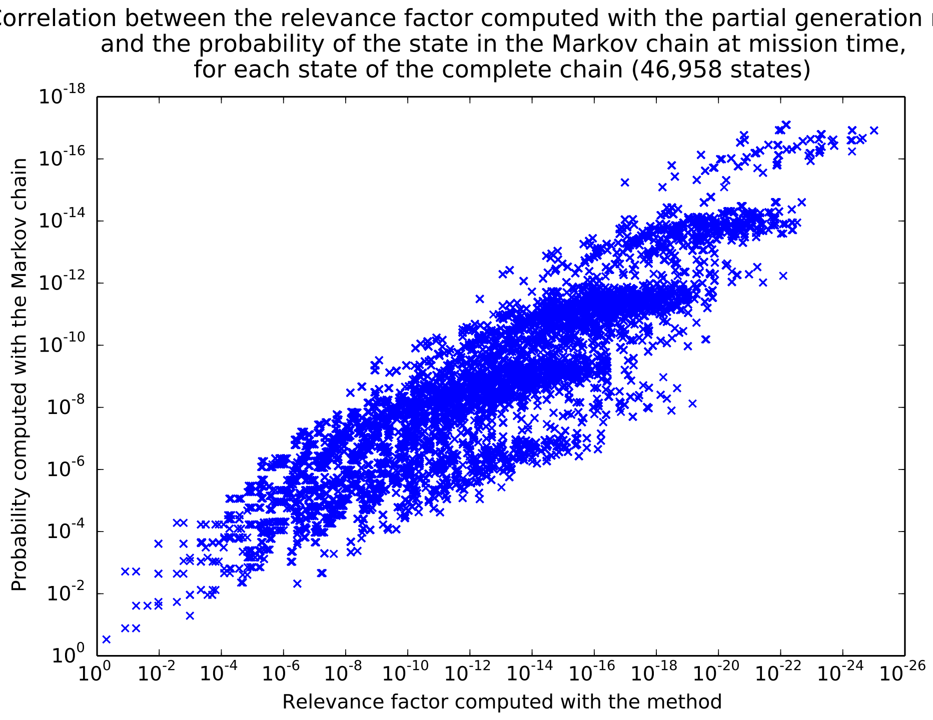


Figure 3.15.: Correlation between the relevance factor computed during the generation of the chain and the probability calculated with the Markov chain for each state of the full chain for the oil production system at mission time $T = 130,000$ hours.

Figure 3.16 shows the same correlation for the "safer" oil production system, at mission time $10,000$ hours. In this case, the distance and the probability are much more correlated. It is interesting to note that the probabilities of the states are much lower for the safer version of the system. The safer system, for which the probability is more concentrated around the nominal state, will give more accurate results with the partial chain than the normal system.

Another type of interesting correlation is to compare the probabilities of the states, whether there are part of the partial chain without sink (threshold $\xi = 1,000,000$ transitions) or of the full chain ($5,209,692$ transitions). This is shown Figure 3.17. Each mark is a state of the partial chain. The x-coordinate is the probability computed with the partial chain and the y-coordinate is the probability computed in the complete chain, at mission time $T = 130,000$ hours. The scales are logarithmic. States that are on the line "$y = x$" are the states which have *exactly* the same probability in the partial chain or in the full chain. These states have the same behavior in both chains. This means that the partial generation did not alter their local behavior. These states are in the core of the Markov chain, and are not affected by the partial generation. The states for which $x \neq y$ are states which are not correctly approximated (there are both underestimate and overestimate states). These states are on the border of the small chain, in that some of their neighbors have not been selected. It is impossible to

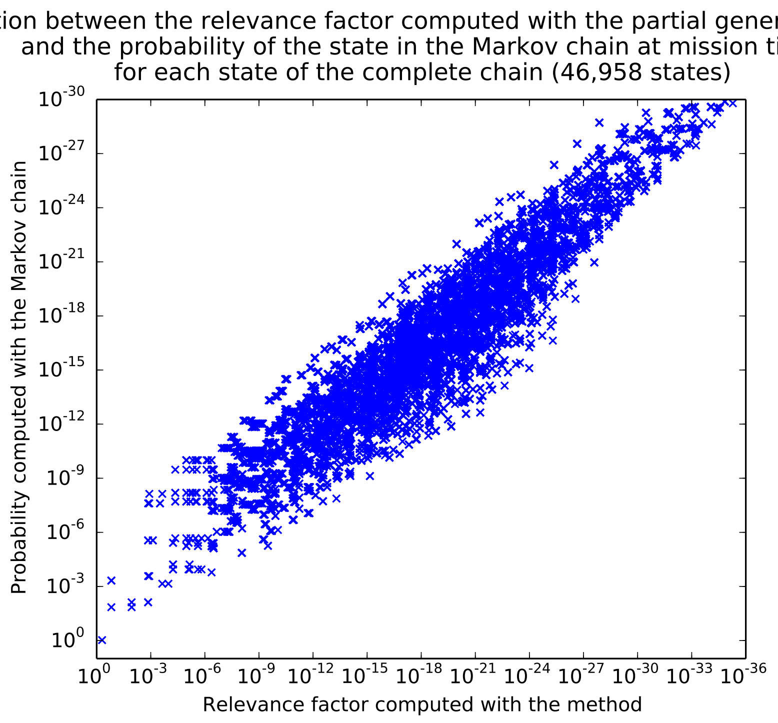


Figure 3.16.: Correlation between the relevance factor computed during the generation of the chain and the probability calculated with the Markov chain for each state of the full chain for the "safer" oil production system, mission time $T = 10,000$ hours.

predict or to calculate, which states will be affected by the partial generation. Because the probabilities of the "affected" states are low (below 10^{-4}) and because some of them are under-probable while others are over-probable, the overall production is evaluated with good accuracy in the partial chain without sink. This is why the reduced chain without sink is accurate, while it is not possible to calculate the error made with this chain without the full chain.

Figure 3.18 shows the same correlation for the "safe" oil production system. The probabilities of the badly approximated states are lower ($< 10^{-8}$), so the production rate is more accurately computed for this version of the system. The states that are wrongly approximated are states in which multiple components have failed. This is why their probability is lower for the safer system.

Figure 3.19 shows the correlation between the probabilities of the most probable states of the partial chain and the probabilities of the most probable states of the full chain. A mark may represent different states in the two chains. The probabilities are sorted, so that the most probable states of the partial chain can be compared to the most probable states of the full chain. The x-coordinate of a mark represent the probability of a state in the partial chain and the y-coordinate represent the probability of a state in the full chain. This shows how the partial generation did not alter the overall behavior of the chain, even if it may alter the evaluation of the performance indicators. The marks on the right of the plot (which have

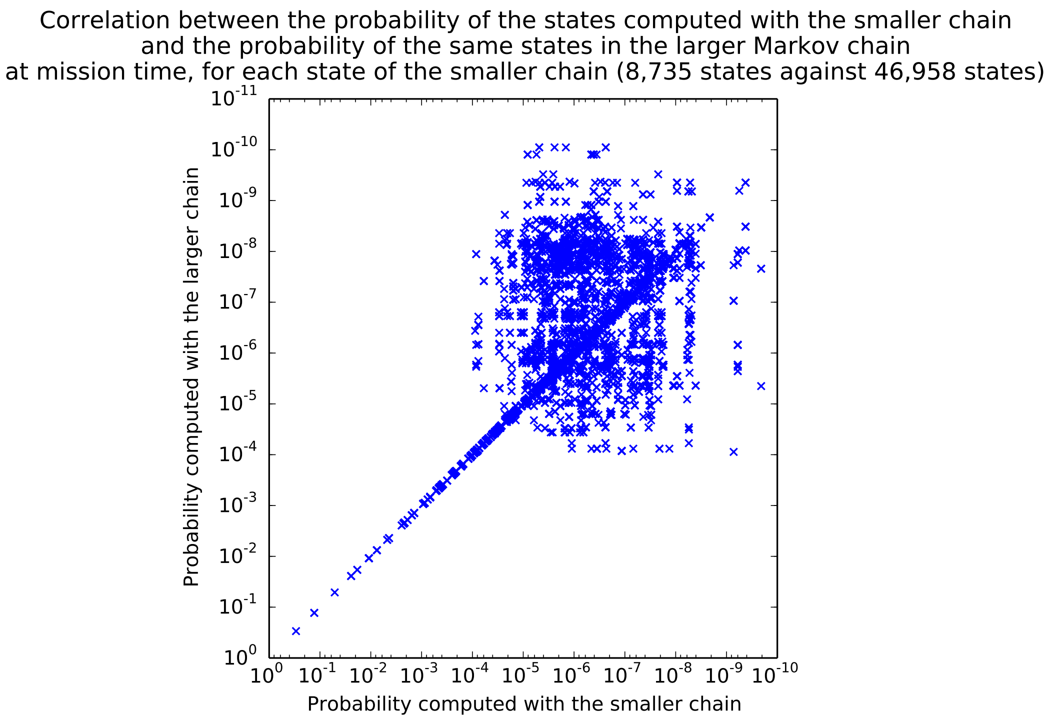


Figure 3.17.: Correlation between the probability of each state of the partial model (threshold $\xi = 1M$ transitions) whether they are in the partial chain or in the full chain, for the oil production system, with mission time $T = 130,000$ hours.

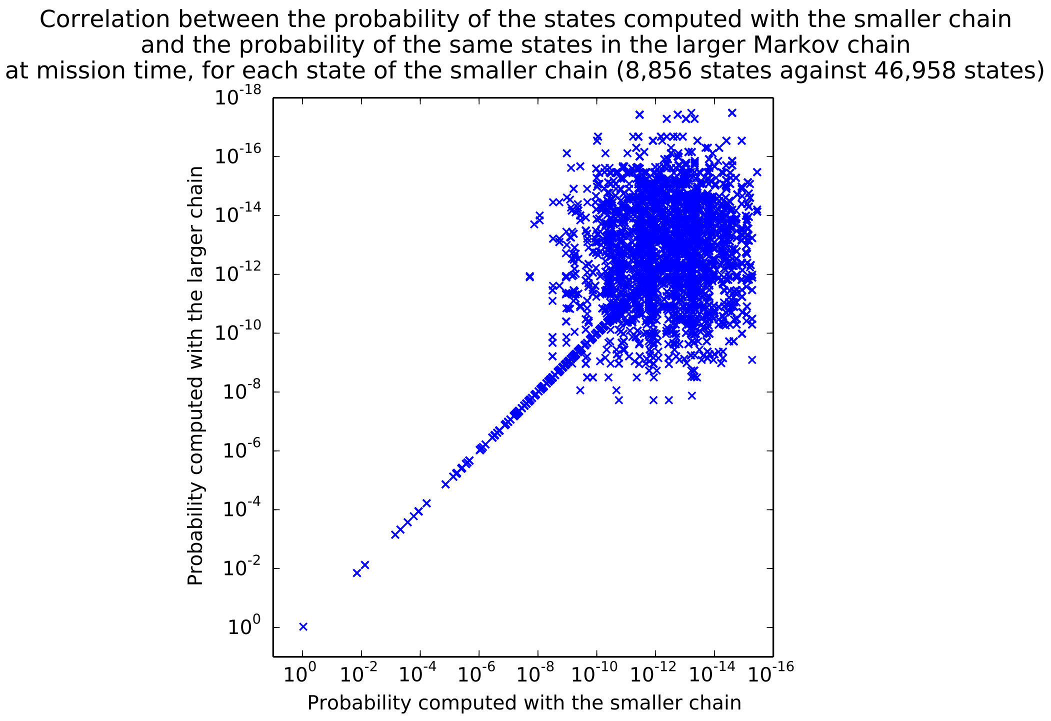


Figure 3.18.: Correlation between the probability of each state of the partial model (threshold $\xi = 1M$ transitions) whether they are in the partial chain or in the full chain, for the “safer” oil production system, with mission time $T = 10,000$ hours.

lower probability in the small chain than in the full chain) are states that are on the edge of the partial chain. These states have very low probability in the partial chain, and there are still some more probable states in the full chain.

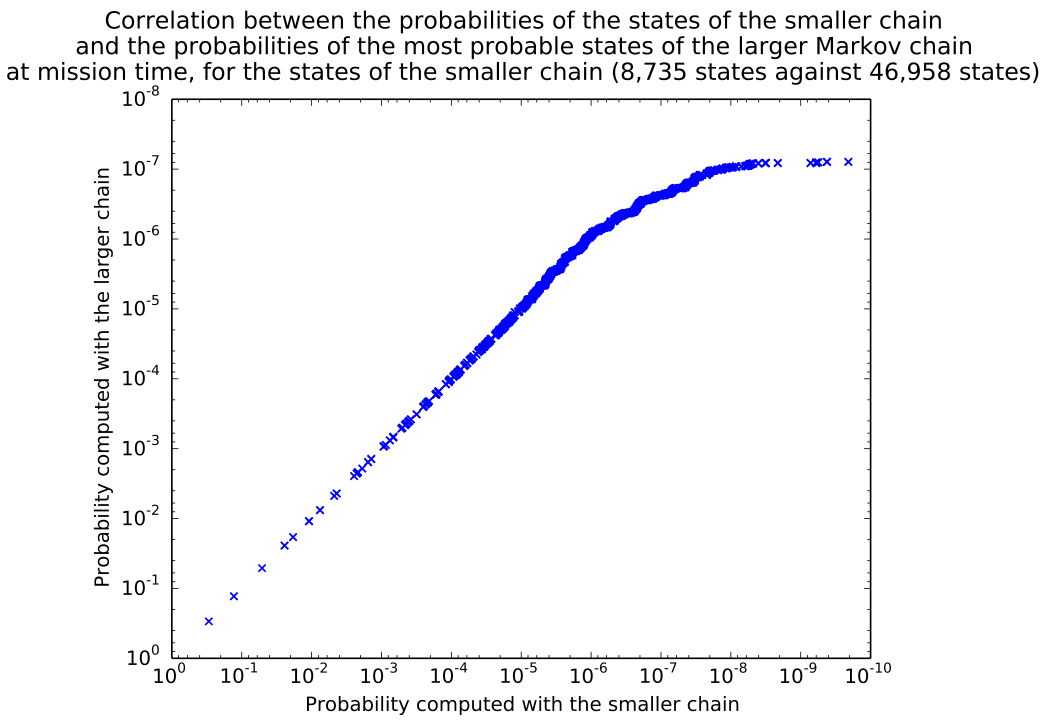


Figure 3.19.: Correlation between the probability of the most probable states in the partial chain and the most probable states in the full chain, for the oil production system, with mission time $T = 130,000$ hours.

First, this may highlight that the selection of the most probable states failed. Second, this may show that the local changes in the behavior of the states in the partial chain *induce* a more global change on the border of the chain.

Figure 3.20 shows the same correlation, but for the "safer" oil production system. The border effects are less influential, which leads to more accurate results.

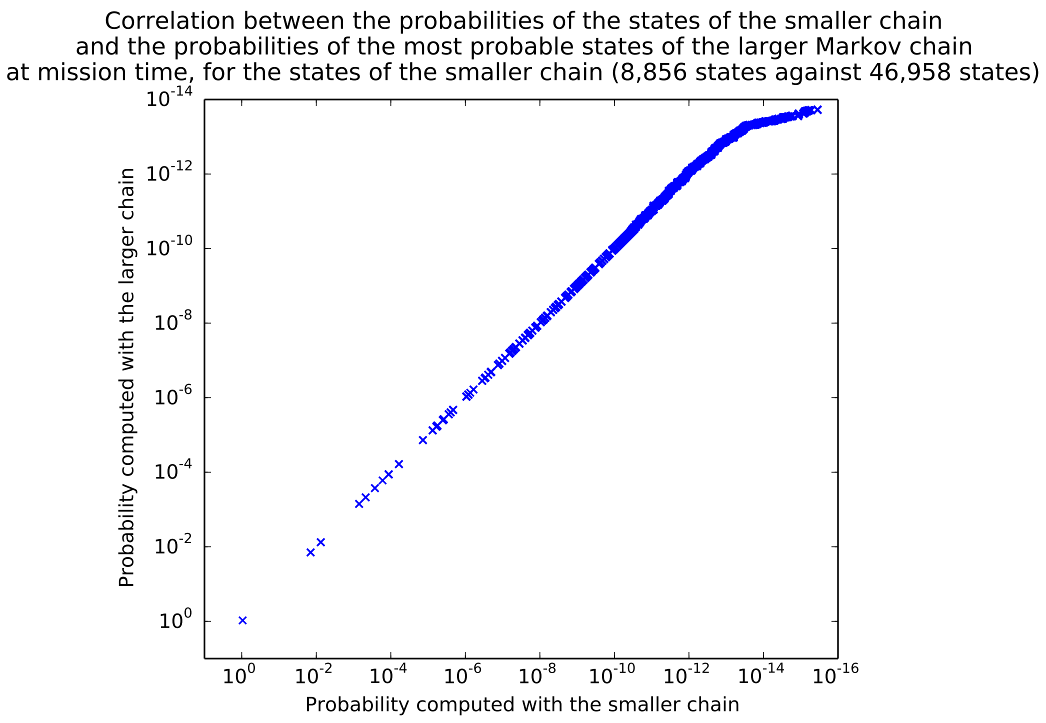


Figure 3.20.: Correlation between the probability of the most probable states in the partial chain and the most probable states in the full chain, for the "safer" oil production system, with mission time $T = 10,000$ hours.

3.2.4. Emergency Nuclear Power Plant Power Supply

The emergency power supply is a critical system whose goal is to provide energy to manage a nuclear power plant. It is an electric power system with repairable components, failures on demand, cold redundancies, and common cause failures.

Description of the System

The system is pictured Figure 3.21. It is used in various articles [Pagès and Gondran 1980] [Bouissou and Bon 2003] [Rauzy 2004]. We shall use here an augmented version of the model given in [Rauzy 2004].

The role of the system is to supply electricity out of the boards LHA or LHB. The regular power supply of boards LHA and LHB comes from the transformer TS. TS is supplied by the NET and by the plant PLT. When the NET is available, PLT works in regular mode. Otherwise the PLT works in standalone mode, which is rather unstable. LHA and LHB can also be powered by the NET alone through transformer TA. Diesels generators DA and DB supply respectively LHA and LHB when these boards are not powered by LGD and LGF. Boards LGD, LGF, LHA and LHB may fail.

To obtain a model which Markov chain cannot be completely generated, the original system is extended with circuit breakers CBx whose role is to protect the boards. They were chosen

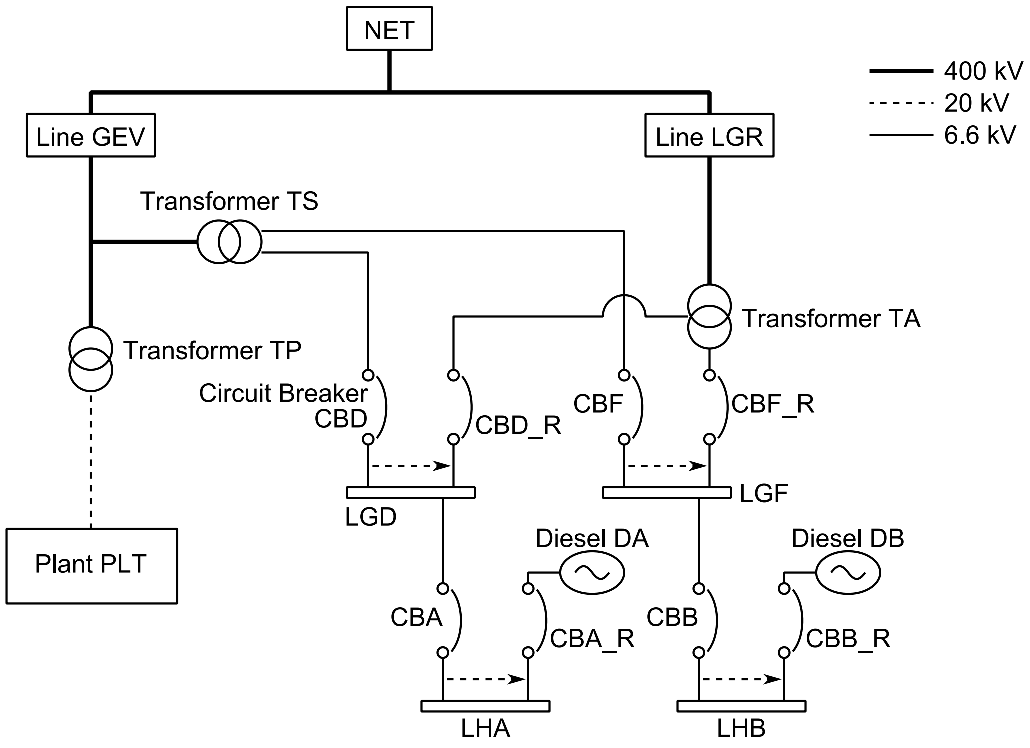


Figure 3.21.: An electric power supply system.

to have an influence on the overall availability, while not being the prevailing components. Careful attention was given to avoid symmetries, useless components and other means to simplify the model.

Table 3.17 gives the reliability data taken from [Rauzy 2004]. Components are repairable, have a failure rate λ, a repair rate μ. The plant PLT may fail on demand to switch to standalone mode with probability γ. Diesels may fail on demand with probability γ and have a common cause failure with rate λ_{cc} and repair rate μ_{cc}. Diesels which failed on demand are repaired with rate μ_d. Moreover, diesels have a 100 times higher failure rate when they are used (warm redundancies).

The problem at hand is to assess the unavailability of the system, typically for a mission time of $10,000$ hours, i.e. about one year.

The system is made of the following elements: 18 "binary" repairable components, the plant which has 2 failure modes (failure and failure on demand), and the 2 diesel generators which have 3 failure modes. The estimated number of states is then about 2^{26} (67 millions) and the estimated number of transitions is 26×2^{26} (1.7 billions). However, the most probable states should not be too numerous, as the components are highly available.

Table 3.17.: Reliability data for the Electric Power Supply System.

	λ (h^{-1})	μ (h^{-1})	γ
NET	1.0×10^{-6}	8.0×10^{-3}	
GEV, LGR	5.0×10^{-6}	1.0×10^{-2}	
TP, TS, TA	2.0×10^{-6}	1.0×10^{-3}	
LGD, LGF, LHA, LHB	2.0×10^{-7}	1.0×10^{-1}	
PLT (regular mode)	1.0×10^{-4}	1.0×10^{-1}	0.5
PLT (standalone mode)	1.0×10^{-1}	1.0×10^{-3}	
DA, DB	1.0×10^{-4}	2.0×10^{-2}	0.001
CBD, CBD_R, CBF, CBF_R CBA, CBA_R, CBB, CBB_R	1.0×10^{-4}	1.0×10^{-1}	

	μ_d (h^{-1})	λ_{cc} (h^{-1})	μ_{cc} (h^{-1})
DA, DB	1.0×10^{-1}	1.0×10^{-4}	1.0×10^{-2}

Results and Analysis

The complete Markov chain cannot be calculated for this system. However, a Markov chain with 1 million transitions proved sufficient for the chosen mission time of $10,000$ hours. To test the scalability of the method, a Markov chain with 100 million transitions is evaluated.

Table 3.18 shows the evaluation of the unavailability of the power supply, and Table 3.19 shows the corresponding sizes of the chains and the calculation times. The relative error for the Markov chain without sink is evaluated with respect to the unavailability obtained with the chain made of 100 million transitions. Of course, it is not possible to calculate the relative error of the unavailability obtained with this chain without sink.

Table 3.18 shows that the system is well approximated by a Markov chain without sink with only $100,000$ transitions, but it is not possible to validate this result without sink or without larger chains. The Markov chain with sink with $100,000$ transitions is not large enough to obtain close bounds at the chosen mission time. However, 1 million transitions for the Markov chain with sink gives close-enough bounds.

Note that $Q = 0$ when $\xi = 10,000$ transitions. This is due to the lack of failure states in such a small model.

Table 3.19 shows the number of transitions, the number of states of the partial chains, and the computation times. At the end of the algorithm which computes the partial chain without sink, transitions leading to candidates are removed (see Algorithm 2.3). It is interesting to see the proportion of transitions that are kept for the Markov chain without sink. When the threshold $\xi = 100,000$ transitions, more than 70% are removed, which means that there are

Table 3.18.: Measures of the unavailability Q of the power supply for several thresholds on the size of the model, with given mission time of $T = 10,000$ hours.

	Partial chain without sink		Partial chain with sink		
Threshold ξ	Q	$\eta(T)$[a]	Q_{lb}	Q_{ub}	$\eta(T)$
$100,000,000$	5.5798×10^{-7}	-	5.5798×10^{-7}	5.5798×10^{-7}	1.6203×10^{-22}
$10,000,000$	5.5798×10^{-7}	0	5.5798×10^{-7}	5.5798×10^{-7}	3.331×10^{-9}
$1,000,000$	5.5798×10^{-7}	1.7922×10^{-6}	5.5797×10^{-7}	5.611×10^{-7}	0.005607
$100,000$	5.5558×10^{-7}	0.0042869	5.4373×10^{-7}	2.6449×10^{-5}	47.643
$10,000$	0	1	0	0.0081711	-

[a] without sink, Q for the highest threshold is used as reference.

numerous transitions leading to the unexplored part of the chain. In these conditions, it is not surprising to see that the Markov chain with sink gives bad results, because 70% of the transitions of the whole model go to the sink. When $\xi = 100$M, "only" 15% of the transitions are redirected to the sink. This means that the border of the chain is cleaner, and better results are expected.

Looking at the stabilization time of big systems given in Section 2.2.2, the time constant for this system is roughly $\frac{1}{\min \mu} \approx 1,000$ hours. At mission time $T = 10,000$ hours, the system is bound to have reached its steady-state, and the steady-state probability of each state is reached.

When $\xi = 100$M transitions, under the steady-state condition of the chain without sink, in the chain with sink, the 15 million transitions leading to the sink could not raise the probability of the sink higher than 10^{-29} (according to the definition of η in Equation 2.11). This means that the 4 million states filter the probability, which never flows up to the border of the chain. It is legitimate to remove such *useless* states.

Table 3.19.: Sizes of the models of the power supply and computation times for several thresholds on the size of the model, with given mission time of $T = 10,000$ hours.

	Partial chain without sink			Partial chain with sink	
Threshold ξ	States	Transitions	Times[a] (s)	Transitions	Times[a] (s)
$100,000,000$	$4,415,444$	$84,655,044$	$34,000 + 52,000$	$100,003,389$	$34,000 + 58,000$
$10,000,000$	$429,917$	$5,886,890$	$3,000 + 2,200$	$10,000,130$	$3,000 + 3,400$
$1,000,000$	$41,177$	$415,160$	$500 + 290$	$1,000,162$	$500 + 590$
$100,000$	$3,988$	$28,601$	$43 + 7$	$100,223$	$43 + 33$
$10,000$	375	$1,723$	$3 + 0$	$10,014$	$3 + 1$

[a] (generation of the chain) + (calculation of the chain).

A “stiffer” version of the system is also analyzed. The goal of such a model is to discuss the problems with stiff Markov chains. To obtain the stiff version of the system, all the failure rates λ were divided by ten and all the repair rates μ where multiplied by ten. The failures on demand are not modified. As each component of the system has its availability roughly multiplied by 100, the expected unavailability of the stiff power supply should be lower.

Tables 3.20 and 3.21 show respectively the calculated unavailability for this stiff model, and the size and calculation times of the chains.

Table 3.20.: Measures of the unavailability Q of the stiff power supply for several thresholds on the size of the model, with given mission time of $T = 10,000$ hours.

	Partial chain without sink		Partial chain with sink		
Threshold ξ	Q	$\eta(T)$[a]	Q_{lb}	Q_{ub}	$\eta(T)$
$100,000,000$	4.9333×10^{-14}	-	4.9333×10^{-14}	4.9333×10^{-14}	7.7064×10^{-36}
$10,000,000$	4.9333×10^{-14}	0	4.9333×10^{-14}	4.9333×10^{-14}	2.6665×10^{-15}
$1,000,000$	4.9333×10^{-14}	0	4.9333×10^{-14}	4.9335×10^{-14}	3.6685×10^{-5}
$100,000$	4.9307×10^{-14}	0.00052095	4.9265×10^{-14}	1.5486×10^{-11}	313.33
$10,000$	0	1	0	1.1935×10^{-6}	-

[a] without sink, Q for the highest threshold is used as reference.

Calculation times of the Markov chains are multiplied by ten for the stiffer models ($580,000$ seconds is 6 days 17 hours). This is due to the repair rates that have been multiplied by ten, as the time complexity of the Markov chain calculator is roughly proportional to $\frac{1}{\max \mu}$ (see Section 2.1.4). Further analysis of the calculation times is done in Section 4.3.5.

Table 3.21.: Sizes of the models of the stiff power supply and computation times for several thresholds on the size of the model, with given mission time of $T = 10,000$ hours.

	Partial chain without sink			Partial chain with sink	
Threshold ξ	States	Transitions	Times[a] (s)	Transitions	Times[a] (s)
$100,000,000$	$4,423,430$	$84,603,586$	$34,000 + 510,000$	$100,004,689$	$34,000 + 580,000$
$10,000,000$	$431,270$	$5,853,661$	$2,900 + 23,000$	$10,000,018$	$2,900 + 36,000$
$1,000,000$	$41,207$	$404,632$	$510 + 2,300$	$1,000,052$	$510 + 5,400$
$100,000$	$3,906$	$25,928$	$45 + 51$	$100,369$	$45 + 350$
$10,000$	388	$1,636$	$4 + 1$	$10,425$	$4 + 10$

[a] (generation of the chain) + (calculation of the chain).

In Table 3.20, it is interesting to see that the same threshold $\xi = 100,000$ gives a more accurate Markov chain without sink in the case of the stiff model, but a less accurate chain with sink. For the higher thresholds, the accuracy is better, with or without sink.

There are two concurrent phenomena which explain this statement. On the one hand, the quantity of interest is the unavailability, which is lower for the stiff version of the system by several orders of magnitude. Hence, to obtain accurate results, the exploration must go deeper in the state space of the system, as failure states are less probable by several orders of magnitude. On the other hand, as the probability of a state with k failed components is roughly $\left(\frac{\lambda}{\mu}\right)^k$ (see Section 2.2.2), the states become more rapidly improbable as the exploration goes on.

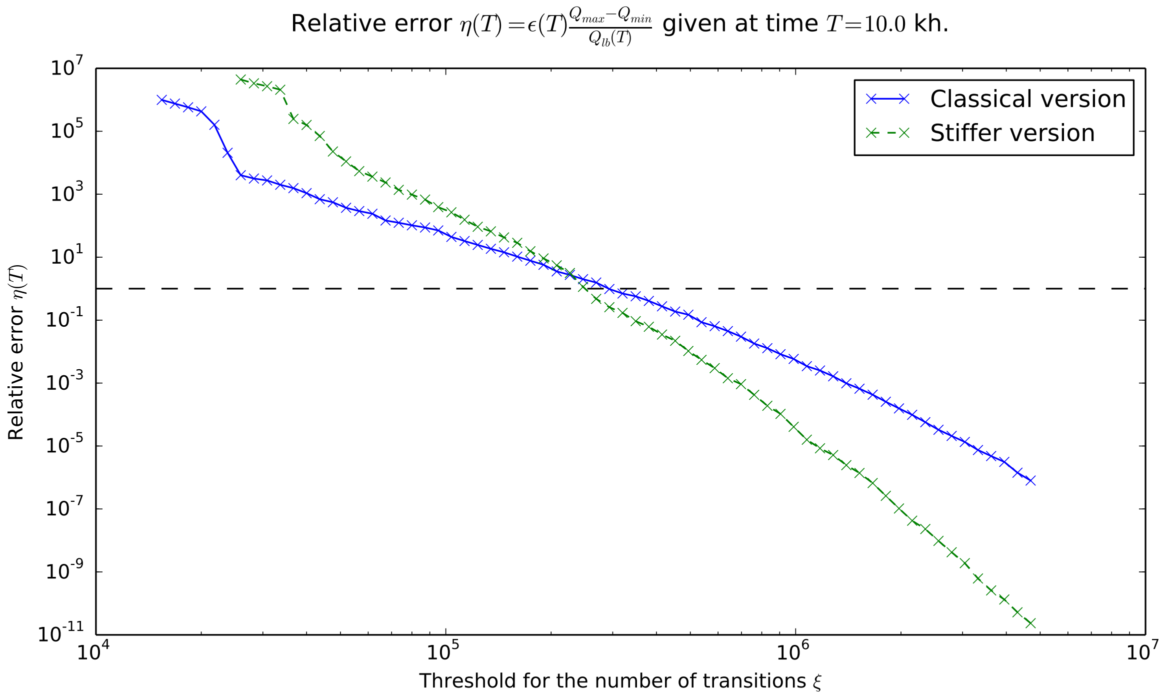


Figure 3.22.: The evolution of the relative error for the unavailability compared between the versions of the power supply at time T with respect to the threshold on the number of transitions.

Figure 3.22 shows the comparison of the evolution of the relative error for the classical and stiff version of the power supply, obtained with Markov chains with sinks. The relative error decreases with the size of the partial models in both cases. However, as stated before, the relative error decreases faster for the stiff power supply. For this mission time, Markov chains obtained with 200,000 transitions yield the same relative error, which is around 4. It is coincidental that the relative error is near 10^0 for both set of chains at this point. On the left of the plot, some points of the relative error are missing for the stiff version. In these points, the unavailability is null (there are no failure states in Markov chains that small), so $\eta = +\infty$.

To further analyze the method, the relevance factor $R()$ used during the generation of the partial Markov chain is compared to the probability computed with the Markov chain without sink (the relevance factor is defined in Equation (2.10)). Figure 3.23 shows this correlation for each state of the power supply in a Markov chain generated up to $\xi = 1,000,000$ transitions, at mission time $T = 10,000$ hours. Each mark on the graph is a state for which the x-coordinate is the relevance factor and the y-coordinate is the probability computed with the Markov chain. Note the logarithmic scales. The plot shows that, overall, the probability decreases when the relevance factor decreases, but the same relevance factor can lead to several orders of magnitude of the probability (for instance, a relevance factor of $\approx 10^{-8}$ gives probabilities from 10^{-14} to 10^{-5}). However, the relative error obtained with this Markov chain is $\approx 10^{-6}$, which means that enough probable states are kept.

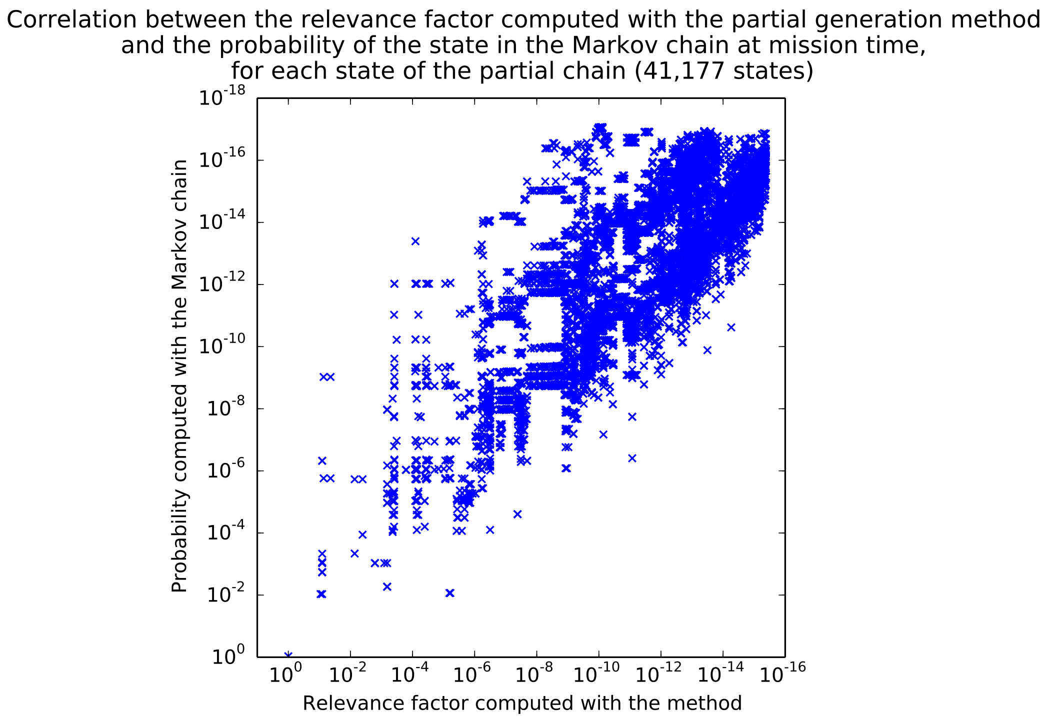


Figure 3.23.: Correlation between the relevance factor computed with the partial generation method and the probability with the Markov chain for each state of the partial chain for the power supply, with threshold $\xi = 1,000,000$ transitions and mission time $T = 10,000$ hours.

Figure 3.24 shows the same correlation between the relevance factor and the probabilities, but applied to the stiff version of the system. In this plot, the relevance factor is slightly more correlated with the probabilities, and, to fewer exceptions, the probability is higher than the relevance factor.

The effects of the sink on a Markov chain have been previously observed on the relative error. It is also interesting to observe the effect on the sink on each state of the chain. Figure 3.25 shows the correlation between the probabilities of states of partial Markov chains obtained with the same threshold $\xi = 100,000$ transitions, at the same mission time $T = 10,000$ hours. The system is the normal power supply. The difference between the two chains is that one is obtained with the partial generation without sink, and the other with sink. Each mark is a

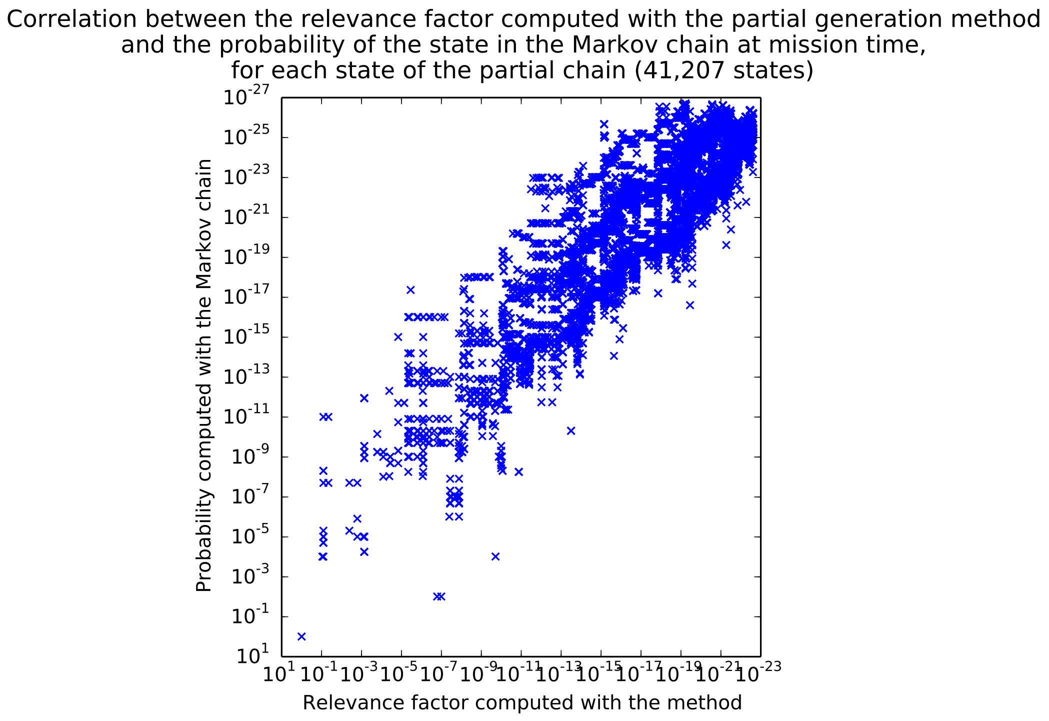


Figure 3.24.: Correlation between the relevance factor computed with the partial generation method and the probability with the Markov chain for each state of the partial chain for the stiff power supply, with threshold $\xi = 1,000,000$ transitions and mission time $T = 10,000$ hours.

state of the chain without sink. The x-coordinate is the probability of the state in the chain without sink, and the y-coordinate is kits probability in the chain with sink. The unavailability calculated with the Markov chain with sink is evaluated to 5.44×10^{-7} (Q_{lb} from Table 3.18), and the sink has a probability of 2.6×10^{-5} (Q_{ub} in Table 3.18), which gives a high relative error of 47.

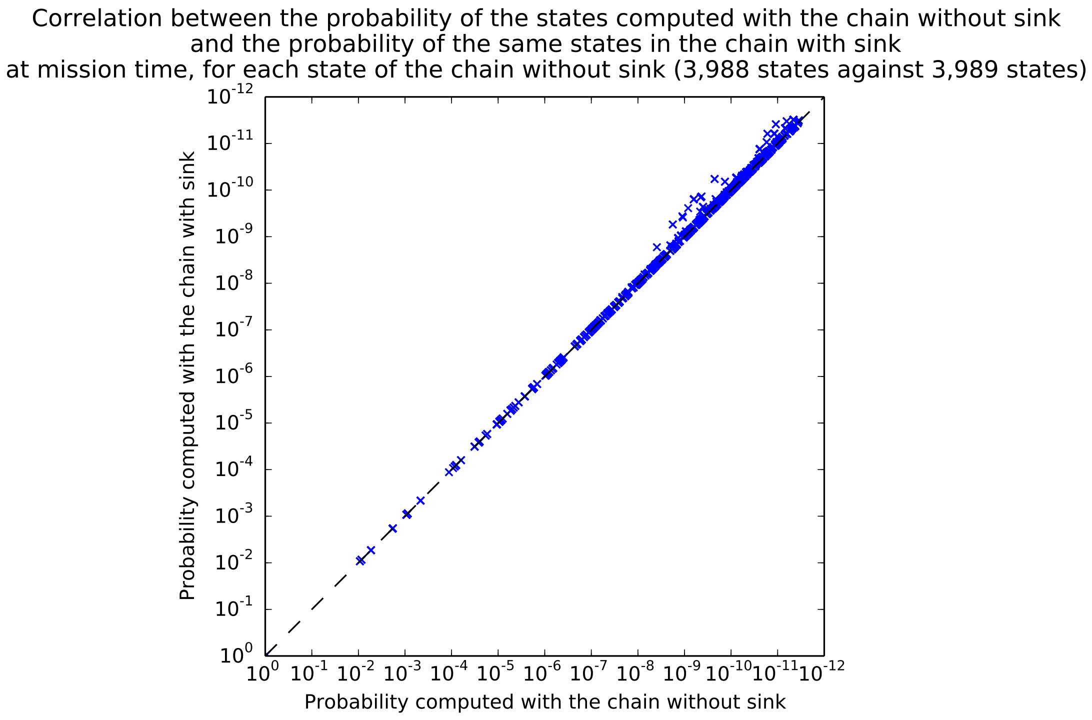


Figure 3.25.: Correlation between the probability of each state of the small chain ($\xi = 100$k transitions) whether they are in the chain with or without sink, for the power supply, with mission time $T = 10,000$ hours.

On Figure 3.25, it is possible to see the states which probability is drained by the sink. The significant changes are only on very improbable states (states with probability lower than 10^{-8}), and affects them by dividing their probability by a factor ten. This leads to a very pessimistic evaluation of the error, even if the Markov chain is not that much influenced by the sink.

Figure 3.26 shows the same correlation for the stiff version of the system. As expected, the probability drained from the states to the sink is lower, and the sink is less influential.

Another type of correlation can be drawn. This is the correlation between the probabilities of states of Markov chains obtained with different thresholds. The probability of each state of the small Markov chain is compared to the probability of the same state in the bigger Markov chain.

Figure 3.27 shows this correlation. The thresholds are $\xi = 100,000$ transitions for the small chain, and $\xi = 5,000,000$ transitions for the large chain. Both Markov chains are generated without sink, and the mission time is $T = 10,000$ hours.

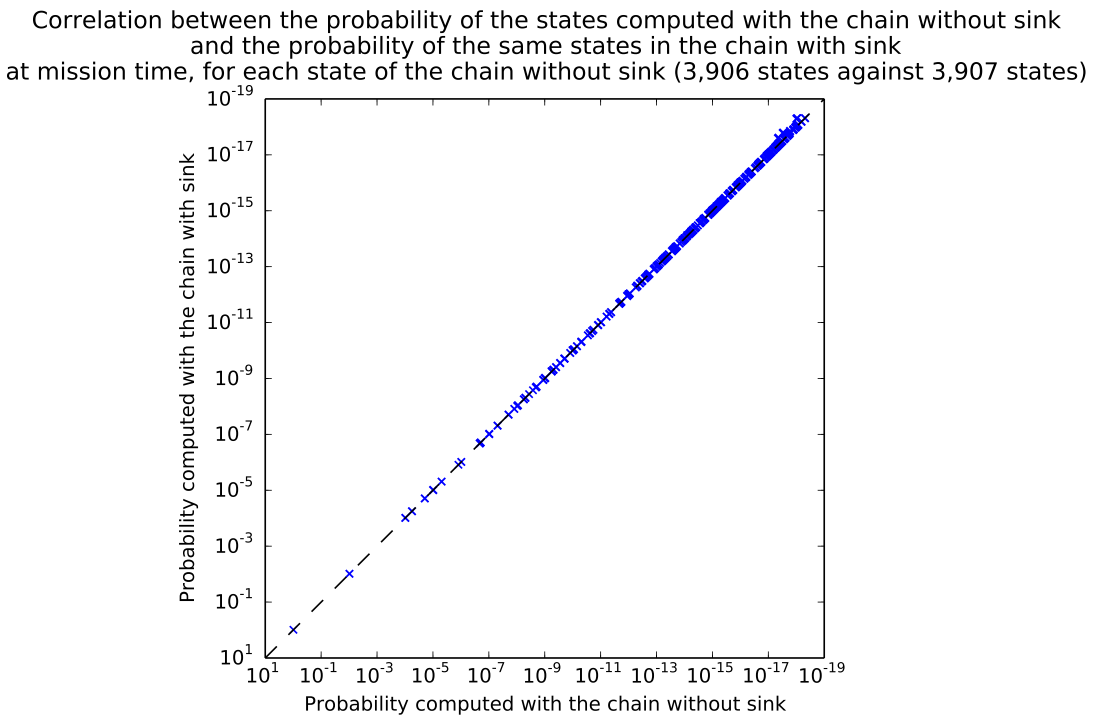


Figure 3.26.: Correlation between the probability of each state of the small chain ($\xi = 100$k transitions) whether they are in the chain with or without sink, for the stiff power supply, with mission time $T = 10,000$ hours.

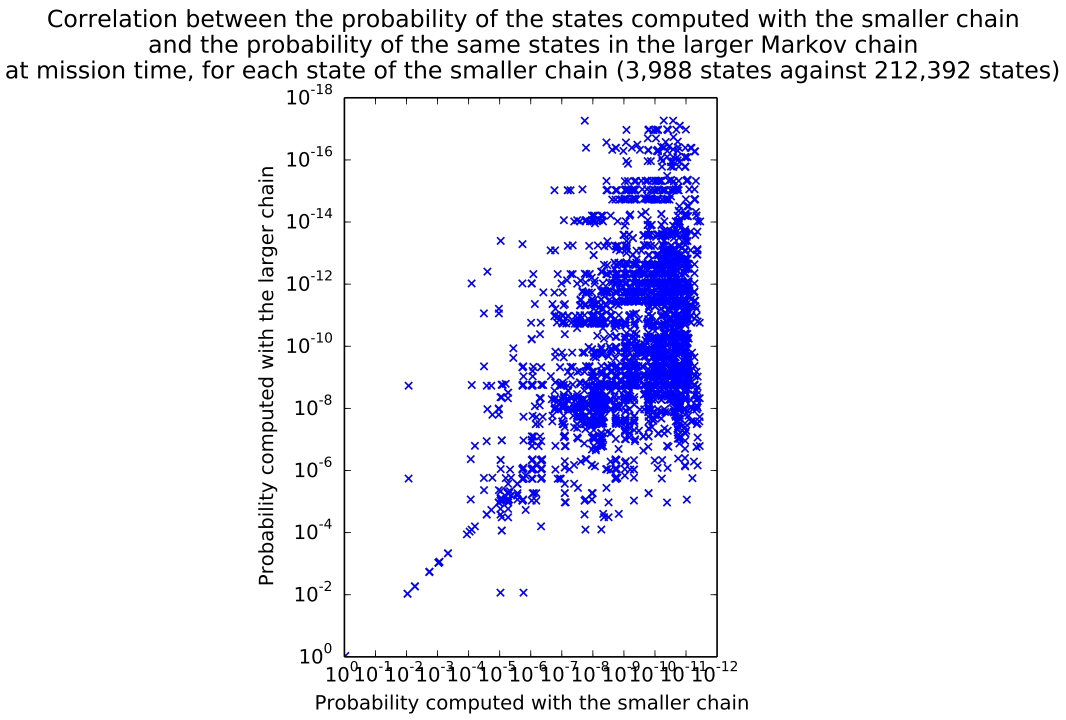


Figure 3.27.: Correlation between the probability of each state of the small chain whether they are in the small chain ($\xi = 100$k transitions) or in the large chain ($\xi = 5$M transitions), for the power supply, with mission time $T = 10,000$ hours.

It is interesting to see that the states change behavior, whether they are part of the small chain or not. Some probable states of the largest chain have probability around 10^{-2} in the large chain and only 10^{-6} in the small chain. Conversely, states with probability 10^{-2} in the small chain are less probable in the large chain (down to 10^{-9}). Only part of the states have the same probability in both chains. This shows that the exploration should use a higher threshold than $100,000$ states, as the confidence in the probabilities of the states is low.

Figure 3.28 shows the correlation for the stiff version of the system. The thresholds and mission time are identical to the previous case. In this case, the correlation is far better, and more states have the same probability in both chains. For the stiff power supply, the local behavior of the states in the small chain are correctly approximated, and the results should be more accurate.

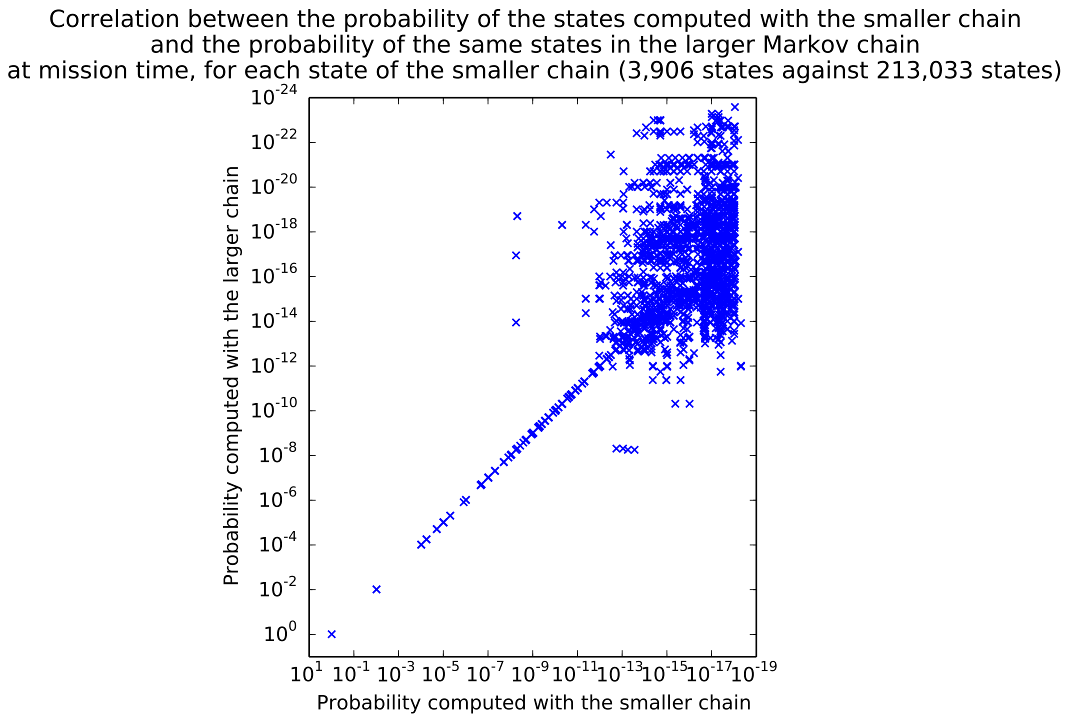


Figure 3.28.: Correlation between the probability of each state of the small chain whether they are in the small chain ($\xi = 100$k transitions) or in the large chain ($\xi = 5$M transitions), for the stiff power supply, with mission time $T = 10,000$ hours.

It is interesting to focus on the upper right border of the plot. Marks on the right border of the plot are states which have the lowest probability in the small Markov chain. These states are on the border of the chain, in the sense that they require many successive failures to be reached. Their probability change when they are part of the full chain (from 10^{-18} to between 10^{-23} and 10^{-12}). This means that adding more states to the chain changes their role, and changes their local behavior.

Figure 3.29 shows the correlation between the probabilities of the most probable states of two Markov chains with different sizes generated for the power supply. The small chain is obtained with threshold $\xi = 100,000$ and the large chain is obtained with $\xi = 5,000,000$. Both are generated without sink. A mark may represent different states in the two chains. States are sorted by their probability in both chains, so that the most probable states of the small chain can be compared to the most probable states of the other chain. The x-coordinate of a mark represents the probability of a state in the small chain and the y-coordinate represents the probability of a state in the large chain. Because the probabilities in the two chains can be correlated, this shows how a small chain is able to have, as a whole, the same behavior as the large chain. Moreover, this shows that only a few states are more probable in the large chain than what is selected in the small chain, even if border effects naturally lower the probability of the states on the borders of the small chain.

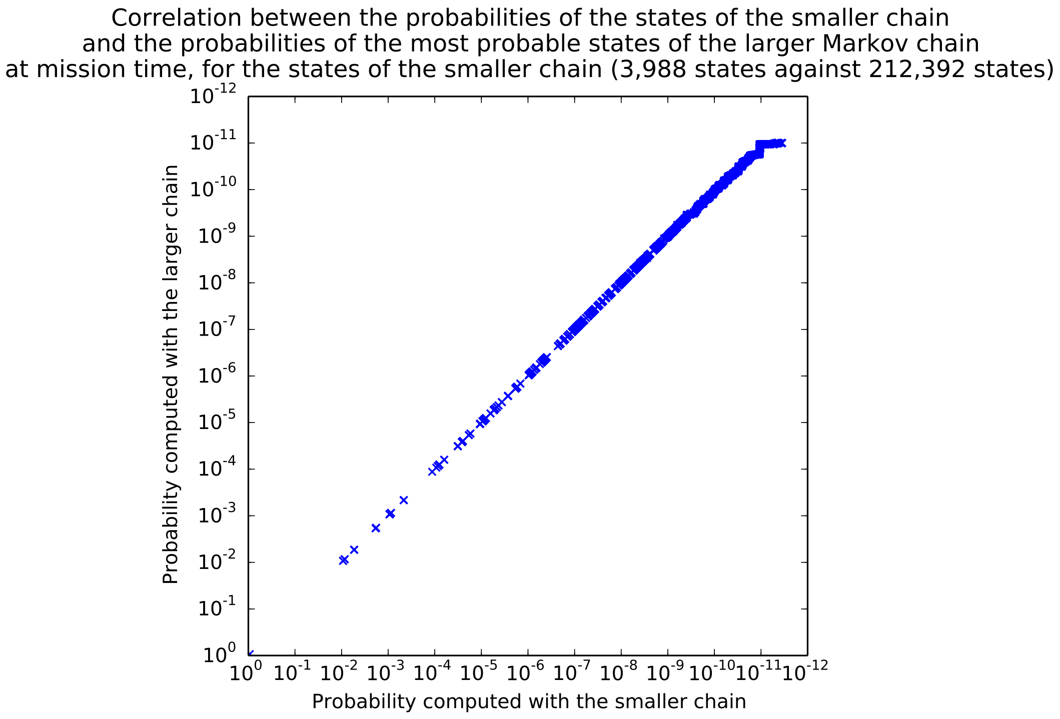


Figure 3.29.: Correlation between the probability of the most probable states of the small chain ($\xi = 100$k transitions) and those of the large chain ($\xi = 5$M transitions), for the power supply, with mission time $T = 10,000$ hours.

Figure 3.30 shows the same correlation for the stiff power supply. The results are slightly better for the stiff version of the model, and confirm that the approximation should be easier.

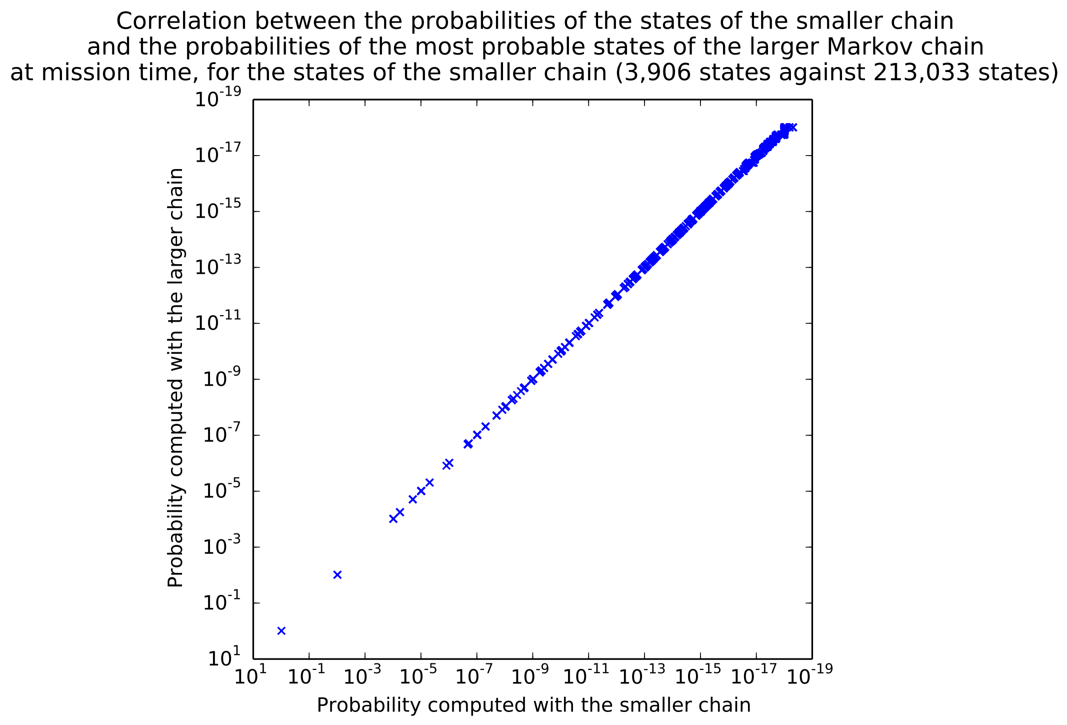


Figure 3.30.: Correlation between the probability of the most probable states of the small chain ($\xi = 100$k transitions) and those of the large chain ($\xi = 5$M transitions), for the stiff power supply, with mission time $T = 10,000$ hours.

3.2.5. Distributed Database

The distributed database system is a system which implements a database distributed over several disk clusters. The whole system is redundant and can be seen as two redundant databases in hot spares. Only the hardware architecture is modeled.

Description of the System

The distributed database is depicted in Figure 3.31. It is taken from [Muntz et al. 1989] where it was used to illustrate the aggregation technique on a system that cannot be completely generated and calculated. Numerical values are taken from [Muntz et al. 1989] for comparison purposes.

The role of the system is to implement a database which has redundant storage and redundant access to the data. There are 2 processor types (A and B), 2 sets of disk controllers with 2 controllers per set and 6 clusters of disks, each consisting of 4 disk units. Each set of controllers controls half of the 6 disk clusters. In a disk cluster, data is replicated so that one disk can fail without affecting the system. To do so, the "primary" data on a disk is replicated such that one third is on each of the other three disks in the same cluster. Thus one disk in each cluster can be inaccessible without losing access to the data. Each processor is linked to both sets of controllers so that each one has access to all data stored in the system. If processor A fails, it has a 0.10 probability of affecting processor B. There are 3 spare units for each processor. On occurrence of a failure of a processor, the unit is immediately replaced by a "hot spare" of the same type.

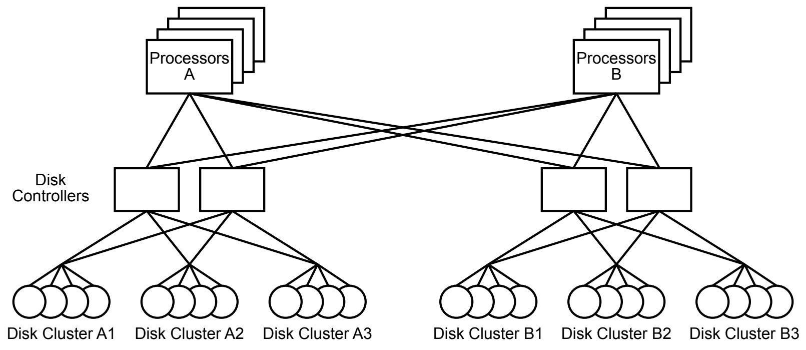


Figure 3.31.: A distributed architecture for a database system, as shown in [Muntz et al. 1989].

Components may fail and are repaired according to the rates given in Table 3.22. Each unit in the system has two failure modes, which occur with equal probability. The failure modes were used in [Muntz et al. 1989] to model hyperexponential repair distributions. The different failure rates for the disk units can be motivated as modeling the different usage of the units according to the type of data stored. Furthermore, different rates cut symmetries of the model.

Table 3.22.: Failure and repair rates for the distributed database, taken from [Muntz et al. 1989].

		Repair rates	
Component	Failure rates	mode 1	mode 2
Processors	1/2000 h^{-1}	1 h^{-1}	1/2 h^{-1}
Controllers	1/2000 h^{-1}	1 h^{-1}	1/2 h^{-1}
Disk set 1	1/6000 h^{-1}	1 h^{-1}	1/2 h^{-1}
Disk set 2	1/8000 h^{-1}	1 h^{-1}	1/2 h^{-1}
Disk set 3	1/10000 h^{-1}	1 h^{-1}	1/2 h^{-1}
Disk set 4	1/12000 h^{-1}	1 h^{-1}	1/2 h^{-1}
Disk set 5	1/14000 h^{-1}	1 h^{-1}	1/2 h^{-1}
Disk set 6	1/16000 h^{-1}	1 h^{-1}	1/2 h^{-1}

Components are repaired by a single repairman which chooses components at random from the set of failed units. The system is defined to be operational if all data is accessible to at least one of the two processors. This means that at least one processor, one controller in each set and 3 out of 4 disk units in each of the 6 disk clusters must be operational. We also assume that operational components continue to fail at the given rates when the system is down.

Results and Analysis

The complete Markov chain cannot be calculated for this system. As each component of the 36 components has two failure modes and one working mode, the number of states of the system is estimated to 3^{36} (1.5×10^{17}, which might be the number of grains of sand on earth).

The results of the partial exploration are shown in Table 3.23, and Table 3.24 shows the corresponding sizes of the chains and the calculation times. The reliability of the system is approximated to 1.9×10^{-5} at the chosen mission time $T = 100$ hours with a partial Markov chain made of 100 million transitions and 890 thousand states, and calculations took 18 hours. The partial exploration with sink barely gives the order of magnitude of the unavailability with 100 million transitions. However, the partial exploration without sink gives acceptable results with 1 million transitions, *if* the unavailability obtained with 100 million transitions is considered accurate.

The estimated unavailability is rather high (10^{-5}) and the state space is very large. This may explain why the partial generation is not very accurate, as the probability is scattered on more than $880,000$ states around the nominal states.

Table 3.23.: Measures of the unavailability Q of the distributed database for several thresholds on the size of the model, with given mission time of $T = 100$ hours.

	Partial chain without sink		Partial chain with sink		
Threshold ξ	Q	$\eta(T)$[a]	Q_{lb}	Q_{ub}	$\eta(T)$
$100,000,000$	1.928×10^{-5}	-	1.9257×10^{-5}	2.7986×10^{-5}	0.45332
$10,000,000$	1.8931×10^{-5}	0.018097	1.879×10^{-5}	0.00020572	9.9488
$1,000,000$	1.7848×10^{-5}	0.074244	1.7363×10^{-5}	0.0010596	60.027
$100,000$	9.7073×10^{-6}	0.4965	9.2189×10^{-6}	0.019199	2081.5
$10,000$	0	1	0	0.037886	-

[a] without sink, Q for the highest threshold is used as reference.

The order of magnitude of the results is different from what was given in [Muntz et al. 1989] (3.3×10^{-6}). There are many facts which might explain such difference. As stated for the fault tolerant database (see Section 3.2.1), the approximations made on the numerical values of the failure rates might change the value of the unavailability. Another source of difference is the dubious modeling of the hypothesis "a single repairman", which would need a closer look at the model. Using an unlimited number of repairers, the obtained unavailability was calculated around 8.2×10^{-6} with approximately the same relative error.

Table 3.24 shows the number of transitions and the number of states of the partial chains, as well as the calculation times. At the end of the algorithm which computes the partial chain without sink, transitions leading to candidates are removed (see Algorithm 2.3). For this system, 92% of the transitions are removed when the threshold $\xi = 100,000,000$ is reached.

This means that most of the transitions are leading to candidates at the end of the exploration. For the partial chain without sink, this means that most of the transitions are removed from the chain, and the behavior of the system is being approximated by a very little part of the whole system. For the partial chain with sink, this means that most of the transitions of the model are directed to the sink. This shows that the model is blooming when $\xi = 100,000,000$, as most discovered transitions are oriented towards the unexplored part of the model.

Table 3.24.: Sizes of the models of the distributed database and computation times for several thresholds on the size of the model, with given mission time of $T = 10$ hours.

	Partial chain without sink			Partial chain with sink	
Threshold ξ	States	Transitions	Times[a] (s)	Transitions	Times[a] (s)
$100,000,000$	$886,297$	$7,901,024$	$64,000 + 400$	$100,282,080$	$64,000 + 1,800$
$10,000,000$	$85,353$	$517,376$	$6,300 + 35$	$10,057,408$	$6,300 + 190$
$1,000,000$	$8,169$	$38,144$	$500 + 3$	$1,001,920$	$500 + 20$
$100,000$	857	$2,688$	$86 + 0$	$101,984$	$86 + 2$
$10,000$	105	432	$9 + 0$	$13,504$	$9 + 0$

[a] (generation of the chain) + (calculation of the chain).

The mission time was chosen at $T = 100$ hours, which may appear a very short time. As the time constant for this system is roughly $\frac{1}{\min \mu} \approx 2$ hours (see Section 2.2.2), the system is steady-state at the chosen mission time. Of course, choosing a more realistic mission time of $10,000$ hours would imply worse relative errors. However, as the bounds are sufficient to give the order of magnitude of the unavailability at $T = 100$ hours, the unavailability of the system could be computed at time $T = 10,000$ hours with the chain without sink.

The calculation times of the Markov chains are very low and inconsistent for this example. The time taken by the chosen calculation algorithm is proportional to the number of transitions and to the time-step of the algorithm, which is $\approx \frac{1}{\max \mu}$ (see Section 2.1.4). This time complexity is verified either within the partial chains without sink, or within the partial chains with sink. However, when $\xi = 100,000,000$, the size of chain with sink is 12 times larger than the chain without sink, but the calculation time is only 4 times longer. These calculations should be done again, in different conditions, to find the source of the inconsistency.

Figure 3.32 shows the evolution of the relative error obtained with partial Markov chains with sink, with respect to the threshold on the number of transitions. Each relative error was calculated at mission time of 100 hours. The relative error decreases as the size of the model increases. The overall evolution of the plot is similar to what can be seen on the other examples, but the relative error decreases more slowly. For this system, the number of transitions has to be multiplied by a hundred to divide the relative error by ten; for the power supply, the number of transitions has to be multiplied by two to divide the relative error by

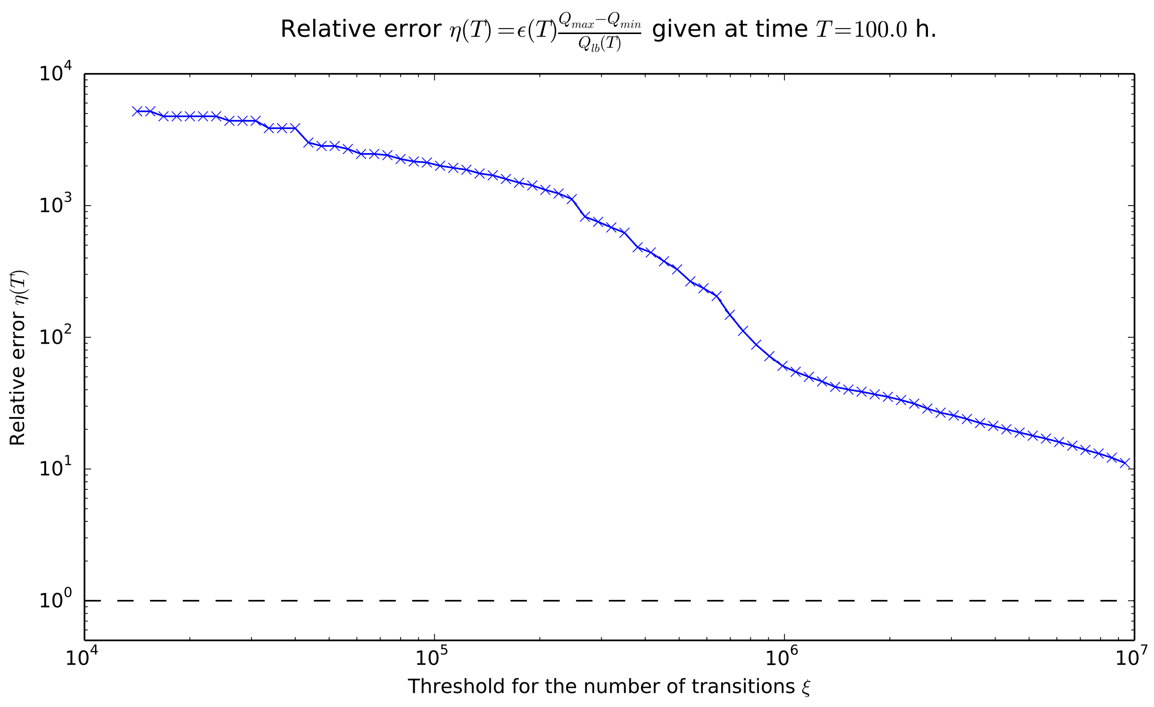


Figure 3.32.: The evolution of the relative error for the unavailability of the distributed database at time $T = 100$ hours with respect to the threshold on the number of transitions.

ten.

To further analyze the method, the relevance factor $R()$ used during the generation of the partial Markov chain is compared to the probability computed with the Markov chain without sink (the relevance factor is defined in Equation (2.10)). Figure 3.33 shows this correlation for each state of the distributed database in a Markov chain generated up to $\xi = 10,000,000$ transitions, at mission time $T = 100$ hours. Each mark on the graph is a state for which the x-coordinate is the relevance factor and the y-coordinate is the probability computed with the Markov chain. Note the logarithmic scales. The plot shows that the relevance factor seems to sort the states rather correctly. This corroborates the fact that the Markov chain without sink gives accurate results with this threshold.

It is interesting to note that symmetries in the architecture of the system appear on this plot, as there are many superposed marks, and blocks of slightly different marks. Most of the states have non negligible probabilities compared to the unavailability of the system, which may explain why the Markov chain with sink is not accurate for this threshold.

The effects of the sink on a Markov chain have been previously observed on the relative error. It is also interesting to observe the effect on the sink on each state of the chain. Figure 3.34 shows the correlation between the probabilities of states of partial Markov chains obtained with the same threshold $\xi = 1\text{M}$ transitions and at the same mission time $T = 100$ hours. One of the chains is obtained with the partial generation without sink, and the other

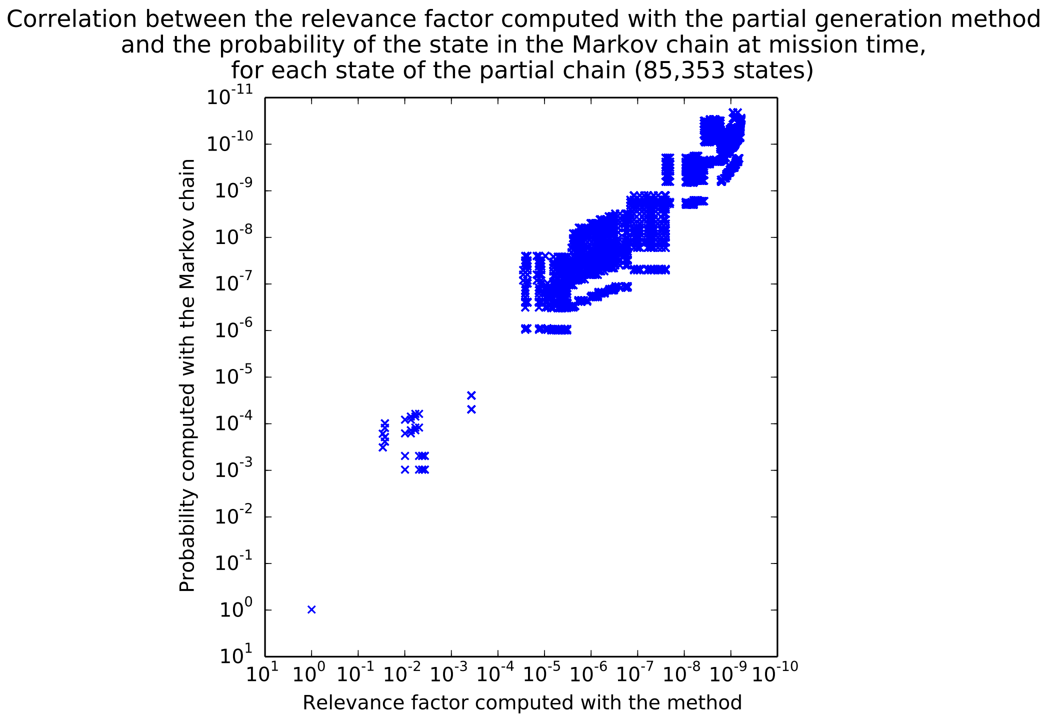


Figure 3.33.: Correlation between the relevance factor computed with the partial generation method and the probability with the Markov chain for each state of the partial chain for the distributed database, with threshold $\xi = 10\mathrm{M}$ transitions and mission time $T = 100$ hours.

is obtained with sink. Each mark is a state of the chain wihtout sink, which x-coordinate is its probability in the chain without sink, and y-coordinate its probability in the chain with sink. The sink has a probability of 1.0×10^{-3}, which gives a high relative error of 60.

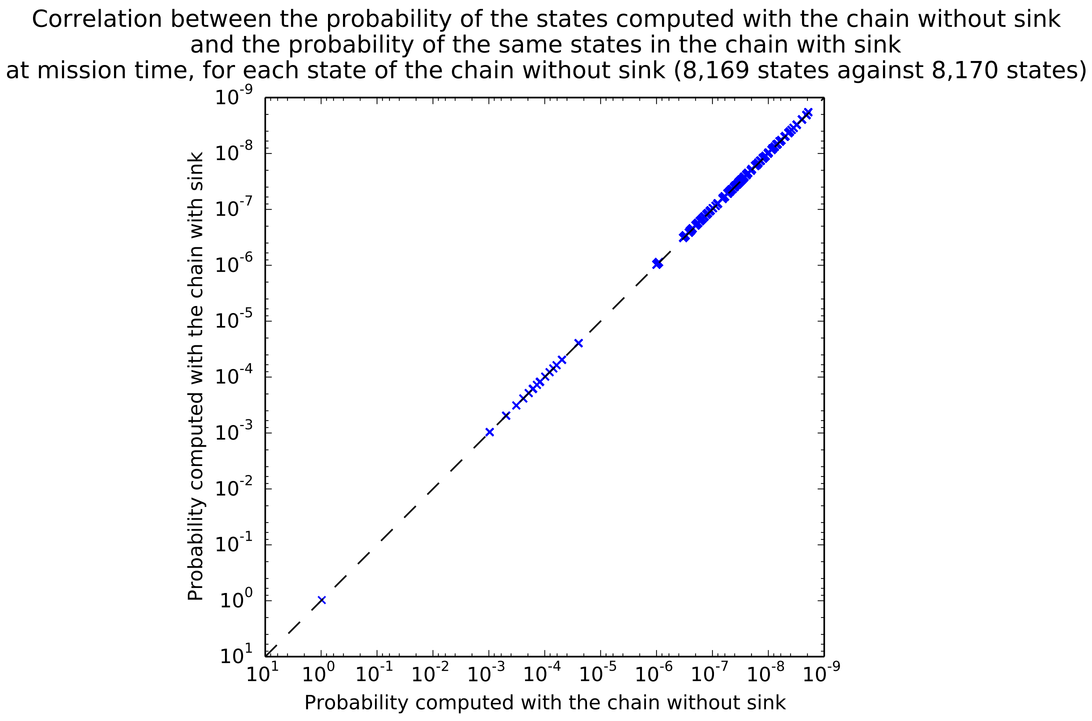


Figure 3.34.: Correlation between the probability of each state of the small chain ($\xi = 1$M transitions) whether they are in the chain with or without sink, for the distributed database, with mission time $T = 100$ hours.

On this figure, it is difficult to see where the probability in the sink comes from, as almost all the states have an equal probability in the chain without or with sink. This can be answered with the analysis of the model. The first transitions leading to the sink are transitions from a states which probability is around 4×10^{-4}. This state is in the 100 most probable states of the chain. All the other $8,000$ states also have transitions to the sink. The effect of the sink is almost not visible because the sink punctures a little amount of probability from almost each state of the chain. Hence, the living part of the chain has an accurate behavior compared to the expected behavior of the system, and the sink does not unbalance this behavior. This inspires confidence in the partial model without sink, which is bound to accurately approximate the behavior of the most probable states of the system, even if this result cannot be formally proven.

Another type of correlation can be drawn. This is the correlation between the probabilities of states of Markov chains obtained with different thresholds. The probability of each state of the small Markov chain is compared to the probability of the same state, in the bigger Markov chain.

Figure 3.35 shows this correlation. The thresholds are $\xi = 1,000,000$ transitions for the small chain and $\xi = 10,000,000$ transitions for the large chain. Both chains are generated without sink, and the mission time is $T = 100$ hours.

The probability of most states of the small chain does not change when they are part of the larger chain, which means that their local behavior is the same, whether they are part of the small chain or the bigger one. Both these chains without sink give accurate results, even if the reference for the error is the probability obtained with $\xi = 100,000,000$, which is not proved to be very accurate. Even if the two selected chains are “close” in their number of transitions, these results also inspire trustworthiness in the partial chains.

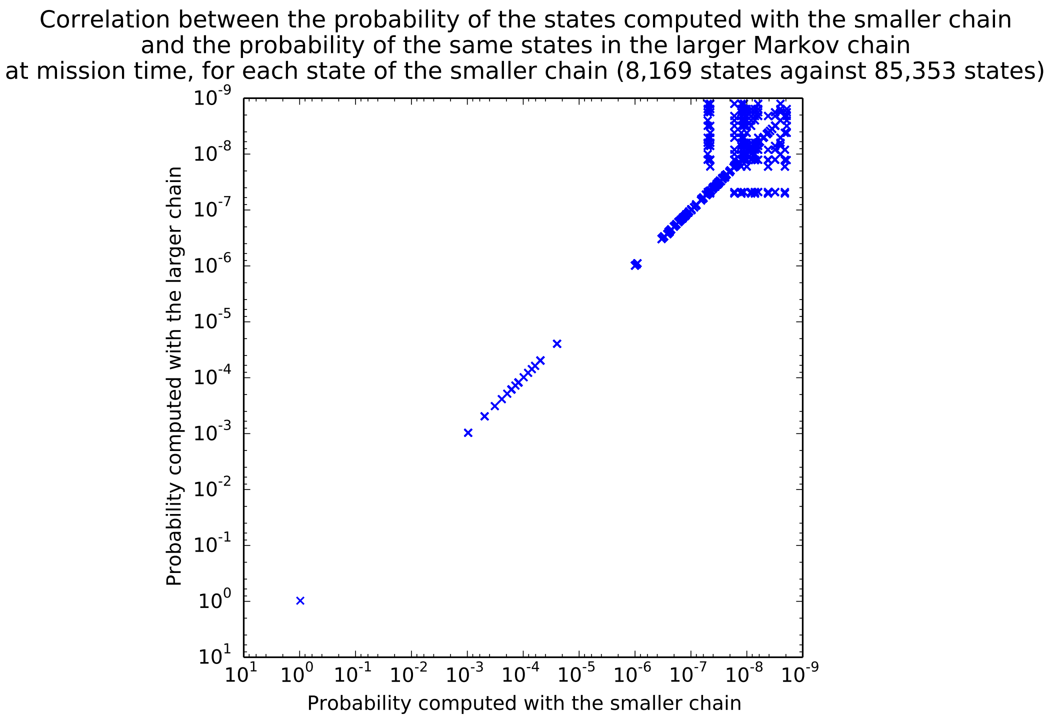


Figure 3.35.: Correlation between the probability of each state of the small chain whether they are in the small chain (ξ = 1M transitions) or in the large chain (ξ = 10M transitions), for the distributed database, with mission time $T = 100$ hours.

Figure 3.36 shows the correlation between the probabilities of the most probable states of two Markov chains with different sizes generated for the distributed database. The small chain is obtained with threshold $\xi = 1,000,000$ transitions, and the large chain is obtained with $\xi = 10,000,000$. Both chains are generated without sink. A mark may represent different states in the two chains. States are sorted by their probability in both chains, so that the most probable states of the small chain can be compared to the most probable states of the large chain. The x-coordinate of a mark represents the probability of a state in the small chain, and the y-coordinate represents the probability of a state in the large chain. Because the probabilities in the two chains can be correlated, this shows how the small chain is able to have, as a whole, the same behavior as the large chain. Moreover, this shows that only a few states are more probable in the large chain than what is selected in the small chain, even if border effects naturally lower the probability of the states on the borders of the small chain.

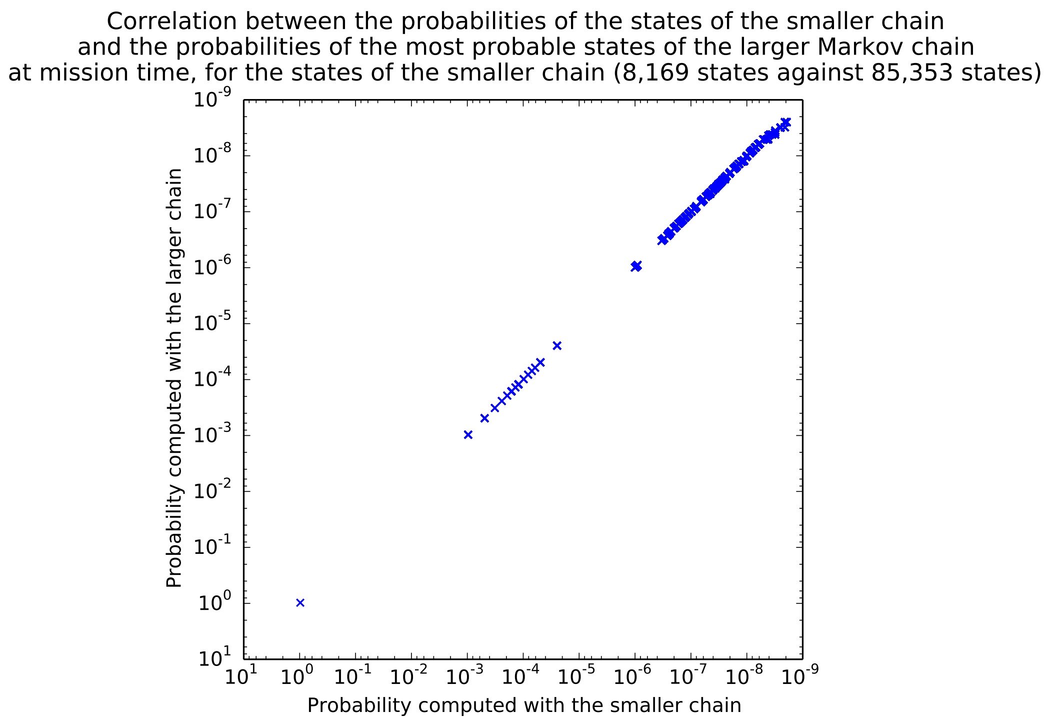


Figure 3.36.: Correlation between the probability of the most probable states of the small chain (ξ = 1M transitions) and those of the large chain (ξ = 10M transitions), for the distributed database, with mission time $T = 100$ hours.

Figure 3.37 plots the evolution of the relative error for the two versions of the database. The first version is the distributed database, and the other version is the version with an unlimited numbers of repairers, which was used at the beginning of this section. The two versions of the system give similar values of the relative error on the unavailability. However, the latter version gives slightly worse results for small Markov chains, and slightly better results for large Markov chains.

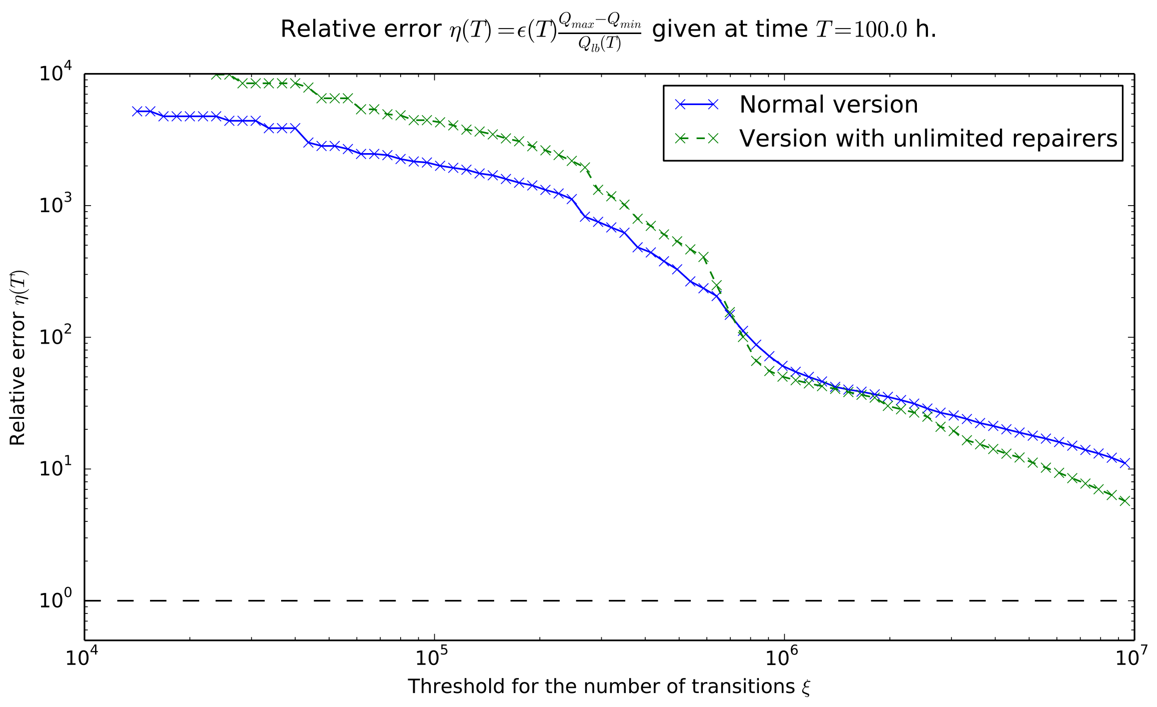


Figure 3.37.: Comparison of the evolutions of the relative error for the unavailability of two versions of the distributed database at time $T = 100$ hours with respect to the threshold on the number of transitions..

3.3. Teachings of the Examples

The various examples were chosen to illustrate the practicability of the method and its performances in many different situations. Some guidelines are drawn from these examples.

The most accurate results are always obtained with the Markov chain without sink with the highest possible number of transitions. It is not possible to quantify the error made in a partial Markov chain without sink. The Markov chain with sink gives bounds on the calculated safety indicator but these bounds are sensitive to the chosen mission time of the system under study.

The partial generation method is most efficient for highly repairable systems, because these systems have a few states in which they live most of the time. In these systems, the high competition between failure rates and repair rates ensure that the selection of the most probable states is effective. The performance indicators of a system are more easily calculated than the safety indicators, as their order of magnitude is usually higher, and their highest value is reached in the state with few failed components.

The different correlation between the probabilities of the states within various Markov chains lead to the following conclusions:

- The relevance factor can only be correlated to the probability of the states for systems which have a single nominal state (this is not the case for non-repairable systems).

- The relevance factor cannot estimate exactly which states are the most probable, but this has only consequences with states that are on the border after the exploration. The core layers of the chain are correctly selected. As a consequence, the partial chains without sink reproduce correctly the overall behavior of larger chains.
- The sink is an absorbing state which draws probability from the states of the partial chain. The states that are most drained are the states on the border of the chain, but the sink can also be connected to the core of the chain. In the latter case, the Markov chain with sink will provide rough bounds on the performance indicators.

It is important to keep in mind that both the generation and calculation times are almost linear in the number of transitions. This means that there is no overhead in using the partial generation method instead of a simpler generation method. Moreover, this means that Markov chains of medium sizes are not relevant and the largest chain should be computed first.

4. Prototype and Tools

In this chapter, the developed tools are presented. AltaRica 3.0 was chosen as the high level language from which the Markov chain is generated (see Chapter 1). The idea behind this chapter is to show the software choices that were made and to illustrate the practicability of the method.

The main tool, `GTSLimMark`, has been developed to test the method, assert its validity and soundness, as well as its performances, and analyze its domain of validity. It is now part of the work of the AltaRica Association, which provides `C++` tools and libraries for AltaRica 3.0. `GTSLimMark` works at the Guarded Transition Systems (GTS) level, as GTS models bear the semantics of the AltaRica 3.0 models.

Another tool was developed, `PyGTS`, to manage the experiments. Its goals are to use the AltaRica 3.0 tool-chain on different models, test different versions of `GTSLimMark` (such as different relevance factor), process the results, store them, and make them analyzable.

The remainder of this chapter is organized as follows. Section 4.1 gives an overview of the steps to calculate the indicators of interest of a system from its AltaRica model. Section 4.2 presents the main tool, `GTSLimMark`, analyzes its time and memory complexity, and presents different practical problems. Section 4.3 presents the project management tool, `PyGTS`. Section 4.4 analyzes the performance bottlenecks of the tools.

4.1. Overview of the Tool-chain

The tool-chain is illustrated in Figure 4.1. The AltaRica model is given; its construction from a system is out of the scope of this thesis (see e.g. [Mortada et al. 2014] [Prosvirnova 2014]).

The tool developed in this PhD is part of the tools developed in the AltaRica Association. The whole AltaRica 3.0 project has more than 150 classes, from basic containers to the GTS animation. The tools run on Windows and Linux, on 32 or 64 bits architectures.

First, the AltaRica model is flattened into a GTS model (see [Prosvirnova and Rauzy 2012]). Second, `GTSLimMark` animates the GTS model to explore the state space of the system. `GTSLimMark` implements both partial exploration methods (without or with sink). Finally, the Markov chain is calculated and reliability indicators are computed (see Section 2.1.4).

My goal is to animate the GTS and generate the reachability graph, which is then output as a Markov chain. To output the reachability graph as a Markov chain, nodes are transformed

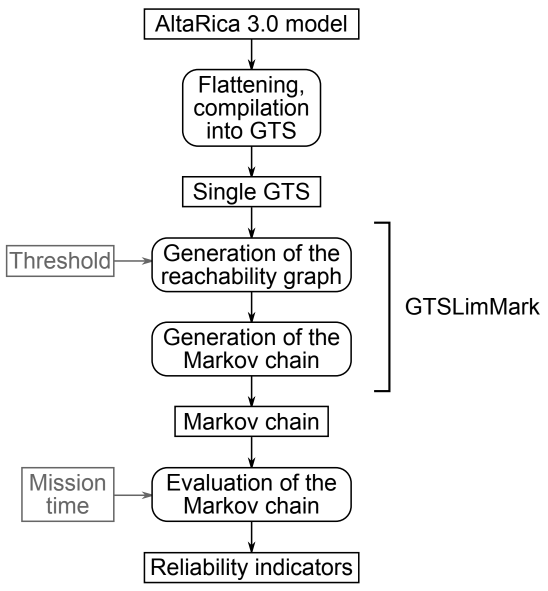

AltaRica 3.0 model
Flattening, compilation into GTS
Single GTS
Threshold
Generation of the reachability graph
Generation of the Markov chain
GTSLimMark
Markov chain
Mission time
Evaluation of the Markov chain
Reliability indicators

Figure 4.1.: The different operations needed to process a model

into states, and events labeling the transitions are used to find the transitions rates in the GTS model. A single call to `GTSLimMark` can output up to two Markov chains: without or with the sink using respectively Algorithms 2.3 and 2.4.

The algorithms used to compute the Markov models are developed in Section 2.1.4. We used the Fast Euler Method (FEM) and the Exponentiation method (EXP), because these are respectively fast and reliable. As the accuracy of the results is not of paramount importance (see Figure 2.9), FEM was used to assess the experiments. The Markov chain solver can output transient and mean probabilities of the states, sojourn times, expected transient reward, and expected mean rewards.

In AltaRica, there is the concept of observers, which are expressions over the variables of the system that can be evaluated for each state of the model. They are different from flow variables because they cannot be used in the computation of a variable of the system. Typical observers are the unavailability of a system, or its production rate. Numeric observers are propagated to the Markov chain and are expressed as rewards. The Markov chain calculator outputs the transient and mean values for these observers.

4.2. `GTSLimMark`: Partial Generation of the Reachability Graph

`GTSLimMark` is written in `C++`, it consists in a dozen classes.

The generation of the reachability graph is the key element of the method. Its principle is detailed in Section 2.3. Here are recalled some notations: let $\langle V, E, T, A, \iota\rangle$ be a GTS and $\Gamma = \langle \Sigma, \Theta \rangle$ be its reachability graph, C be the set of candidates, Σ be the set of nodes of the reachability graph, Θ be the set of edges of the reachability graph, ι be the source node, $R(\sigma)$

be the calculated relevance factor between the source node and $\sigma \in \Sigma$, and $l(\sigma \to \tau)$ be the length of the edge between the nodes σ and τ.

The variation of Dijkstra's algorithm can be summarized as shown in Algorithm 4.1.

Algorithm 4.1: Dijkstra's algorithm, reminder of Algorithm 2.1

```
1  while $C \neq \emptyset$ do
2      let $\sigma$ be the candidate with the highest $R$ in $C$
3      remove $\sigma$ from $C$, add it to $\Sigma$
4      foreach successor $\tau$ of $\sigma$ do
5          if $\tau \in C$ and $R(\tau) > R(\sigma) + l(\sigma \to \tau)$ then
6              $R(\tau)$ is raised to $R(\sigma) + l(\sigma \to \tau)$
7          else if $\tau \notin C$ and $\tau \notin \Sigma$ then
8              $\tau$ is added to candidates
9              $R(\tau)$ is set to $R(\sigma) + l(\sigma \to \tau)$
10         transition $\sigma \to \tau$ is recorded in $\Theta$
```

Denoting $|\Theta|$ the number of edges in the reachability graph (the number of transitions in the system), and $|\Sigma|$ the number of nodes (the number of states in which the system can be), Dijkstra's algorithm can run in $\mathcal{O}(|\Theta| + |\Sigma| \times \log |\Sigma|)$ in time with optimal containers, as shown by [Fredman and Tarjan 1984], using Strict Fibonacci Heaps for instance (see [Brodal et al. 2012]). We aim for $\mathcal{O}(|\Theta| \times \log_2 |\Sigma|)$, using less optimized but more convenient containers. The critical operations are lines 5–10, because they are done $|\Theta|$ times. Hence, the following properties must be verified by the data sets to achieve $\mathcal{O}(|\Theta| \times \log_2 |\Sigma|)$ time ($|C| \leq |\Sigma_{final}|$ because all states from Σ were candidate first):

P.1 Retrieving an already existing candidate or explored state takes $\mathcal{O}(\log_2 n)$ time, where n is the size of the considered set.

P.2 Raising the relevance factor of an already stored candidate is $\mathcal{O}(\log_2 |C|)$.

P.3 Adding a candidate is $\mathcal{O}(\log_2 |C|)$.

P.4 Calculating $l(\sigma \to \tau)$ takes at most $\mathcal{O}(\log_2 |\Sigma|)$.

P.5 Adding an edge takes at most $\mathcal{O}(\log_2 |\Sigma|)$.

Least critical instructions are lines 2–4. They are executed $|\Sigma|$ times.

P.6 Getting the closest candidate is $\mathcal{O}(\log_2 |C|)$.

P.7 Removing a candidate is $\mathcal{O}(\log_2 |C|)$.

P.8 Adding a state to Σ is $\mathcal{O}(\log_2 |\Sigma|)$.

Worst case time for the calculation of the successors of σ (line 4) is not easily predictable and depends on the GTS model. As the size of the GTS model is always limited compared to the size of the reachability graph, this is not a problem.

In practice, the following rule of thumb can be applied to the safety related models: in each state, there is a possible evolution for each variable of the GTS model. Hence $|\Theta| \approx |V| \times |\Sigma|$, where $|V|$ is the number of variables of the GTS. When variables are Boolean, $|\Sigma| \approx 2^{|V|}$.

4.2.1. Data Structures

The chosen data structures are detailed here for nodes σ, transitions $(e : \sigma \to \tau)$, and the data sets Σ and Θ.

A node of Σ is characterized by the values of the variables of the GTS. Hence $\sigma \in \Sigma$ is represented in memory as a vector of assignments on V: a typed value is associated to each variable of the GTS. Available types for the variables are: integer, floating-point decimal, string literals, or enumerations (enumerations are a set of user-specified constants). These types are easily ordered (enumerations can be translated to integer constants), so nodes of the reachability graph can be distinguished and sorted using the values of the variables.

Properties P.1, P.3, P.7, and P.8 are easily satisfied with an AVL tree: insertions, searches, and deletions are $\mathcal{O}(\log_2 n)$ where n is the number of elements in the tree. Note that insertions and searches in an AVL tree are done by ordering its nodes. The order relation in the AVL tree is based on the values of the variables in each node. It cannot be the relevance factor of the node, as this quantity may vary for candidates (property P.2). Consequently, it is not possible to satisfy property P.6 with this AVL tree, so the set of candidates must use another data structure. The set of explored nodes Σ is encoded with an AVL tree.

Binary heaps are a good structure to sort the states. Insertions and deletions are $\mathcal{O}(\log_2 n)$ where n is the number of elements in the heap and retrieving the highest priority element is $\mathcal{O}(1)$, so properties P.3, P.6 and P.7 are satisfied. Binary heaps can be seen as priority queues, with an interesting ability: the priority of an element can be raised while keeping the heap sorted in $\mathcal{O}(\log_2 n)$ operations (property P.2). Note that searches for a specific node in a binary heap are $\mathcal{O}(n)$ so a binary heap cannot be used alone for the set of candidates (property P.1).

In the current implementation of `GTSLimMark`, the set of candidates C is encoded by an AVL tree backed by a binary heap. The candidates are encoded by a list of typed values to store σ and a real which stores $R(\sigma)$. The reachability graph is encoded by an adjacency list, as it already existed in the AltaRica project and satisfied property P.5, whose nodes are $\sigma \in \Sigma$, and edges are the transitions labeled by the values of the transition rates.

Finally, the property P.4 is verified with the chosen relevance factor in Section 2.3.2 equation (2.10). To calculate the length of the transition $\sigma \to \tau$, the sum $\sum_k \lambda_{\sigma\to k}$ is computed right after the selection of the best candidate σ line 2, and $l(\sigma \to \tau) = \frac{\lambda_{\sigma\to\tau}}{\sum_k \lambda_{\sigma\to k}}$ is computed inside the loop line 4. This provides the calculation of all the lengths of all the transitions with only 2 traversals of all the transitions, and takes $\mathcal{O}(|\Theta|)$ time.

The times can be measured and compared to the theoretical complexity. This is done on some examples in Section 4.3.5.

Memory Optimizations

The states and the transitions may occupy a similar space in memory. For instance, with Boolean systems: $|\Sigma|$ is $\mathcal{O}(|\sigma| \times 2^{|V|})$ where $|\sigma|$ is $\mathcal{O}(|V|)$, so $|\Sigma|$ is $\mathcal{O}(|V| \times 2^{|V|})$, and $|\Theta|$ is $\mathcal{O}(|\sigma \to \tau| \times |V| \times 2^{|V|})$ where $|\sigma \to \tau|$ is $\mathcal{O}(1)$. They both need to be optimized, in order to explore the largest systems. In practice, Σ is larger than Θ.

There are multiple ways to improve the efficiency of the storage of the nodes $\sigma \in \Sigma$. Two of these solutions are: the compression of each node, or the compression of the whole set of nodes.

The first approach consists in compressing each node so that it takes less space in memory. We recall that a state is a vector of typed values (so-called variants), one for each variable of the GTS. These variants take a lot of space in memory. This is because they are composed of a type and a value. As the value can be of several basic types, it is hardcoded in an `union`. Hence, the size of a variant in memory is rather large (in practice, 24 bytes). One variant is created for each variable of each node of Σ, thus many values that are of the same type and that have the same value are duplicated (for instance, the Boolean values `True` and `False`). The idea is to create a factory which will create and index the different variants, as they are needed. Thus, a node can be expressed as an array of indexes instead of an array of variants. The size of the index is chosen before the compilation of `GTSLimMark`. It is currently set to 1 byte, which enables 256 different variants, which was enough for all our tests so far. The factory uses a hash table to store the already created variants. For small binary models where there are only two different variants, the memory used has been measured up to 18 times lower with the factory. The overhead of the factory is largely amortized for bigger systems.

The second approach consists in compressing the whole set of nodes. This way has not been explored yet, but one can imagine containers similar to Binary Decision Diagram (BDD) to encode the set Σ. The difficulty is obviously to encode the nodes in a decision diagram for variables of various types.

The transitions are currently implemented as a doubly linked list of outgoing transitions for each node. A simpler implementation would reduce the transition to 3 pointers (the source node, the destination node, and the event of the GTS model).

4.2.2. Initial State(s)

The GTS model provides the initial state ι (or initial assignment). In this state, the variables usually have their default value. The initialization is not given in Algorithm 4.1 because it only consists in putting ι in C with $R(\iota) = 1$, before the execution of the algorithm. As $R(\iota) = 1$, the choice of ι is important for the exploration, and ι should be the most probable state of the system (see Section 2.3.1).

It could be interesting to add expert knowledge to some GTS models and begin the generation of the reachability graph with several initial states. Having one or more initial states is not a problem: they can be added to C before the execution of the algorithm. The $R(\sigma)$ value used for these multiple initial states has to be chosen according to expert knowledge.

4.2.3. Using Synchronizations of AltaRica 3.0 Models

Synchronizations are a very useful modeling mechanism in AltaRica. They are used for instance to model repair teams, and synchronize the beginning of a repair with the allocation of an available repairman. However, their inherent behavior induces "immediate" events in the GTS model.

Immediate events are represented in the Markov chain by infinite transition rates, which leads to the concept of instantaneous states [Chung 1960] [Anderson 1991]. It is not possible to cut the outgoing transitions of instantaneous states, because that would unbalance the model. Moreover, the removal of all the outgoing transitions of an instantaneous state would transform it into an absorbing state.

However, instantaneous states can easily be suppressed, as shown in Figure 4.2. Note that concurrent instantaneous outgoing transitions must all have a probability γ, which is the relative probability to take a transition instead of the others. The sum of the outgoing probabilities of infinite transitions must be 1. This latter condition can be relaxed by dividing each probability by the sum of the outgoing probabilities.

As shown in Figure 4.2, each incoming non-instantaneous transition with rate λ of a state with at least one instantaneous outgoing transition is transformed into non-instantaneous transitions with rate $\lambda \times \gamma_i$ leading to each state that are reachable through an instantaneous transition with probability γ_i.

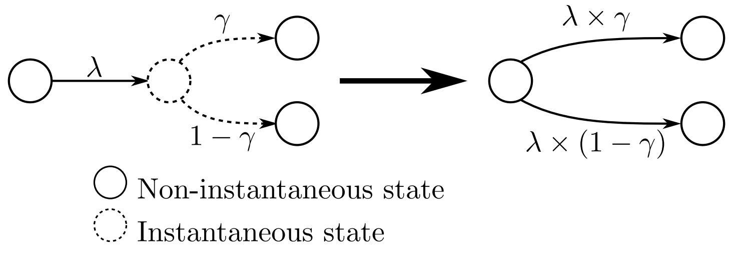


Figure 4.2.: Removing instantaneous states from Markov models.

The choice was made to expand the instantaneous states during the exploration. Each time an instantaneous state is encountered during the exploration (by finding an instantaneous outgoing transition of τ, which is done after line 8 of Algorithm 2.2), it is expanded instead of being considered as a sole candidate. Reachable states from an instantaneous states are also expanded if necessary before being added to candidates.

The number of equivalent transitions for each instantaneous state is the number of paths from the instantaneous state to all the closest non-instantaneous states. This may lead to several cascaded expansions, as well as useless expansions of the same instantaneous state that may be encountered several times during the exploration. This number is prone to combinatorial explosion, and affects the worst-case time of the partial generation method in unpredictable ways. However, in the conducted experiments, it was not significant (see Section 4.3.5, where the computation time is measured for some models with a repair team). More sophisticated expansions could be developed to tackle this problem.

Another side effect is the multiplication of transitions between some states, because there may be several different paths between two non-instantaneous states reachable through instantaneous states.

To avoid such useless computations, the data structures may be modified to store all the incoming and outgoing transitions of all instantaneous states (candidates and explored). Thus, it would be possible to expand instantaneous states, at lower cost, when the exploration is finished, and to avoid multiple transitions between the same states. This is a tradeoff between memory and computation time.

4.2.4. Limiting the Exploration

As discussed in Section 2.3.3, the limiting criterion for the exploration $|\Gamma_S| < \mathbb{S}$ (line 3 of Algorithm 2.2) can be of various natures. Some practical thresholds are:

- The number of transitions ξ,
- The number of states,
- The in-memory size of the reachability graph.

The real concern is the memory and time that can be allocated to the exploration. Hence, the most convenient threshold is the number of transitions ξ, as it is correlated to both the memory size of the reachability graph and to the time taken by the computation of the Markov model.

It would be possible to encode other thresholds, such as the number of states (which is a more convenient measure for safety engineers), or the actual in-memory size of the objects.

The relevance factor of the state would have been a convenient threshold, but it should not be used as a threshold. Some tests were conducted and the relevance factor for the last explored state does not indicate whether the evaluation will be accurate or not. For the computing modules in Section 3.2.2, the relevance factor of the last explored state keeping 343 transitions is 1.7×10^{-3}, which gives 0.60% error at mission time. For the oil production system in Section 3.2.3, the relevance factor of the last explored state keeping $173,656$ transitions is 6.6×10^{-8}, which gives 0.32% error at mission time.

4.2.5. Remaining Candidates

As seen in Algorithm 2.2, after the exploration is stopped, the remaining candidates must be dealt with. The obvious and straightforward way to deal with the remaining candidates is to ignore them. However, the relevance factor $R(\sigma)$ for the remaining candidates may lead to situations where packs of candidates C_{R_l} have the exact same relevance factor as the last explored state $R_l = R(\sigma_{last})$. This is because lots of the transitions in the reachability graph have the same rates (the number of events of the GTS is very low compared to $|\Theta|$), and because of numerical approximations.

It is rather illegitimate to keep some candidates that have a relevance factor R_l and discard some others that have the same relevance factor, because it implies that the algorithm chooses some candidates in the pack C_{R_l} in a random manner. In this case, two runs of `GTSLimMark` on the same model with the same threshold *may* lead to different results.

Thus, the exploration is continued after the limiting threshold is reached, until there are no more candidates with the same $R(\sigma) = R_l$. This leads to slightly larger models. Both endings (with or without keeping the candidates with the same $R(\sigma) = R_l$) were tested and the numerical fluctuations are, in practice, negligible.

4.3. `PyGTS`: Project and Result Management

Here are described developments done for `PyGTS`, a tool which is used to animate the AltaRica tool-chain. Its main goal is to automatize repetitive actions such as the numerous calls to the tools in the AltaRica tool-chain, timing measurement, and extraction and storage of the results.

4.3.1. The `PyGTS` Package

`Python 3` was chosen for this tool, because of its flexibility. `PyGTS` is delivered as a pure `Python` package. Figure 4.3 shows the different classes used to build `PyGTS`. `PyGTS` is organized around 4 parts which are:

- the models, which are `Python` representations of the AltaRica objects (`GTS`, `RG`, `State`),
- the `Study` and `StudyCase`, which are interfaces to the tools, and provide parsers for the intermediate models and parsers for the output of the Markov chain calculator,
- the Data Access Objects (`DAO*`) which uses the `SQLite` module of `Python` to store and access results of the various experiments in databases,
- the `Queue` and `Task`, which handles multitasking.

The Markov model parser and `Python` representation of the `RG` are mainly used for debug purposes, as graphical representation of Markov chains quickly become unreadable with large chains. The `PyGTS` can draw small Markov models (see for instance Figure 3.2, where only 419

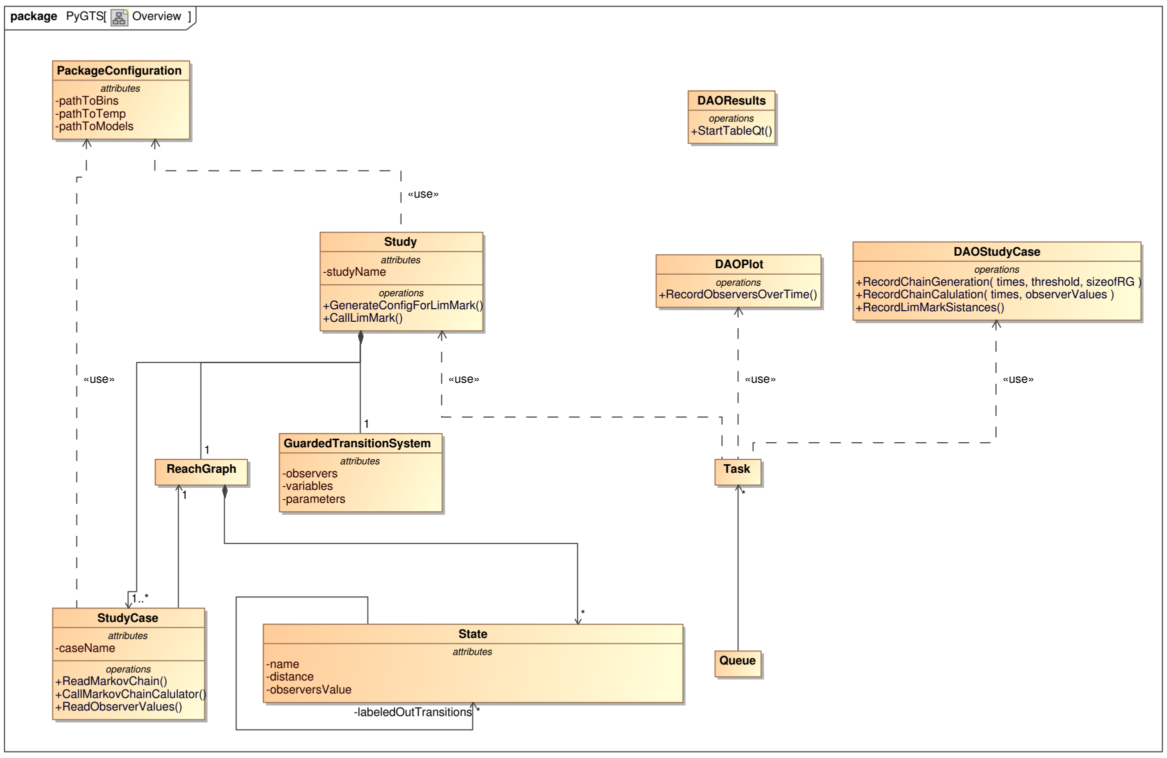


Figure 4.3.: Working on PyGTS classes.

states are represented). However, it enables interesting features such as the comparison of the probability to be in each state with the relevance factor (see for instance the oil production system Section 3.2.3).

A `Study` denotes a single study of the model. The main parameters of the studies are the threshold on the number of transitions and the parameters for the Markov chain calculator: the mission time, the calculation algorithm, and the main parameter for the calculation algorithm (called ρ, see Section 2.1.4). A single study may lead to multiple Markov chains, as the generation algorithm generates both chains without or with sink in the same run. A `StudyCase` bears these informations. There are 2 types of useful `StudyCase`: the “partial”, and the “partial with sink”.

To compute the probability to be in the sink, a special observer must be included in the initial AltaRica model and its value must be 0.0 in all states (there is no theoretical limitation behind this, and this may be relaxed in future implementations of the tool-chain). The value of this observer is adjusted to 1.0 in the sink state, so that the probability to be in the sink is automatically output by the Markov chain evaluator.

Figure 4.4 shows the different steps that are handled by PyGTS from the AltaRica model to the storage of the results in the database. These steps are repeated for each study, i.e. for each threshold.

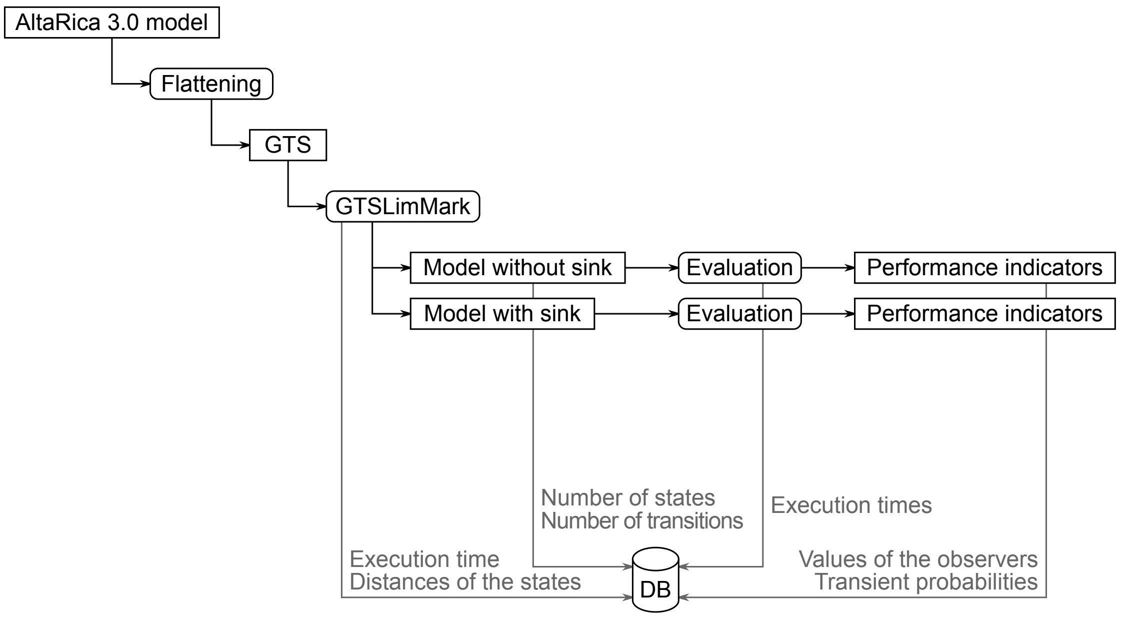


Figure 4.4.: Typical workflow with `PyGTS`.

4.3.2. Database

The tables recording the results are shown in Figure 4.5. The current implementation of the database may be optimized and normalized. In practice, there is one database for each example. In the current implementation, in `ResultsForCaseStudy` there is one entry for each observer of the GTS whose value is to be recorded at mission time.

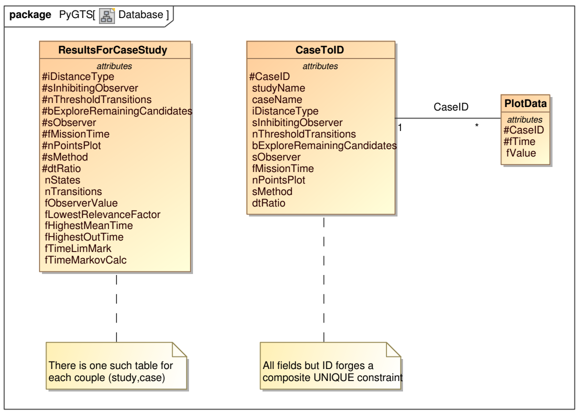


Figure 4.5.: Working with `PyGTS`. Fields prefixed with # are forming a primary key.

In some specific cases, it is interesting to plot the transient evolution of the value of the observers of the model over time. The value of each observer at each time step is recorded in the `PlotData` table.

To enable the correlations (such as Figure 3.15), the relevance factor associated to each state is read, as well as its probability at mission time, for each case study.

For the experiments developed in this thesis, there is more than $5,400$ entries in ResultsForCaseStudy, more than $550,000$ transient values of observers, and $4,200,000$ state distances and probabilities.

4.3.3. Multitasking

The PyGTS takes advantage of the multiprocessing module in Python and of the database to compute multiple StudyCase at once. This is necessary, as the total cumulative computation time taken for the examples is roughly $1,200$ hours. Queuing is done to limit the number of parallel studies on the same computer. This is necessary to balance the memory and computation capabilities, so that measured times are realistic (any operating system allows 100 studies to be launched at once, even if octo-core processors can only treat 8 of them concurrently).

The main queue bears the tasks, which are unstacked and launched in separate processes. There are atomic tasks dedicated to part the workflow in PyGTS. There are 4 kind of elementary tasks:

1. Flatten the AltaRica model and call GTSLimMark with the correct parameters,
2. Read the Markov chain to extract its size,
3. Call MarkovChainCalc and parse its output to record the transient value of the observers and/or of the probabilities,
4. Remove the intermediate models.

4.3.4. Handling Results

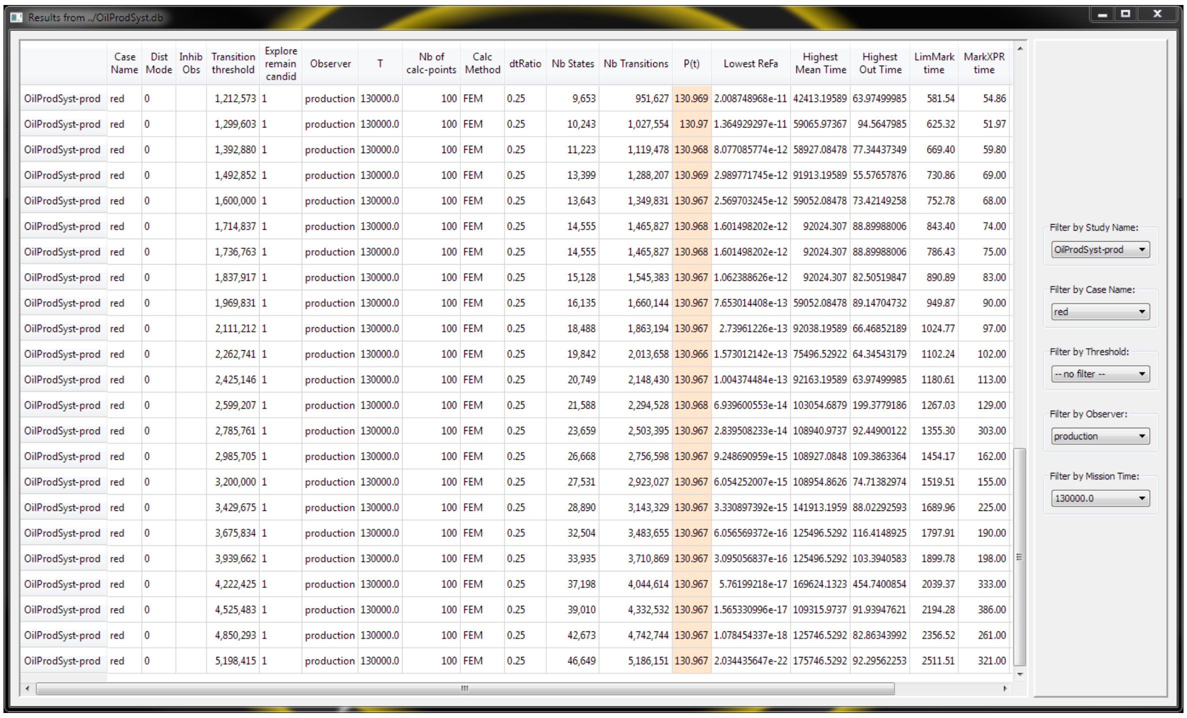
Python provides programming interfaces for several interesting tools, such as matplotlib and PyQt. matplotlib is used to programmatically plot data. PyQt is used to show an interactive table which displays the results as soon as they are available.

Figure 4.6 shows an interface to the database, which was done with PyQt. It is useful to have an overview over the data before extracting the interesting data. Filters are used to select the interesting point of view through the thousands of entries of the different tables (there are more than $3,200$ entries for the sole oil production system).

All the plots and tables shown in Chapter 3 were generated with PyGTS with the help of matplotlib.

4.3.5. Evaluation of the Run-time Complexity

GTSLimMark and MarkovChainCalc are the two most time consuming tools. The Markov chain solver using the FEM method runs in worst-case $\mathcal{O}(|\Theta| \times \frac{T}{dt})$ time, where $|\Theta|$ is the number



Results from ../OilProdSyst.db

	Case Name	Dist Mode	Inhib Obs	Transition threshold	Explore remain candid	Observer	T	Nb of calc-points	Calc Method	dtRatio	Nb States	Nb Transitions	P(t)	Lowest ReFa	Highest Mean Time	Highest Out Time	LimMark time	MarkXPR time
OilProdSyst-prod	red	0		1,212,573	1	production	130000.0	100	FEM	0.25	9,653	951,627	130.969	2.008748968e-11	42413.19589	63.97499985	581.54	54.86
OilProdSyst-prod	red	0		1,299,603	1	production	130000.0	100	FEM	0.25	10,243	1,027,554	130.97	1.364929297e-11	59065.97367	94.5647985	625.32	51.97
OilProdSyst-prod	red	0		1,392,880	1	production	130000.0	100	FEM	0.25	11,223	1,119,478	130.968	8.077085774e-12	58927.08478	77.34437349	669.40	59.80
OilProdSyst-prod	red	0		1,492,852	1	production	130000.0	100	FEM	0.25	13,399	1,288,207	130.969	2.989771745e-12	91913.19589	55.57657876	730.86	69.00
OilProdSyst-prod	red	0		1,600,000	1	production	130000.0	100	FEM	0.25	13,643	1,349,831	130.967	2.569703245e-12	59052.08478	73.42149258	752.78	68.00
OilProdSyst-prod	red	0		1,714,837	1	production	130000.0	100	FEM	0.25	14,555	1,465,827	130.968	1.601498202e-12	92024.307	88.89988006	843.40	74.00
OilProdSyst-prod	red	0		1,736,763	1	production	130000.0	100	FEM	0.25	14,555	1,465,827	130.968	1.601498202e-12	92024.307	88.89988006	786.43	75.00
OilProdSyst-prod	red	0		1,837,917	1	production	130000.0	100	FEM	0.25	15,128	1,545,383	130.967	1.062388626e-12	92024.307	82.50519847	890.89	83.00
OilProdSyst-prod	red	0		1,969,831	1	production	130000.0	100	FEM	0.25	16,135	1,660,144	130.967	7.653014408e-13	59052.08478	89.14704732	949.87	90.00
OilProdSyst-prod	red	0		2,111,212	1	production	130000.0	100	FEM	0.25	18,488	1,863,194	130.967	2.73961226e-13	92038.19589	66.46852189	1024.77	97.00
OilProdSyst-prod	red	0		2,262,741	1	production	130000.0	100	FEM	0.25	19,842	2,013,658	130.966	1.573012142e-13	75496.52922	64.34543179	1102.24	102.00
OilProdSyst-prod	red	0		2,425,146	1	production	130000.0	100	FEM	0.25	20,749	2,148,430	130.967	1.004374484e-13	92163.19589	63.97499985	1180.61	113.00
OilProdSyst-prod	red	0		2,599,207	1	production	130000.0	100	FEM	0.25	21,588	2,294,528	130.968	6.939600553e-14	103054.6879	199.3779186	1267.03	129.00
OilProdSyst-prod	red	0		2,785,761	1	production	130000.0	100	FEM	0.25	23,659	2,503,395	130.967	2.839508233e-14	108940.9737	92.44900122	1355.30	303.00
OilProdSyst-prod	red	0		2,985,705	1	production	130000.0	100	FEM	0.25	26,668	2,756,598	130.967	9.248690959e-15	108927.0848	109.3863364	1454.17	162.00
OilProdSyst-prod	red	0		3,200,000	1	production	130000.0	100	FEM	0.25	27,531	2,923,027	130.967	6.054252007e-15	108954.8626	74.71382974	1519.51	155.00
OilProdSyst-prod	red	0		3,429,675	1	production	130000.0	100	FEM	0.25	28,890	3,143,329	130.967	3.330897392e-15	141913.1959	88.02292593	1689.96	225.00
OilProdSyst-prod	red	0		3,675,834	1	production	130000.0	100	FEM	0.25	32,504	3,483,655	130.967	6.056569372e-16	125496.5292	116.4148925	1797.91	190.00
OilProdSyst-prod	red	0		3,939,662	1	production	130000.0	100	FEM	0.25	33,935	3,710,869	130.967	3.095056837e-16	125496.5292	103.3940583	1899.78	198.00
OilProdSyst-prod	red	0		4,222,425	1	production	130000.0	100	FEM	0.25	37,198	4,044,614	130.967	5.76199218e-17	169624.1323	454.7400854	2039.37	333.00
OilProdSyst-prod	red	0		4,525,483	1	production	130000.0	100	FEM	0.25	39,010	4,332,532	130.967	1.565330996e-17	109315.9737	91.93947621	2194.28	386.00
OilProdSyst-prod	red	0		4,850,293	1	production	130000.0	100	FEM	0.25	42,673	4,742,744	130.967	1.078454337e-18	125746.5292	82.86343992	2356.52	261.00
OilProdSyst-prod	red	0		5,198,415	1	production	130000.0	100	FEM	0.25	46,649	5,186,151	130.967	2.034435647e-22	175746.5292	92.29562253	2511.51	321.00

Filter by Study Name: OilProdSyst-prod

Filter by Case Name: red

Filter by Threshold: -- no filter --

Filter by Observer: production

Filter by Mission Time: 130000.0

Figure 4.6.: A Screenshot of the Qt interface provided by `PyGTS` and presenting the results for the oil production system.

of transitions, T is the mission time, and dt the time step (see Section 2.1.4). The theoretical worst-case time for the current implementation of `GTSLimMark` is $\mathcal{O}(|\Theta| \times \log_2 |\Sigma|)$, as detailed in Section 4.2.1, where $|\Sigma|$ is the number of states.

These run-times can be verified on sufficiently large examples. Figure 4.7 and Figure 4.8 show the execution times of the tools on two examples. The partial model is generated up to ξ transitions, and the resulting Markov chain is calculated. The overall evolution of the time taken to partially generate the Markov chain follows the predicted complexity.

Because each mark on the plot is a single experiment, the measured times may be inaccurate and fluctuating. There are many hardware and software considerations that might explain the random variations of the execution times, among which the concurrent hard drive disk accesses, and the current number of concurrent processes. The computer used to conduct the experiments was dedicated to this task, but it is a common desktop machine. To obtain more accurate results, the experiments should be done multiple times under the same software and hardware conditions.

Figure 4.7 shows the execution times for the distributed database (see Section 3.2.5). For this example, the time taken by the transient solve of the Markov model is much lower than the time taken by the partial generation of the chain. Linear regressions are plotted to have an overview of the time complexities. For `GTSLimMark`, the influence of the $\log_2 |\Sigma|$ is not significant and the time is almost linear with respect to the number of transitions of the partial model.

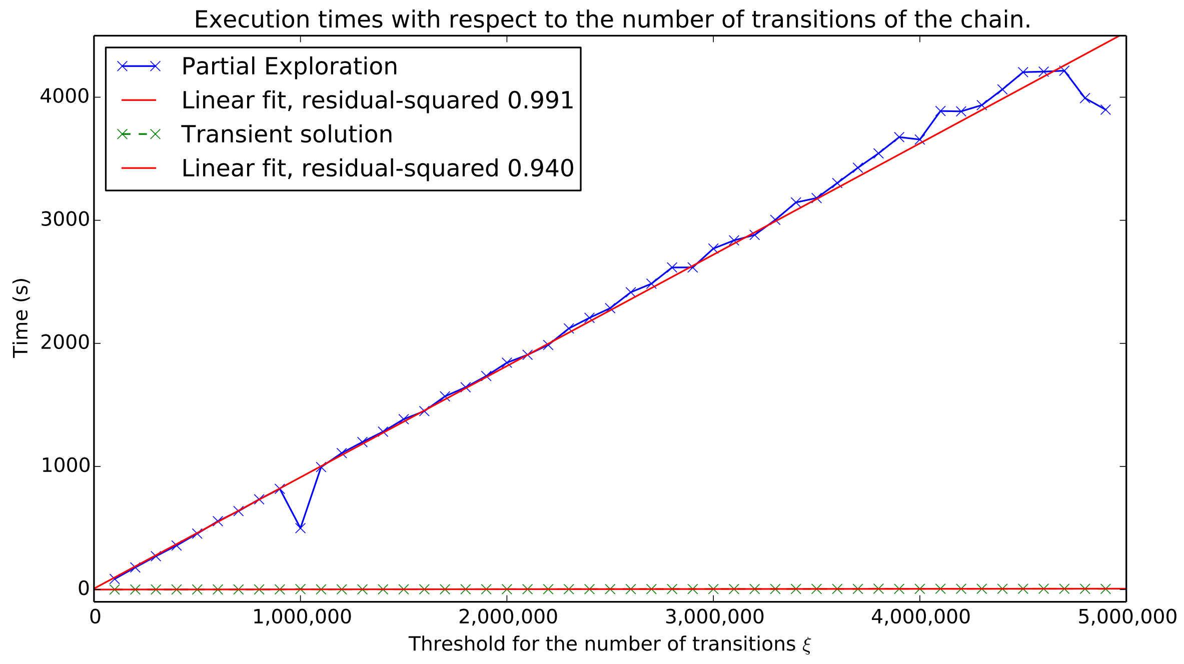


Figure 4.7.: Execution times measured when exploring and solving the distributed database example.

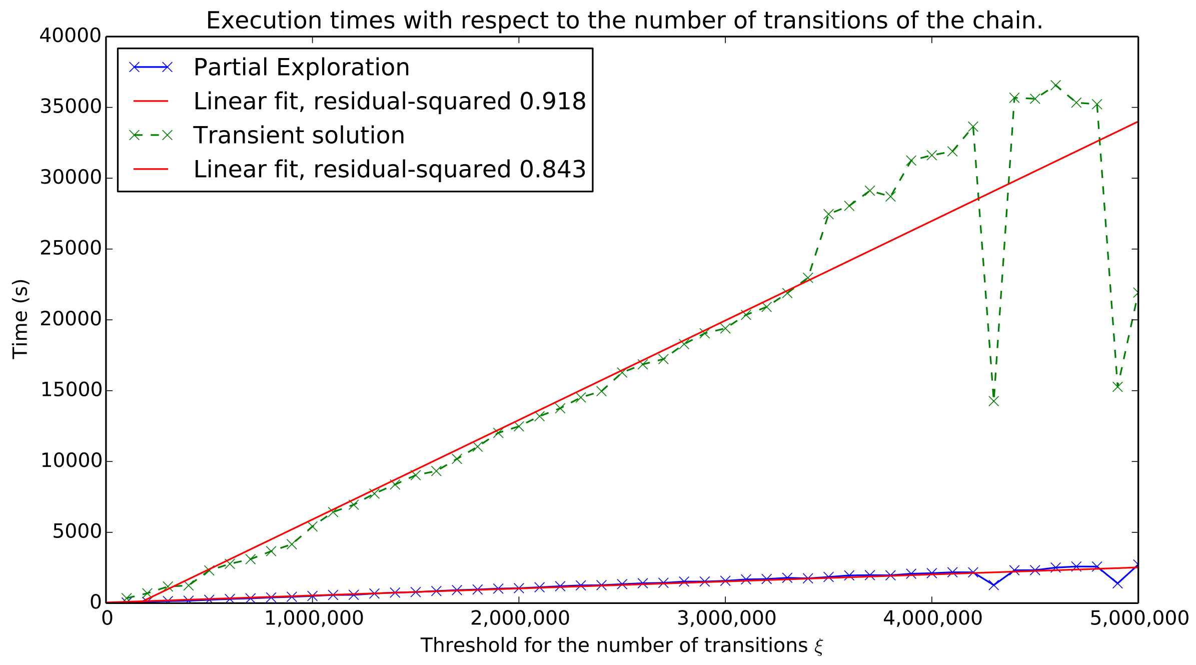


Figure 4.8.: Execution times measured when exploring and solving the stiff emergency power supply example.

Figure 4.8 shows the execution times for the stiff version of the emergency power supply (see Section 3.2.4). For this example, the time taken to calculate the Markov chain is higher than the time taken to generate it. This is due to the fact that the required time step for the iterations of the Markov chain solver is very low, because there are very high repair rates. The overall evolution of the time with respect to the number of transitions for the Markov solver is roughly linear. There are some fluctuations that may be explained by the following. The time step of the Markov chain solver is inversely proportional to the highest element of the transition matrix. By definition, the highest rates are on the diagonal of the transition matrix, as these elements represent the outgoing rates in each state. When ξ grows, there are new failures and repairs in the system, so there might be higher transition rates for some states. Consequently, the time step is reduced suddenly.

Summary

The overall usefulness of `PyGTS` is not obvious. However, it was really useful to manage the experiments, and do the systematic studies of the models, for several different set of parameters. The database is used to have robust and efficient storage of the results without the intermediate files (the Markov chain with sink with $100,000,000$ transitions for the emergency power supply weights nearly 7 gigabytes) and provides the means to take a global perspective on the results.

4.4. Bottlenecks

The size of the models that can be treated with the developed tool-chain is mostly limited by the size of the memory of the computer on which the tools run. The compilation of AltaRica models into GTS models is not a problem, as the number of elements of the model is low (< 100) and the compilation is straightforward.

The calculation of the Markov chain is limited by the number of transitions of the Markov chain. As developed in [Rauzy 2004], the algorithms to calculate a Markov chain use a vector which size is the number of states of the chain, and a sparse matrix whose elements are the transitions of the chain. The sparse matrix can be encoded as a linked list of linked lists of transition rates storing outgoing transition rates for each state of the chain. Hence the memory consumption of the tool is roughly $\mathcal{O}(|\Theta|)$, the number of edges in the reachability graph.

The generation of the reachability graph involves more data sets, as detailed in Section 4.2.1. Current implementation can generate more than $100,000,000$ transitions on a desktop computer with 16 gigabytes of memory. The current implementation of Algorithm 2.2 uses several times more memory than the tool used to compute the Markov chain. With better implementation, the size of the memory structures may be comparable, and the bottleneck may be the calculation of the Markov chain.

Conclusions and Outlooks

Summary

The objective of this thesis is to provide a method to calculate the reliability indicators of a system using Markov chains. The major drawback of the Markov chains is the state space explosion. The idea is to generate partial Markov chains from higher-level descriptions (in this thesis, AltaRica 3.0 models). The partial generation is based on the selection of the most probable states of the Markov chain.

The selection is done by sorting the states according to a relevance factor and using the well-known Dijkstra's algorithm. The relevance factor is based on the probability to go from one state to each of its outgoing states. Hence, it takes into account the overall distribution of the probability in the chain. Thanks to Dijkstra's algorithm, finding the most relevant states is efficient and does not require the complete Markov chain. Furthermore, it is possible to bound the calculated reliability indicators using the Markov chain with sink.

The partial generation was tested on several examples to identify its strengths and weaknesses. The systems that are the most successfully partially generated are repairable systems that are highly available. For a given system, the most accurate reliability indicators are obtained without bounds and with the highest number of transitions in the partial chain. The bounds on the reliability indicators are useful to calculate the error made due to the partial generation, but are usually pessimistic.

The partial generation method is implemented in a tool that is part of the AltaRica 3.0 project. The current implementation can generate Markov chains with more than 100 million transitions.

The relevance factor is used to select the most probable states during the generation of the Markov chain only. It has no intent to replace the calculation of the Markov chain, but rather tries to foresee the main flows of probability through the chain. With this respect, the correlations shown in the experiments highlight that the relevance factor successfully selects most of the most probable states of the Markov chain.

The remainder of the conclusion focuses on two possible improvements of the partial generation of Markov chains. First, the possibility of improving the current relevance factor is discussed. Second, the partial generation is combined with the calculation of the Markov chain. The latter outlook is more interesting, as it suggests a fundamental improvement of the method.

Towards a better relevance factor

The choice of the relevance factor is discussed in Section 2.3.2. The chosen relevance factor mostly takes into account the competition between outgoing states.

It is adapted to unrepairable systems because it can select the most probable branches where the most probable absorbing states are. It also suits repairable systems because the repairs have a heavy influence on the spread of the probabilities, as the competition between outgoing states is unbalanced by repairs. This latter point makes the relevance factor able to consider the *depth* of the state space of the system in terms of number of failures with respect to their frequency.

However, this relevance factor does not take time into account. The mission time changes the distribution of the probabilities in the Markov chain. For instance, a non-repairable system with low mission time is expected to be working whereas it is expected to be failed with a high mission time. The probability will flow from the nominal state to the failed states in this case.

In previous experiments, other relevance factors were tested. The chosen relevance factor must satisfy the following two properties: the factor should be monotonous when exploring the state space (see Section 2.3.2), and its calculation must be easy (see Section 4.2).

The most obvious relevance factor to test is one which only takes into account the mean time to leave the state through transitions: $R(\tau) = \min\limits_{\text{parents } \sigma} \left(R(\sigma) + \dfrac{1}{\lambda_{\sigma \to \tau}} \right)$. Traversing multiple transitions will add time, and exploration could be stopped when this time is larger than a threshold depending on the mission time, as probability would never reach these states within the mission time.

This factor does not yield good results for repairable systems, because the time constant of a repairable system is $\frac{1}{\min \mu}$ (see Section 2.2.2). As a consequence, the depth of the Markov chain is approximated by the mission time, when it should be approximated by the longest repair time. Moreover, the selection of the most probable states proved wrong on very simple examples.

Another factor was constructed using both temporal aspects of outgoing probabilities and outgoing times. Denoting $p(\tau) = p(\sigma) \times \frac{\lambda_{\sigma \to \tau}}{\sum_k \lambda_{\sigma \to k}}$ and $t(\tau) = t(\sigma) + \frac{1}{\sum_k \lambda_{\sigma \to k}}$, the new relevance factor is defined by $R(\tau) = \min\limits_{\text{parents } \sigma} \big(t(\tau)/p(\tau) \big)$ (t is always increasing, and p is always decreasing, so R is always increasing). This relevance factor takes into account both the competition between states and the average time to get out of each state, but mixed results were obtained on simple examples.

A better relevance factor might be obtained by focusing on equilibria between failures and repairs of single components, because this drives the probability in the depths of the state space (see Section 2.2.2). This relevance factor would be, however, dedicated to repairable systems.

It is rational to think that the spread of the probability cannot be predicted without the calculation of the whole Markov chain. In this perspective, the solution might not be to enhance the relevance factor, but to *compensate* its flaws.

Towards an adaptive calculation method

Using Dijkstra and a relevance factor to select the most probable states of a system gives good results but cannot be always accurate, because the behavior of a Markov chain cannot be approximated without taking time into account.

The idea is to use Dijkstra and the relevance factor to preselect the most probable states, and adapt this selection by calculating the *actual probability* of the partial chain. With these probabilities, the values of the relevance factor can be modified to reflect the actual probabilities of the states. This process would be iterated until the mission time is reached.

Hence, the systems for which this adaptive method will be the most efficient are the non-repairable systems. The selection of the states would be adapted to the mission time, and this selection would *slide* to keep only the most probable states at each point of time. Moreover, the error on the calculation could be controlled with the same concept of sink. Eventually, this adaptive selection would lead to the calculation of the reliability indicators without calculation of the whole chain.

Doing these iterations would keep the idea of the partial construction of the Markov chain from a higher-level modeling language, while making the relevance factor less influential.

Bibliography

Anderson, W. J. (1991). *Continuous-Time Markov Chains: An Applications-Oriented Approach (Springer Series in Statistics)*. Springer.

Bobbio, A. and Raiteri, D. C. (2004). Parametric fault trees with dynamic gates and repair boxes. In *Proceedings of the Annual Reliability and Maintainability Symposium (RAMS'2004)*, pages 459–465, Los Angeles, CA, USA. IEEE.

Bon, J. and Collet, J. (1994). An algorithm in order to implement reliability exponential approximations. *Reliability Engineering & System Safety*, 43(3):263–268.

Bouissou, M. (2002). Boolean Logic Driven Markov Processes: a powerful new formalism for specifying and solving very large Markov models. In *Proceedings of the 6th Probabilistic Safety Assessment and Management Conference*.

Bouissou, M. and Bon, J. (2003). A new formalism that combines advantages of fault-trees and markov models: Boolean logic driven markov processes. *Reliability Engineering & System Safety*, 82(2):149–163.

Bouissou, M. and Lefebvre, Y. (2002). A path-based algorithm to evaluate asymptotic unavailability for large markov models. In *Proceedings of Annual Reliability and Maintainability Symposium, RAMS'2002*, pages 32–39, Seattle, USA. IEEE.

Brameret, P.-A., Rauzy, A., and Roussel, J.-M. (2012). Assessing the dependability of systems with repairable and spare components. In Barbet, J., editor, *Actes du Congrès Lambda-Mu 18*.

Brameret, P.-A., Rauzy, A., and Roussel, J.-M. (2015). Automated generation of partial markov chain from high level descriptions. *Reliability Engineering and System Safety*, TBA. Accepted for Publication.

Brameret, P.-A., Roussel, J.-M., and Rauzy, A. (2013). Preliminary system safety analysis with limited markov chain generation. In *Proceedings of 4th IFAC Workshop on Dependable Control of Discrete Systems, DCDS 2013*, York (Great Britain). IFAC.

Brisbois, J., Lanore, J.-M., Villemeur, A., Berger, J.-P., and De Guio, J.-M. (1990). Les études probabilistes de sûreté des centrales nucléaires françaises de 900 et 1 300 mw. *Revue générale nucléaire*, (6):522–535.

Brodal, G. S., Lagogiannis, G., and Tarjan, R. E. (2012). Strict fibonacci heaps. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 1177–1184. ACM.

Bušić, A., Djafri, H., and Fourneau, J.-M. (2012). Bounded state space truncation and censored markov chains. In *Proceedings of IEEE 51st Annual Conference on Decision and Control (CDC)*, pages 5828–5833, Maui, HI, USA. IEEE.

Carrasco, J. A. (2003). Computation of bounds for transient measures of large rewarded markov models using regenerative randomization. *Computers & Operations Research*, 30(7):1005–1035.

Chung, K. L. (1960). *Markov Chains: With Stationary Transition Probabilities (Grundlehren der mathematischen Wissenschaften)*. Springer.

Collet, J. and Renault, I. (1997). Path probability evaluation with repeated rates. In *Proceedings of Annual Reliability and Maintainability Symposium, RAMS'97*, pages 184–187, Philadelphia, PA, USA. IEEE.

Courtois, P.-J. (1977). *Decomposability: queueing and computer system applications*. ACM monograph series. Academic Press, New York, NY.

Courtois, P.-J. and Semal, P. (1984). Bounds for the positive eigenvectors of nonnegative matrices and for their approximations by decomposition. *Journal of the ACM (JACM)*, 31(4):804–825.

Courtois, P.-J. and Semal, P. (1995). Computable dependability bounds for large markov chains. In *Predictably Dependable Computing Systems*, pages 507–518. Springer.

Dayar, T., Pekergin, N., and Younès, S. (2006). Conditional steady-state bounds for a subset of states in markov chains. In *Proceeding from the 2006 workshop on Tools for solving structured Markov chains*, page 3. ACM.

Dijkstra, E. (1959). A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271.

Dingle, N. J., Harrison, P. G., and Knottenbelt, W. J. (2004). Uniformization and hypergraph partitioning for the distributed computation of response time densities in very large markov models. *Journal of parallel and distributed computing*, 64(8):908–920.

Dubreuil Chambardel, A., Villemeur, A., Berger, J., and Moroni, J. (1991). Living probabilistic safety assessment of french 1300 mwe pwr nuclear power plant unit: methodology, results and teaching. Technical report, Electricite de France (EDF), 92-Clamart (France).

Dugan, J., Bavuso, S., and Boyd, M. (1992). Dynamic fault-tree models for fault-tolerant computer systems. *Reliability, IEEE Transactions on*, 41(3):363–377.

Dugan, J. B., Sullivan, K. J., and Coppit, D. (2000). Developing a low-cost high-quality software tool for dynamic fault-tree analysis. *Reliability, IEEE Transactions on*, 49(1):49–59.

Euler, L. (1768). *Institutionum calculi integralis.* Number v. 1 in Institutionum calculi integralis. imp. Acad. imp. Saènt.

Fernandes, P., Plateau, B., and Stewart, W. J. (1998). Efficient descriptor-vector multiplications in stochastic automata networks. *Journal of the Association for Computing Machinery*, 45(3):381–414.

Fourneau, J.-M., Pekergin, N., and Younès, S. (2007). Censoring markov chains and stochastic bounds. In Wolter, K., editor, *Formal Methods and Stochastic Models for Performance Evaluation*, volume 4748 of *Lecture Notes in Computer Science*, pages 213–227. Springer Berlin Heidelberg.

Fourneau, J.-M. and Plateau, B. (1991). A methodology for solving markov models of parallel systems. *Journal of Parallel and Distributed Computing*, 12:370–387.

Fredman, M. L. and Tarjan, R. (1984). Fibonacci heaps and their uses in improved network optimization algorithms. In *Foundations of Computer Science, 1984. 25th Annual Symposium on*, pages 338–346.

Goyal, A., Lavenberg, S. S., and Trivedi, K. S. (1987). Probabilistic modeling of computer system availability. *Annals of Operations Research*, 8(1):285–306.

Grassmann, W. K. (1977). Transient solutions in markovian queueing systems. *Computers & Operations Research*, 4(1):47–53.

Lal, R. and Bhat, U. (1988). Reduced system algorithms for markov chains. *Management Science*, 34(10):1202–1220.

Malhotra, M. and Trivedi, K. S. (1995). Dependability modeling using Petri-nets. *Reliability, IEEE Transactions on*, 44(3):428–440.

Markov, A. (1971). Extension of the limit theorems of probability theory to a sum of variables connected in a chain. In Howard, R., editor, *Dynamic Probabilistic Systems (Volume I: Markov Models)*, chapter Appendix B. John Wiley & Sons, Inc.

Markov, A. A. (1906). Extension of the law of large numbers to dependent quantities. *Izv. Fiz.-Matem. Obsch. Kazan Univ.(2nd Ser)*, 15:135–156.

Mercier, S. (2007). Discrete random bounds for general random variables and applications to reliability. *European journal of operational research*, 177(1):378–405.

Mercier, S. (2008). Bounds and approximations for continuous-time markovian transition probabilities and large systems. *European Journal of Operational Research*, 185(1):216–234.

Merle, G., Roussel, J.-M., and Lesage, J.-J. (2011). Algebraic determination of the structure function of dynamic fault trees. *Reliability Engineering & System Safety*, 96(2):267–277.

Moler, C. and Van Loan, C. (1978). Nineteen dubious ways to compute the exponential of a matrix. *SIAM review*, 20(4):801–836.

Moler, C. and Van Loan, C. (2003). Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. *SIAM review*, 45(1):3–49.

Montani, S., Portinale, L., Bobbio, A., Varesio, M., and Codetta-Raiteri, D. (2006). A tool for automatically translating dynamic fault trees into dynamic bayesian networks. In *Reliability and Maintainability Symposium, 2006. RAMS'06. Annual*, pages 434–441, Newport Beach, CA. IEEE.

Mortada, H., Prosvirnova, T., and Rauzy, A. (2014). Safety assessment of an electrical system with altarica 3.0. In Ortmeier, F. and Rauzy, A., editors, *Model-Based Safety and Assessment*, volume 8822 of *Lecture Notes in Computer Science*, pages 181–194. Springer International Publishing.

Muntz, R. R., de Souza e Silva, E., and Goyal, A. (1989). Bounding availability of repairable computer systems. *SIGMETRICS Perform. Eval. Rev.*, 17(1):29–38.

Pagès, A. and Gondran, M. (1980). Fiabilité des systèmes. *Collection de la Direction des Etudes et Recherche dEDF, Eyrolles*, 39.

Papadimitriou, C. H. (1994). *Computational Complexity*. Addison Wesley, Boston, MA 02116, USA.

Pribadi, Y., Voeten, J. P. M., and Theelen, B. D. (2001). Reducing markov chains for performance evaluation. In *Proceedings of PROGRESS'01*, pages 173–179. STW Technology Foundation.

Prosvirnova, T. (2014). *AltaRica 3.0: a Model-Based approach for Safety Analyses*. Theses, Ecole Polytechnique.

Prosvirnova, T., Batteux, M., Brameret, P.-A., Cherfi, A., Friedlhuber, T., Roussel, J.-M., and Rauzy, A. (2013). The altarica 3.0 project for model-based safety assessment. In *Proceedings of 4th IFAC Workshop on Dependable Control of Discrete Systems, DCDS 2013*, pages 127–132, York (Great Britain). IFAC.

Prosvirnova, T. and Rauzy, A. (2012). Système de transitions gardées : formalisme pivot de modélisation pour la sûreté de fonctionnement. In Barbet, J., editor, *Actes du Congrès Lambda-Mu 18*.

Rauzy, A. (1993). New algorithms for fault trees analysis. *Reliability Engineering & System Safety*, 40(3):203–211.

Rauzy, A. (2002). Mode automata and their compilation into fault trees. *Reliability Engineering & System Safety*, 78(1):1–12.

Rauzy, A. (2004). An experimental study on iterative methods to compute transient solutions of large markov models. *Reliability Engineering & System Safety*, 86(1):105–115.

Rauzy, A. (2008). Guarded transition systems: a new states/events formalism for reliability studies. *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, 222(4):495.

Reibman, A. and Trivedi, K. (1988). Numerical transient analysis of markov models. *Computers & Operations Research*, 15(1):19 – 36.

SAE (1996). ARP 4761, guidelines and methods for conducting the safety assessment process on civil airborne systems and equipment. *SAE International, December.*

Stamatis, D. H. (2003). *Failure mode and effect analysis: FMEA from theory to execution.* ASQ Quality Press.

Stewart, W. (1994). *Introduction to the numerical solution of Markov chains*, volume 1. Princeton University Press.

Valiant, L. G. (1979). The complexity of enumeration and reliability problems. *SIAM Journal of Computing*, 8(3):410–421.

Vesely, W., Dugan, J., Fragola, J., Minarick, J., and Railsback, J. (2002). Fault tree handbook with aerospace applications. Technical report, National Aeronautics and Space Administration.

Walker, M. and Papadopoulos, Y. (2009). Qualitative temporal analysis: Towards a full implementation of the fault tree handbook. *Control Engineering Practice*, 17(10):1115–1125.

Zhao, Y. Q. and Liu, D. (1996). The censored markov chain and the best augmentation. *Journal of Applied Probability*, 33(3):623–629.

A. Guarded Transition Systems

As defined in [Prosvirnova and Rauzy 2012], a Guarded Transition System (GTS) is a quintuple $\langle V, E, T, A, \iota \rangle$, where:

- $V = S \uplus F$ is a set of variables, divided into two disjoint subsets: the subset S of state variables and the subset F of flow variables.
- E is a set of events.
- T is a set of transitions. A transition is a triple $\langle e, G, P \rangle$, denoted as $e : G \rightarrow P$, where $e \in E$ is an event, G is a guard, i.e. a Boolean formula built over V, and P is an instruction built over V, called the action of the transition. The action modifies only state variables.
- A is an assertion, i.e. an instruction built over V. The assertion modifies only flow variables.
- ι is the initial assignment of variables of V.

In a GTS, states of the system are represented by variable assignments. A transition $e : G \rightarrow P$ is said fireable in a given state σ if its guard G is satisfied in this state, i.e. if $G(\sigma) = \texttt{true}$. The firing of that transition transforms the state σ into the state $\sigma' = A(P(\sigma))$, i.e. σ' is obtained from σ by applying successively the action of the transition and the assertion.

Guarded Transition Systems are implicit representations of labeled Kripke structures, i.e. of graphs whose nodes are labeled by variable assignments and whose edges are labeled by events. The so-called reachability graph $\Gamma = \langle \Sigma, \Theta \rangle$ of a GTS $\langle V, E, T, A, \iota \rangle$ is the smallest Kripke structure such that:

- $\iota \in \Sigma$.
- If $\sigma \in \Sigma$, $e : G \rightarrow P$ is a transition of T and $G(\sigma) = \texttt{true}$ (the transition is fireable in σ), then $\sigma' = A(P(\sigma)) \in \Sigma$ and $e : \sigma \rightarrow \sigma' \in \Theta$.

If exponential distributions are associated with events of E, the Krypke structure $\Gamma = \langle \Sigma, \Theta \rangle$ can be interpreted as a Continuous Time Homogeneous Markov Chain (for sake of brevity we shall just write Markov Chain in the remainder of the article). The reliability indicators (such as system unavailability) can be defined by associating a reward (a real number) with each state of the chain.