PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
FACULDADE DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

PRECISÃO DE SIMULAÇÕES PARA SOLUÇÃO DE MODELOS ESTOCÁSTICOS

DIONE TASCHETTO

Dissertação de Mestrado apresentada como requisito para obtenção do título de Mestre em Ciência da Computação pelo Programa de Pós-graduação da Faculdade de Informática. Área de concentração: Ciência da Computação.

Orientador: Paulo Henrique Lemelle Fernandes

Porto Alegre, Brasil
2010

Dados Internacionais de Catalogação na Publicação (CIP)

T197p Taschetto, Dione
Precisão de simulações para solução de modelos estocásticos / Dione Taschetto. – Porto Alegre, 2010.
91 p.

Diss. (Mestrado) – Fac. de Informática, PUCRS.
Orientador: Prof. Dr. Paulo Henrique Lemelle Fernandes.

1. Informática. 2. Redes de Autômatos Estocásticos. 3. Avaliação de Desempenho (Informática). 4. Simulação e Modelagem em Computadores. I. Fernandes, Paulo Henrique Lemelle. II. Título.

CDD 003.3

Ficha Catalográfica elaborada pelo
Setor de Tratamento da Informação da BC-PUCRS

Pontifícia Universidade Católica do Rio Grande do Sul
FACULDADE DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

TERMO DE APRESENTAÇÃO DE DISSERTAÇÃO DE MESTRADO

Dissertação intitulada "**Precisão de Simulações para Solução de Modelos Estocásticos**", apresentada por Dione Taschetto, como parte dos requisitos para obtenção do grau de Mestre em Ciência da Computação, Processamento Paralelo e Distribuído, aprovada em 12/03/10 pela Comissão Examinadora:

Prof. Dr. Paulo Henrique Lemelle Fernandes – Orientador — PPGCC/PUCRS

Prof. Dr. Fernando Luís Dotti – — PPGCC/PUCRS

Dr. Afonso Henrique Corrêa de Sales - — Bolsista PNPD FACIN/PUCRS

Prof. Dr. Nicolas Maillard - — UFRGS

Homologada em 14/12/2010, conforme Ata No. 24 pela Comissão Coordenadora.

Prof. Dr. Fernando Gehm Moraes
Coordenador.

PUCRS
Campus Central
Av. Ipiranga, 6681 – P32– sala 507 – CEP: 90619-900
Fone: (51) 3320-3611 – Fax (51) 3320-3621
E-mail: ppgcc@pucrs.br
www.pucrs.br/facin/pos

“A journey of a thousand miles
begins with a single step.”
Lao-tzu, The Way of Lao-tzu

Agradecimentos

A Deus, por tudo.

À minha família por toda dedicação durante estes dois anos de apoio e confiança, muito obrigado.

Ao meu orientador, Professor Dr. Paulo Fernandes, pelas importantes considerações que contribuíram para a realização deste trabalho.

Aos professores e pesquisadores Nicolas Maillard, Fernando Luís Dotti e Afonso Sales por terem aceitado o convite para compor a banca de defesa de mestrado.

A todos os colegas do PEG e do PALEOPROSPEC pelo companheirismo, especialmente à Thais Webber por ter sido meu braço direito em todas as etapas do mestrado e ao Ricardo Czekster pelas dicas com Perl e LaTeX.

Aos meus grandes amigos Gabriel Pedebos e Paulo Júnior Pivetta pelo apoio e a amizade, e à Rita Berardi pelas dicas e pelo incentivo.

Ao pessoal do LAD, principalmente ao Antônio Lima e ao Elder Bernardi por terem me auxiliado na configuração e na manutenção das máquinas.

Aos demais colegas, professores e funcionários do programa de pós-graduação, obrigado pela atenção, pela paciência e pelas sugestões.

À CAPES, pelo apoio financeiro e à PUCRS pela oportunidade.

Muito obrigado a todos!

PRECISÃO DE SIMULAÇÕES PARA SOLUÇÃO DE MODELOS ESTOCÁSTICOS

RESUMO

Através de formalismos Markovianos é possível modelar diversos sistemas e resolvê-los através de soluções computacionais específicas possibilitando prever ou avaliar seus padrões de comportamento. O formalismo de Redes de Autômatos Estocásticos (SAN) permite descrever modelos Markovianos de forma compacta e modular. Além disso, é utilizado para obter índices de desempenho de sistemas através de soluções numéricas iterativas que se baseiam em um descritor e um vetor cujo tamanho é igual ao espaço de estados do modelo. Dependendo do tamanho do modelo esta operação torna-se computacionalmente onerosa e muitas vezes impraticável. Um método alternativo para calcular índices a partir de um modelo é a simulação, principalmente porque ela simplesmente exige a definição de um gerador de números pseudo-aleatórios e funções de transição entre estados que permitem a criação de uma trajetória. O processo de amostragem pode ser diferente para cada técnica estabelecendo algumas regras para coleta de amostras para posterior análise estatística. As técnicas de simulação, normalmente requerem muitas amostras para calcular índices de desempenho estatisticamente relevantes. Este trabalho proporciona comparações da precisão dos resultados de alguns modelos Markovianos obtidos a partir da execução de diferentes técnicas de simulação. Além disso, propõe uma maneira distinta de simular modelos Markovianos usando um método baseado em estatística *Bootstrap* para minimizar o efeito de escolha das amostras. A eficácia do método proposto, denominado *Bootstrap simulation*, é comparado com resultados da solução numérica para um conjunto de exemplos descritos por meio do formalismo de modelagem SAN.

Palavras-chave: Avaliação de Desempenho; Redes de Autômatos Estocásticos; Simulação; Precisão.

ACCURACY OF SIMULATION FOR STOCHASTIC MODELS SOLUTION

ABSTRACT

The use of Markovian formalisms make possible the use and the computational solution of several systems enabling the prediction and evaluation of their behavior standards. The Stochastic Automata Networks (SAN) formalism provides a compact and modular description for Markovian models. Moreover, SAN is suitable to derive performance indices for systems analysis and interpretation using iterative numerical solutions based on a descriptor and a state space sized probability vector. Depending on the size of the model this operation is computationally onerous and sometimes impracticable. An alternative method to compute indices from a model is simulation, mainly because it simply requires the definition of a pseudorandom generator and transition functions for states that enable the creation of a trajectory. The sampling process can be different for each technique, establishing some rules to collect samples for further statistical analysis. Simulation techniques often demand lots of samples in order to calculate statistically relevant performance indices. This work provides comparisons with accuracy of results from some Markovian models which were obtained from the execution of different simulation techniques. It also proposes a different way to simulate Markovian models by using a Bootstrap-based statistical method to minimize the effect of sample choices. The effectiveness of the proposed method, called *Bootstrap simulation*, is compared to the numerical solution results for a set of examples described using SAN modeling formalism.

Keywords: Performance Evaluation; Stochastic Automata Networks; Simulation; Precision.

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Lista de Siglas

SAN	*Stochastic Automata Networks*
GSPN	*Generalized Stochastic Petri Nets*
PEPA	*Performance Evaluation Process Algebra*
PSS	*Product State Space*
RSS	*Reachable State Space*
ASP	*Alternate Service Pattern*
FAS	*First Available Server*
RS	*Resource Sharing*

Sumário

Capítulo 1

Introdução

Ao analisar um sistema é possível utilizar formalismos que facilitam sua representação permitindo descrevê-lo de forma a evidenciar as diferentes configurações que ele pode apresentar. Os formalismos são comumente utilizados na modelagem de sistemas computacionais, sistemas industriais ou sistemas que envolvam fenômenos da natureza. A modelagem por formalismos permite a aplicação de soluções computacionais para resolvê-los e possibilita sua análise a partir dos resultados obtidos. Esta análise permite avaliar seus padrões de comportamento prevendo eventos ou cenários de maior ou menor probabilidade de ocorrência.

Dentre as soluções utilizadas para a resolução de sistemas através de formalismos, destacam-se principalmente os métodos analíticos e a simulação. Os métodos analíticos consistem, normalmente, no emprego de métodos matemáticos iterativos, como o Método da Potência [9, 37], o Método de Arnoldi [5] e o Método GMRES (*Generalized minimal residual method*) [34]. Estes métodos constituem-se, basicamente, na multiplicação de um vetor por uma matriz, denominada matriz de taxas de transição (escala de tempo contínua) ou matriz de probabilidades de transição (escala de tempo discreta). A solução estacionária (vetor resultante), do processo iterativo, irá conter as probabilidades de permanência em cada estado do sistema modelado. A obtenção da matriz, utilizada neste processo, se dá a partir de um grafo ou autômato que representa o sistema como um conjunto finito de estados possíveis e transições entre estes estados. Os estados correspondem às configurações do sistema e as transições identificam as mudanças possíveis de um estado para outro. Estas transições são rotuladas por eventos que, em escala de tempo contínuo, apresentam as taxas ou frequências em que as mesmas ocorrem [37]. Esta matriz de transição de estados é derivada de um grafo também denominado Cadeia de Markov.

Cadeias de Markov é um formalismo analítico, não estruturado, amplamente utilizado por sua facilidade de representação e baixa complexidade de resolução. Porém, uma de suas limitações é a ocorrência da explosão do seu espaço de estados, que acontece quando as configurações possíveis do sistema aumentam significativamente, tornando-as difíceis de serem tratadas através de uma única matriz de transição correspondente [37]. Dentre as alternativas de modelagem conhecidas, que tentam minimizar este problema, sobressaem-se formalismos estruturados, como por exemplo, *Stochastic Au-*

tomata Networks (SAN) [19, 31, 32], *Generalized Stochastic Petri Nets (GSPN)* [2, 16] e *Perfomance Evaluation Process Algebra (PEPA)* [22, 23]. Estes formalismos apresentam soluções numéricas mais eficientes em memória, para grandes modelos, do que as Cadeias de Markov.

Outra solução bastante utilizada na resolução de sistemas é a simulação. Esta solução, geralmente baseia-se na geração de números aleatórios que atuam nas decisões de comportamento e na interação entre os componentes do sistema analisado. No entanto, a simulação consiste em aproximações, sendo necessário executá-la por um tempo suficientemente grande para que ela se aproxime da solução obtida com os métodos iterativos.

Na simulação, além de uma matriz de transição, trabalha-se com uma função de transição que define a ocorrência dos eventos através de uma análise nas taxas discretizadas [38] de cada linha da matriz. Esta discretização corresponde à conversão dessas taxas em probabilidades e constitui o vetor, também de probabilidades, a partir da coleta de amostras obtidas através de trajetórias disparadas no modelo.

A escolha da solução mais adequada em uma determinada situação deve considerar critérios como a disponibilidade de tempo, recursos computacionais existentes e o nível de precisão requerido nos resultados.

1.1 Justificativa

Embora as soluções analíticas apresentem precisão nos resultados, as mesmas deparam-se com limitações que estão ligadas, principalmente, no que diz respeito ao armazenamento da matriz de transição e dos vetores utilizados no processo iterativo. A utilização do formalismo SAN se sobressai às Cadeias de Markov por apresentar uma forma de representação compacta em memória, permitindo assim modelar sistemas com maior número de configurações (estados). Além disso, no que diz respeito a SAN algumas técnicas para otimizar a resolução de modelos têm sido aprimoradas com a finalidade de acelerar a convergência na resolução dos métodos iterativos. Neste contexto pode ser mencionada a solução *Split* [14], a qual adota primitivas de álgebra tensorial bastante eficientes, combinadas com técnicas de armazenamento para matrizes esparsas. No entanto, conforme diminui o nível de abstração da realidade modelada exigindo que ainda mais estados sejam representados, aumenta a carga computacional aplicada às ferramentas numéricas que resolvem este formalismo. Este fato faz com que a resolução do mesmo também alcance os limites computacionais.

Quando se deseja resolver modelos com um número de estados que ultrapassam a capacidade atual de resolução dos métodos iterativos, destaca-se então a simulação. A simulação apresenta-se como uma alternativa viável por possibilitar a exploração de métodos e técnicas de armazenamento em memória de forma menos complexa que nos métodos iterativos. Com isso, torna-se possível a resolução de modelos com um número de estados ainda maior.

Ao resolver estes modelos através da simulação, um fator crítico é a precisão dos resutados obtidos através desta solução. Mesmo assim, trabalhos que analisam especificamente a precisão dos

resultados da simulação ainda são desconhecidos até o momento.

Uma análise minuciosa em relação aos resultados de diferentes técnicas de simulação pode evidenciar particularidades ainda não questionadas e apontar características específicas sobre o comportamento de cada uma delas. Além disso, os resultados obtidos por simulações podem ser comparados com os resultados obtidos por métodos iterativos, para um modelo equivalente, o que possibilita demonstrar o erro de cada uma das técnicas. Dentre as técnicas de simulação conhecidas para a resolução de formalismos Markovianos estão a *Forward Simulation* [21], a *Backward Coupling Simulation* [21, 33] e a simulação tradicional [3], chamada neste trabalho de *Permanence Time Simulation*.

1.2 Objetivos

O objetivo geral deste trabalho é evidenciar quantitativamente de forma comparativa a precisão dos resultados observados para alguns modelos Markovianos ao aplicar diferentes técnicas de simulação. Esta comparação visa auxiliar na escolha da melhor técnica a ser adotada, dentre as que serão analisadas, sendo elas: $a)$ *Forward Simulation*; $b)$ *Backward Coupling Simulation*; e $c)$ *Permanence Time Simulation*. Além disso, será realizado um estudo no método *Bootstrap* [18] a fim de propor a adaptação do mesmo como uma nova técnica de simulação visando a obtenção de resultados mais precisos considerando também os tempos de processamento de cada técnica.

As técnicas de simulação serão analisadas comparando seus resultados com os da solução iterativa, obtida através do Método da Potência [9, 37] utilizando sistemas modelados com o formalismo SAN, equivalentes em ambas as abordagens. Para resolvê-los e obter o vetor de probabilidades com os resultados, desenvolveu-se um simulador que realiza trajetórias no estado global destes sistemas, o qual corresponde ao estado da Cadeia de Markov correspondente à SAN. Vale destacar que isso pode ser realizado porque toda SAN possui uma Cadeia de Markov subjacente [37]. Como o objetivo é observar o impacto da simulação em relação à precisão dos resultados, acredita-se que o formato esparso de representação dos modelos seja independente do formalismo. Logo, optou-se por realizar a simulação desta forma, pela praticidade de tratar um Gerador Infinitesimal[1] de uma Cadeia de Markov, do que tratar um Descritor Markoviano [37] de SAN, o qual é composto por várias matrizes de transição de menor dimensão, porém com operadores tensoriais.

O foco das comparações e análises que serão realizadas neste trabalho será em relação ao vetor de probabilidades resultante ao aplicar as técnicas de simulação. A partir dele, a precisão será avaliada analisando o erro relativo entre o resultado obtido com a simulação e o resultado da solução iterativa.

Para identificar particularidades que possam causar discrepâncias nos resultados, de forma a acarretar perda de precisão, pretende-se observar outros pontos, como a relevância no tamanho da trajetória e na quantidade de amostras envolvidas no processo de simulação. Para obter os resultados em tempo hábil foi utilizada também a paralelização na implementação das técnicas de simulação.

[1] O Gerados Infinitesimal de Cadeias de Markov será visto no próximo capítulo.

Como propósito da paralelização, deseja-se reduzir o tempo de processamento na tentativa de obter resultados mais precisos, uma vez que, normalmente, uma das desvantagens da simulação comparado às soluções iterativas é o tempo despendido até atingir uma aproximação da solução estacionária[2].

1.3 Estrutura do Volume

Para a satisfação dos objetivos apontados é necessário o entendimento sobre os seguintes assuntos: Formalismos Markovianos, abordados no Capítulo 2, onde é descrito o formalismo de Cadeias de Markov e de Redes de Autômatos Estocásticos (SAN), ilustrando suas respectivas representações e os modelos utilizados neste trabalho; Simulação, abordado no Capítulo 3, onde são tratados assuntos sobre o processo de simulação e as diferentes técnicas utilizadas para a resolução de modelos Markovianos.

No Capítulo 4 são descritos os testes e os resultados obtidos com as técnicas de simulação conhecidas. Em seguida, no Capítulo 5 é apresentada a nova técnica de simulação baseada no método *Bootstrap* e os respectivos resultados obtidos com ela. Por fim, no Capítulo 6 são apresentadas a conclusão, a contribuição e os trabalhos futuros referentes a esta dissertação.

[2] A solução estacionária do método iterativo pode ser obtida subtraindo os resultados do vetor da solução atual com os resultados da solução do vetor anterior. Se o mesmo for zero, ou próximo de zero dado uma certa tolerância, significa que a solução estacionária foi então atingida [26].

Capítulo 2

Formalismos Markovianos

Formalismos Markovianos são constantemente utilizados na modelagem de diversos sistemas [8, 10, 13, 17, 29, 35, 41]. O emprego destes formalismos unidos a soluções computacionais baseadas em métodos matemáticos para sua resolução, apresentam-se como boas práticas para avaliar o desempenho de sistemas e analisar seus padrões de comportamento. A seguir são apresentados os formalismos de Cadeias de Markov e de Redes de Autômatos Estocásticos (SAN), demonstrando suas respectivas formas de representação.

2.1 Cadeias de Markov

Cadeias de Markov é um formalismo matemático utilizado para a modelagem de sistemas proposto pelo matemático russo Andrei Andreyevich Markov em 1906 [37]. Através deste formalismo o funcionamento dos sistemas pode ser descrito através de um conjunto de estados possíveis e de transições entre estes estados. As transições são modeladas por um processo estocástico[1] de tempo contínuo ou discreto, os quais são definidos por distribuições de probabilidade exponenciais ou geométricas respectivamente. Este trabalho irá tratar apenas sistemas de tempo contínuo, porém com estados discretos.

Em um sistema com espaço de estados discreto, as variáveis que compõem seu estado mudam de valor instantaneamente. Neste caso, podem ser mencionados os nodos de um grafo, sendo que é possível apenas estar em um ou em outro estado (nodo) do grafo, não havendo um estado intermediário.

Para facilitar a compreensão da representação de um sistema modelado através do formalismo de Cadeias de Markov é ilustrada uma cadeia na Figura 2.1. Os nodos da figura representam os estados $S = \{00, 01, 02, 10, 11, 12\}$ do sistema e as setas correspondem às transições entre um estado e outro. Estas setas são rotuladas por taxas (μ_0, μ_1, μ_2, ..., μ_5), que são definidas por valores que determinam a frequência do disparo das transições.

[1] Um processo estocástico é qualquer tipo de evolução temporal que seja analisável em termos de probabilidade.

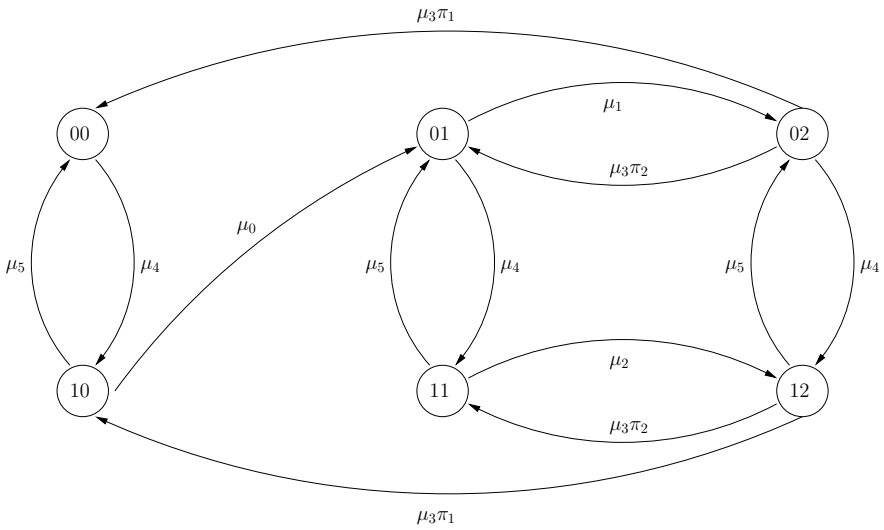


Figura 2.1: Cadeia de Markov

A Cadeia de Markov apresentada na Figura 2.1 pode também ser representada através de uma matriz, denominada Matriz de Taxas de Transição, conforme pode ser observado na Tabela 2.1. As taxas, na matriz, são distribuídas de forma que cada linha i e coluna j indicam a taxa de transição de um estado para outro (representado entre parênteses) na Cadeia de Markov, sendo a dimensão da matriz igual ao número de estados do modelo, ou seja $|S|$. Na resolução deste formalismo é necessário que esta matriz de transição seja um Gerador Infinitesimal [37] de forma que a soma de cada linha i da matriz seja igual a $zero$. Para realizar este ajuste, adiciona-se à diagonal principal, o complemento da soma de todos os elementos não diagonais de cada linha, resultando então no Gerador Infinitesimal Q da Tabela 2.2.

Tabela 2.1: Matriz de Taxas de Transição

		j					
		0 (00)	1 (01)	2 (02)	3 (10)	4 (11)	5 (12)
i	0(00)	0	0	0	μ_4	0	0
	1 (01)	0	0	μ_1	0	μ_4	0
	2 (02)	$\mu_3\pi_1$	$\mu_3\pi_2$	0	0	0	μ_4
	3 (10)	μ_5	μ_0	0	0	0	0
	4 (11)	0	μ_5	0	0	0	μ_2
	5 (12)	0	0	μ_5	$\mu_3\pi_1$	$\mu_3\pi_2$	0

Feito isso, têm-se um sistema de equações, de forma que $\pi Q = 0$, onde deseja-se encontrar o vetor π (não confundir o vetor π com as probabilidades π_1 e π_2 multiplicadas pela taxa μ_3 da matriz). Como a resolução de um sistema deste tipo torna-se muito complexa com o aumento de estados do modelo, um dos métodos mais utilizados para a obtenção do vetor é o Método da Potência [37]. Para

Tabela 2.2: Gerador Infinitesimal Q

$$Q = \begin{pmatrix} -(\mu_4) & 0 & 0 & \mu_4 & 0 & 0 \\ 0 & -(\mu_1+\mu_4) & \mu_1 & 0 & \mu_4 & 0 \\ \mu_3\pi_1 & \mu_3\pi_2 & -(\mu_3\pi_1+\mu_3\pi_2+\mu_4) & 0 & 0 & \mu_4 \\ \mu_5 & \mu_0 & 0 & -(\mu_5+\mu_0) & 0 & 0 \\ 0 & \mu_5 & 0 & 0 & -(\mu_5+\mu_2) & \mu_2 \\ 0 & 0 & \mu_5 & \mu_3\pi_1 & \mu_3\pi_2 & -(\mu_5+\mu_3\pi_1+\mu_3\pi_2) \end{pmatrix}$$

que este método possa ser aplicado, é necessário primeiramente transformar o Gerador Infinitesimal Q em uma matriz chamada Matriz de Probabilidades de Transição ou Matriz Estocástica P (uma vez que a solução também corresponde a um vetor de probabilidades). Feito isso, não podem haver valores negativos na matriz e a soma de todos os elementos de cada linha devem ser iguais a um. Este processo corresponde à discretização da matriz Q [38] que pode ser realizada dividindo toda a matriz pelo seu maior elemento Λ_{max} em módulo, e somando a ela uma matriz identidade I de mesma dimensão, conforme a equação 2.1.

$$P = I + \frac{1}{|\Lambda_{max}|}Q \tag{2.1}$$

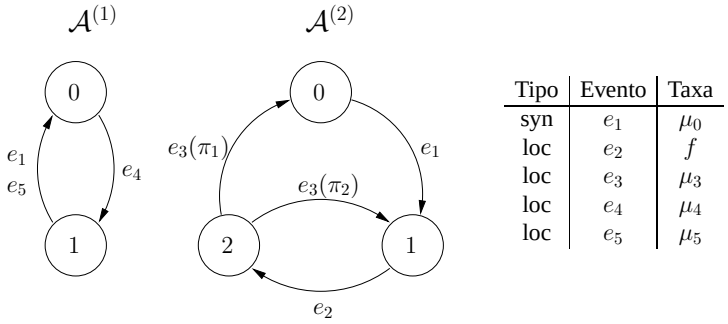
O método da potência consiste, basicamente, em sucessivas multiplicações de um vetor qualquer por uma matriz, de forma iterativa. Em Cadeias de Markov esta matriz corresponde à matriz de probabilidades de transição P, sendo que no final do processo iterativo resulta-se em um novo vetor π, de forma que $\pi P = \pi$, sendo a dimensão de π igual ao número de estados do modelo. Este vetor resultante conterá as probabilidades de permanência em cada estado sendo ele a solução estacionária que, neste caso, baseia-se em um critério de parada que pode ser um erro aceitável, resultante da diferença dos vetores entre duas iterações. A expressão a seguir ilustra o processo iterativo, sendo $\pi^{(n)}$ a solução estacionária:

$$\begin{aligned} \pi^{(1)} &= \pi^{(0)}P \\ \pi^{(2)} &= \pi^{(1)}P \\ \pi^{(3)} &= \pi^{(2)}P \\ &\cdots \\ \pi^{(n)} &= \pi^{(n-1)}P \end{aligned}$$

Um dos principais problemas das Cadeias de Markov é a ocorrência da explosão do seu espaço de estados, que ocorre conforme aumentam as configurações possíveis (nível de detalhe) do sistema que está sendo modelado. Este fato torna as Cadeias de Markov difíceis de serem tratadas através de uma única matriz de transição correspondente [32, 37] pois aumenta também a dimensão da matriz envolvida no processo iterativo. Para minimizar este problema, formalismos estruturados como SAN podem ser utilizados.

2.2 Redes de Autômatos Estocásticos

Redes de Autômatos Estocásticos (SAN) é um formalismo baseado em Cadeias de Markov, proposto por Plateau em 1985 [31]. Consiste em uma forma modular de descrever sistemas complexos com grandes espaços de estados a serem modelados. Para isso necessitam da resolução de modelos matemáticos robustos, que utilizam a álgebra tensorial ou de Kronecker [4, 15] como forma de armazenamento compacto. Desta forma, este formalismo consegue manter o mesmo poder de solução obtido com a utilização de Cadeias de Markov, propondo para tal um novo formato de armazenamento que reduz o problema da explosão do espaço de estados [32, 37].



Tipo	Evento	Taxa
syn	e_1	μ_0
loc	e_2	f
loc	e_3	μ_3
loc	e_4	μ_4
loc	e_5	μ_5

Figura 2.2: Modelo SAN com eventos locais, sincronizantes, taxas funcionais e probabilidades funcionais

Este formalismo é composto por no mínimo dois ou mais autômatos, sendo que cada um deles é definido na forma de um grafo constituído de um conjunto finito de estados e transições entre estes estados, conforme pode ser observado na Figura 2.2, sendo os estados do sistema representados pelos nodos. O estado em que um autômato qualquer se encontra é chamado de estado local, enquanto que o conjunto de estados que todos os autômatos se encontram é denominado estado global, o qual corresponde ao estado na Cadeia de Markov equivalente ao modelo em estudo. Isto é possível, pois toda SAN pode ser representada por uma Cadeia de Markov, a qual consiste em um único autômato estocástico [37].

A Cadeia de Markov subjacente ao modelo da Figura 2.2 é a da Figura 2.1, apresentada anteriormente. Essa equivalência é feita pelas combinações de estados possíveis entre os autômato $A^{(1)}$ e $A^{(2)}$ da SAN. Para entendê-la considere o estado local do autômato $A^{(1)}$ igual a 0 e o estado local do autômato $A^{(2)}$ igual a 1, neste caso, o estado global na Cadeia de Markov da Figura 2.1 é o estado 01. Sendo assim, o espaço de estados é calculado pelo produto cartesiano da dimensão de todos os autômatos de uma SAN, definido como espaço de estados produto (*product state space* - PSS). No caso do modelo da Figura 2.2 o PSS é $2 \times 3 = 6$.

Para a resolução deste formalismo é conhecida a ferramenta analítica PEPS (*Performance Evaluation of Parallel Systems*) [6, 27] que emprega métodos matemáticos iterativos, como o Método

da Potência [9, 37], o Método de Arnoldi [5] e o Método GMRES (*Generalized minimal residual method*) [34]. Uma das funcionalidades providas por esta ferramenta é a utilização de uma função de atingibilidade[2]. Esta função representa quais estados globais da SAN são atingíveis durante o comportamento do sistema modelado e constitui então o chamado espaço de estados atingível (*reachable state space* - RSS) do modelo. No caso do exemplo mencionado (Figura 2.2), todos os estados são atingíveis, no entanto se algum deles não o fosse, o espaço de estados seria menor do que 6.

Além do PSS e do RSS, o formalismo SAN, ao contrário de Cadeias de Markov, apresenta como uma de suas principais características a possibilidade de dividir um sistema complexo em subsistemas que interagem ocasionalmente. Esta interação ocorre através de eventos sincronizantes ou até mesmo através de taxas funcionais [11]. Estes eventos são os responsáveis por disparar as transições entre os estados de um autômato e são descritos com mais detalhes a seguir.

2.2.1 Eventos Locais e Eventos Sincronizantes

Para que possam ocorrer transições entre os estados de um autômato é necessário que existam eventos associados a estas transições. Eventos locais caracterizam-se por alterar o estado de apenas um autômato do modelo, possibilitando com que os autômatos tenham comportamentos paralelos. Sendo assim, um evento local não interfere no estado dos demais. Os eventos locais podem ser observados no autômato da Figura 2.2 sendo estes representados por e_2, e_3, e_4 e e_5.

Um evento sincronizante é caracterizado por alterar o estado de dois ou mais autômatos de forma simultânea. Sua ocorrência se dá em todos os autômatos envolvidos estabelecendo assim um sincronismo entre eles[20]. Na Figura 2.2 este evento é representado por e_1, onde pode ser notada sua presença em ambos os autômatos.

A seguir serão apresentados alguns conceitos referentes às taxas funcionais que também estabelecem relação entre os diferentes autômatos de um modelo.

2.2.2 Taxas Funcionais e Probabilidades Funcionais

A utilização de taxas funcionais e/ou probabilidades funcionais é outra forma de representar interações entre os autômatos de uma SAN sem alterar o estado de todos os autômatos envolvidos. Taxas funcionais podem ser definidas por funções que refletem a avaliação dos estados atuais do modelo. Na Figura 2.2 o evento e_2 do autômato $A^{(2)}$ apresenta uma taxa funcional sobre o autômato $A^{(1)}$, definida pela função f, descrita abaixo da figura. Neste caso, de acordo com a função f, a taxa do evento e_2 depende do estado que o autômato $A^{(1)}$ encontra-se, ou seja, a taxa do evento e_2 será igual a μ_1 caso o estado do autômato $A^{(1)}$ estiver em 0 ou igual a μ_2 caso estiver no estado 1.

No autômato $A^{(2)}$, pode-se também observar probabilidades para diferentes transições do evento e_3. Essas probabilidades representadas por π_1 e π_2 definem a probabilidade de escolha, ou seja,

[2] A nomenclatura da função de atingibilidade pode ser encontrada na documentação da ferramenta PEPS [30].

quando da ocorrência do evento e_3, o autômato $A^{(2)}$ estando no estado 2 irá para o estado 0 com probabilidade π_1 ou para o estado 1 com probabilidade π_2, consequentemente, $\pi_1 + \pi_2$ deve ser igual a 1. Tanto a taxa de ocorrência como a probabilidade de escolha podem ser definidas como valores constantes ou valores funcionais. Quando as taxas ou probabilidades são definidas como valores funcionais, estas são então ditas taxas funcionais ou probabilidades funcionais respectivamente.

Em resumo, a utilização de funções para definir taxas ou probabilidades permite associar a um mesmo evento diferentes valores. Assim sendo, as mesmas são expressas por funções que levam em consideração os estados atuais dos autômatos de um modelo variando, desta forma, seu valor conforme os estados em que se encontram os autômatos envolvidos na função [20].

2.2.3 Modelos

Nesta seção são exibidos alguns sistemas modelados com o formalismo SAN, os quais são utilizados para a realização dos testes deste trabalho.

Alternate Service Pattern - ASP

Padrão de Serviço Alternado (*Alternate Service Pattern* - ASP) é um sistema modelado com o formalismo de Redes de Filas de Espera [9] e consiste em uma rede aberta em que alguns servidores apresentam mais de um padrão de atendimento. A Figura 2.3 mostra um sistema ASP com quatro filas: F_1, F_2, F_3 e F_4; cada qual com capacidade finita K_1, K_2, K_3 e K_4 respectivamente.

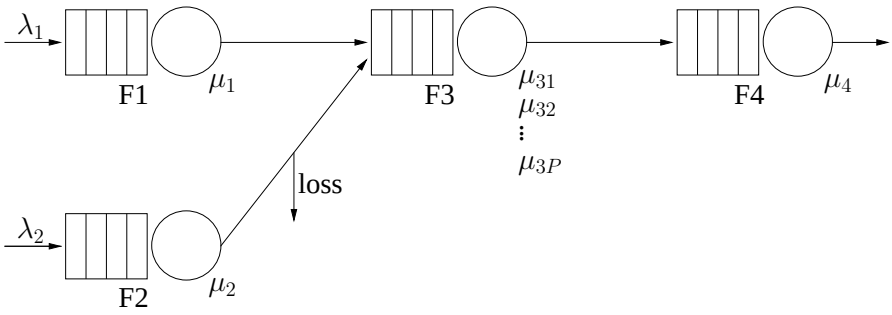


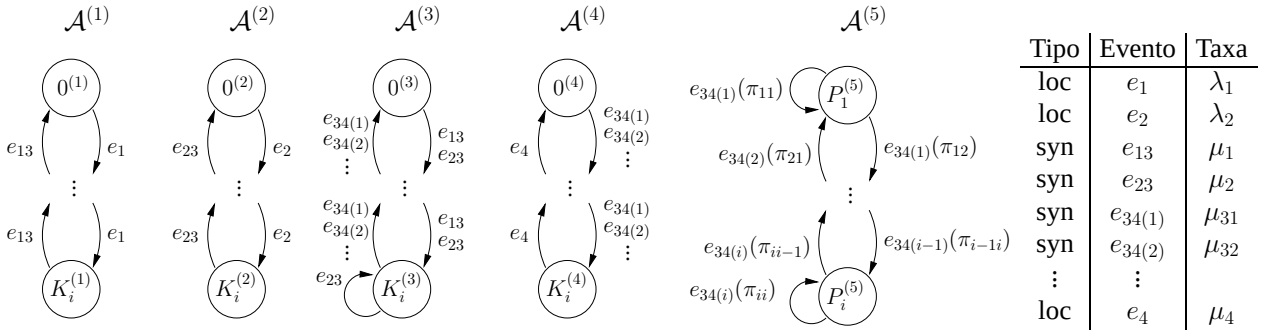
Figura 2.3: Sistema ASP modelado em Filas de Espera

A taxa de chegada nas filas F_1 e F_2 são λ_1 e λ_2, respectivamente. A fila F_1 atende a uma taxa μ_1 e faz roteamento de seus clientes para a fila F_3, caso ela possa recebê-los. Esse é o chamado *comportamento bloqueante*, visto que o cliente não deixa F_1 se F_3 não puder recebê-lo. O atendimento na fila F_2 acontece a uma taxa μ_2, devendo o cliente ser roteado para a fila F_3, caso esta tenha capacidade para recebê-lo. Caso F_3 não possa receber o cliente vindo de F_2, este deixa o sistema, sendo isso chamado *comportamento de perda*, representado na Figura 2.3 pela flecha indicada por *loss*.

A fila F_3 atende cada cliente segundo um dos P padrões de serviço que comporta, cujas taxas são μ_{31}, μ_{32}, ..., μ_{3P}. Os clientes somente deixam F_3 quando há espaço em F_4 para atendê-los, sendo

esse mais um caso de roteamento com comportamento bloqueante. Por fim, os clientes de F_4 são atendidos segundo a taxa μ_4 e deixam o sistema.

Esse mesmo sistema pode ser modelado usando o formalismo SAN. Para exemplificar, suponha a SAN da Figura 2.4, com quatro autômatos de K estados cada, mais um autômato de P estados. O PSS para este modelo é igual a $(K+1)^4 \times P$, sendo o mesmo igual ao RSS.



Tipo	Evento	Taxa
loc	e_1	λ_1
loc	e_2	λ_2
syn	e_{13}	μ_1
syn	e_{23}	μ_2
syn	$e_{34(1)}$	μ_{31}
syn	$e_{34(2)}$	μ_{32}
⋮	⋮	
loc	e_4	μ_4

Figura 2.4: *Alternate Service Pattern* - ASP

Agora analisamos mais atentamente a Figura 2.4 para entender como a representação desse sistema ASP modelado em SAN realmente funciona. As filas F_1, F_2, F_3 e F_4 passam a ser representadas pelos autômatos $A^{(1)}$, $A^{(2)}$, $A^{(3)}$ e $A^{(4)}$, respectivamente, cada um deles com capacidade K. Um quinto autômato, identificado por $A^{(5)}$, é usado para representar o padrão de serviço de F_3 que, segundo a figura, possui P estados, ou seja, P padrões de serviço com que a fila, representada pelo autômato A^3, atende seus clientes.

Os eventos de chegada nos autômatos $A^{(1)}$ e $A^{(2)}$ são representados pelos eventos e_1 e e_2, respectivamente, enquanto que o evento e_4 representa a saída do autômato $A^{(4)}$ para o exterior, sendo estes eventos locais.

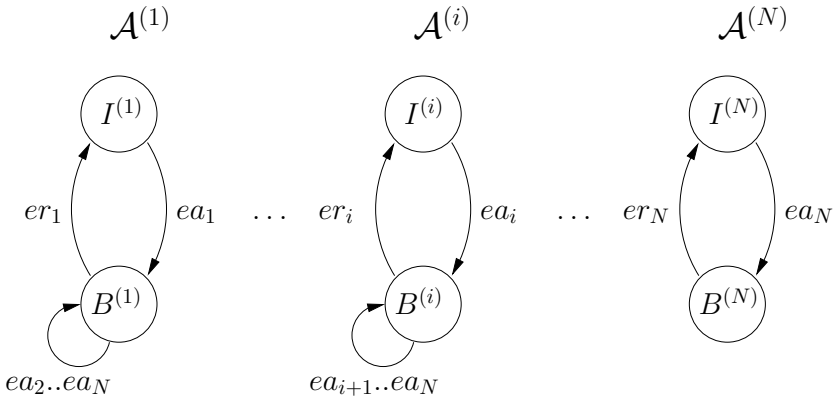
Os eventos indicados com índice duplo são os eventos sincronizantes, como é o caso do evento e_{13}, que corresponde à saída do autômato $A^{(1)}$ para o autômato $A^{(3)}$. O evento e_{23} pode corresponder a duas situações: a) o estado do autômato $A^{(3)}$ é diferente daquele correspondente à sua capacidade máxima, acontecendo então o roteamento do cliente do autômato $A^{(2)}$ para o autômato $A^{(3)}$; b) o estado do autômato $A^{(3)}$ é aquele de capacidade máxima, indicando que o cliente sai de $A^{(2)}$ direto para o exterior, caracterizando comportamento de perda, representado pelo *loop* no último estado do autômato $A^{(3)}$. Esse *loop* significa que o cliente saiu de $A^{(2)}$ mas não foi roteado para $A^{(3)}$, por isso $A^{(3)}$ mantém seu estado.

Os eventos restantes, $e_{34(1)}, e_{34(2)}, \ldots, e_{34(i)}$ representam o roteamento de um cliente do autômato $A^{(3)}$ para o autômato $A^{(4)}$, de acordo com os padrões de serviço $P_1, P_2, \ldots, P_i$ respectivamente. Por isso, esses eventos sincronizam três autômatos: $A^{(3)}$, $A^{(4)}$ e $A^{(5)}$, como visto na Figura 2.4.

***First Available Server* - FAS**

O modelo *First Available Server* (FAS) analisa a disponibilidade de N servidores, conforme pode ser notado no modelo SAN da Figura 2.5. Cada servidor (de índice i, onde $i \in [1..N]$) é representado por um autômato $A^{(i)}$, composto por dois estados: $I^{(i)}$ (*idle*/disponível) e $B^{(i)}$ (*busy*/ocupado). Neste exemplo os pacotes chegam nos servidores desde que pelo menos um deles não esteja ocupado. Este modelo pode ser visto como um quadro de análise de diferentes filas onde cada pacote na fila pode avançar para o primeiro servidor disponível. Sendo assim o pacote chega primeiro no servidor 1, se este não está disponível o pacote tenta o servidor 2, se este também não estiver disponível o pacote tenta o servidor 3 e assim sucessivamente até o servidor N.

Os eventos sincronizantes deste modelo são ea_i $(i = 2 \dots N)$ que representam a ocupação dos servidores. Os eventos locais são: ea_1 (chegada do pacotes) e er_i (para tornar os servidores disponíveis). Todos os eventos apresentam taxas constantes. O PSS do modelo é formado por 2^N estados, sendo o PSS também igual ao RSS.



Type	Event	Rate
loc	ea_1	λ
loc	er_1	μ
syn	ea_2	λ
loc	er_2	μ
⋮	⋮	⋮
syn	ea_N	λ
loc	er_N	μ

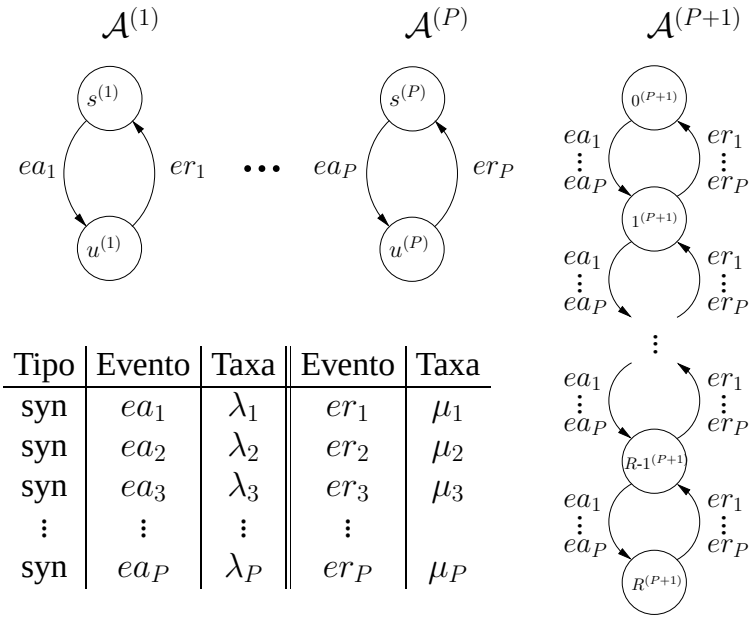
Figura 2.5: *First Available Server* - FAS

***Resource Sharing* - RS**

O modelo da Figura 2.6 apresenta um exemplo clássico, que representa um sistema de compartilhamento de recursos, sendo R o número de recursos e P o número de processos. Cada processo é representado por um autômato $A^{(i)}$ $(i = 1 \dots P)$ composto por dois estados: $S^{(i)}$ (em repouso) e $U^{(i)}$ (em uso). Os recurso são representados pelo autômato $A^{(P+1)}$ e tem $R + 1$ estados indicando o número de recursos em uso.

Este modelo apresenta apenas eventos sincronizantes, visto que os eventos ea_i representam a aquisição de um recurso com taxa constante λ_i e os eventos er_i representam a liberação de um recurso com taxa constante μ_i. O PSS deste modelo é formado por $2^P \times (R + 1)$ estados globais, porém ao contrário dos demais modelos nem todos os seus estados são atingíveis.

Utilizando métodos analíticos é possível resolver um grande número de modelos. No entanto,



Tipo	Evento	Taxa	Evento	Taxa
syn	ea_1	λ_1	er_1	μ_1
syn	ea_2	λ_2	er_2	μ_2
syn	ea_3	λ_3	er_3	μ_3
⋮	⋮	⋮	⋮	
syn	ea_P	λ_P	er_P	μ_P

Figura 2.6: *Resource Sharing* - RS

conforme aumenta o número de configurações possíveis dos sistemas modelados através de formalismos Markovianos como SAN esgota-se a capacidade de resolução destes métodos, visto que os limites computacionais são atingidos, principalmente no que diz respeito à memória exigida para armazenar os componentes que resolvem este formalismo (matriz e vetor). Devido a este problema, destaca-se então a simulação como uma alternativa para a resolução de modelos com grande número de configurações.

Capítulo 3

Simulação

Segundo Shannon [36] a simulação consiste em um processo de elaboração de um modelo de um sistema real (ou hipotético) e a condução de experimentos com a finalidade de entender o comportamento de um sistema ou avaliar sua operação. Para Law e Kelton [26], simulação corresponde a uma gama variada de métodos e aplicações que reproduzem o comportamento de sistemas reais, usualmente utilizando-se de ferramentas computacionais.

Através da simulação é possível manipular os componentes que constituem modelos Markovianos, para então encontrar o vetor de probabilidades, resultante da resolução dos mesmos. Como este vetor consiste em aproximações da solução iterativa, o tempo de simulação deve ser grande o suficiente para que os resultados possam convergir em uma solução próxima da estacionária, após uma série de execuções ou coletas de amostras. Isto é necessário, pois um dos grandes problemas da simulação é a precisão dos resultados, uma vez que a mesma consiste na tiragem de números pseudo-aleatórios para definição de variáveis, regidas por distribuições de probabilidades, conforme descrito a seguir.

3.1 Variáveis Aleatórias

Modelos estocásticos apresentam variáveis aleatórias para definir as taxas de transição entre os estados que os compõem. Praticamente todos os sistemas apresentam determinada fonte de aleatoriedade, como o tempo entre chegadas, tempos de serviço, tempo entre a ocorrência de uma falha, etc.

Pode-se definir variável aleatória como uma função ou regra que associa um número real a cada ponto do espaço amostral. Espaço amostral, por sua vez, é o conjunto de todos os resultados possíveis para um experimento [26]. A teoria dos grandes números, definida por Jacob Bernoulli em 1692 [7], estabelece que, em uma série imensa de experimentos, a freqüência relativa de um evento se aproxima cada vez mais da sua probabilidade. Assim, dada uma longa série de experimentos, pode-se, com erro desprezível, calcular a probabilidade de um evento, ou então, dada a probabilidade de um evento, pode-se calcular o número de vezes que ele pode ocorrer em uma longa série de tentativas.

A distribuição de probabilidade associa uma probabilidade a cada resultado numérico de um experimento, ou seja, dá a probabilidade de cada valor de uma variável aleatória. Por exemplo, no lançamento de um dado cada face tem a mesma probabilidade de ocorrência que é dada por $\frac{1}{6}$. Como os valores das distribuições de probabilidades são também probabilidades, e como as variáveis aleatórias devem assumir um de seus valores, têm-se duas regras a seguir que se aplicam a qualquer distribuição de probabilidade:

1. a soma de todos os valores de uma distribuição de probabilidade deve ser igual a 1, desta forma tem-se: $\sum V(x) = 1$, onde x abrange todos os valores possíveis;

2. a probabilidade de ocorrência de um evento deve ser maior ou igual a $zero$ e menor ou igual a 1, ou seja, $0 \leq V(x) \leq 1$ para todo valor de x.

No exemplo do lançamento de um dado, como todas as faces têm a mesma probabilidade de ocorrência, que é $\frac{1}{6}$, ao somá-las obtemos o valor 1, que corresponde à primeira regra citada anteriormente. O valor $\frac{1}{6}$ é maior do que $zero$ e menor do que 1, assim satisfaz também a segunda regra.

Para gerar a variável aleatória no processo de simulação utiliza-se a distribuição de probabilidade uniforme, descrita a seguir.

3.1.1 Distribuição Uniforme

A distribuição uniforme é uma distribuição cuja faixa de valores aleatórios estão entre duas variáveis a e b. Sua função densidade de probabilidade é constante dentro de um intervalo de valores da variável aleatória x. Cada um dos possíveis valores que x pode assumir seguindo esta distribuição tem a mesma probabilidade de ocorrer. Sua função de densidade é:

$$f(x) = \begin{cases} \frac{1}{b-a} & \text{se } a \leq x \leq b; \\ 0 & \text{caso contrário.} \end{cases}$$

e sua distribuição de probabilidade é dada por:

$$F(x) = \begin{cases} 0 & \text{se } x < a; \\ \frac{x-a}{b-a} & \text{se } a \leq x \leq b; \\ 1 & \text{se } b < x. \end{cases}$$

Os parâmetros a e b são números reais, sendo que $a < b$ e sua faixa de valores vão de a à b.

Discutida a geração de variáveis aleatórias, necessárias no processo de simulação empregado neste trabalho, é preciso compreender como o número aleatório gerado irá definir a transição na Cadeia de Markov. Esta transição é dada então através de uma função de transição, que estabelecerá o próximo estado a ser visitado, baseado nas taxas discretizadas (probabilidades) [38] da matriz correspondente ao modelo que deseja-se simular. Esta função de transição é definida a seguir.

3.2 Função de Transição

Neste trabalho, o vetor de probabilidades será obtido simulando o espaço de estados global de uma SAN, o qual corresponde ao estado na Cadeia de Markov que ela representa. Além disso, vale ressaltar que apenas o espaço de estados atingível (RSS) dos modelos são considerados no processo de simulação, o que reduz consideravelmente o número de estados em determinados modelos, como por exemplo o modelo RS, descrito na Seção 2.2.3. Assume-se então, um conjunto $S = \{s_0, s_1, s_2, ..., s_k\}$, o qual corresponde aos estados do modelo (sendo $k = |RSS|$), a matriz P composta pelas probabilidades de transição do mesmo e um gerador de números aleatórios $U(0..1)$ [21]. Este gerador é necessário, pois, a transição entre os estados na simulação, ocorre através de uma função de transição $\phi(s_i, U)$, a qual se baseia na tiragem (sorteio) de um valor uniformemente distribuído entre 0 e 1. O retorno da função ϕ consiste no intervalo que se apresenta a variável resultante do gerador de números aleatórios. Esta variável irá indicar a transição para o próximo estado da Cadeia de Markov durante a simulação, sendo s_i o estado corrente. A função de transição é definida pela expressão 3.1 a seguir e mostra como são realizadas estas transições entre os estados.

$$\phi(s_i, U) = \begin{cases} s_0 & \textit{para } U \in [0, P_{i0}) \\ s_1 & \textit{para } U \in [P_{i0}, P_{i0} + P_{i1}) \\ \vdots & \vdots \\ s_j & \textit{para } U \in \left[\sum_{l=0}^{j-1} P_{il}, \sum_{l=0}^{j} P_{il}\right) \\ \vdots & \vdots \\ s_r & \textit{para } U \in \left[\sum_{l=0}^{r-1} P_{ir}, 1\right] \end{cases} \tag{3.1}$$

Para exemplificar a aplicação desta expressão considere a matriz de probabilidades da Tabela 3.1.

Tabela 3.1: Matriz de Probabilidades

$P =$

i/j	0	1	2
0	0,10	0,65	0,25
1	0,25	0,55	0,20
2	0,30	0,25	0,45

A função de transição irá definir para qual estado s_i, para $i = \{0, 1, 2\}$, ocorrerá a transição de acordo com o intervalo indicado nas expressões 3.2, 3.3 e 3.4.

$$\phi(s_0, U) = \begin{cases} \text{s}_0 & \text{para } U \in [0, 0, 10) \\ \text{s}_1 & \text{para } U \in [0, 10, (0, 10 + 0, 65)) \\ \text{s}_2 & \text{para } U \in [(0, 10 + 0, 65), 1] \end{cases} \tag{3.2}$$

$$\phi(s_1, U) = \begin{cases} \text{s}_0 & \text{para } U \in [0, 0, 25) \\ \text{s}_1 & \text{para } U \in [0, 25, (0, 25 + 0, 55)) \\ \text{s}_2 & \text{para } U \in [(0, 25 + 0, 55), 1] \end{cases} \tag{3.3}$$

$$\phi(s_2, U) = \begin{cases} \text{s}_0 & \text{para } U \in [0, 0, 30) \\ \text{s}_1 & \text{para } U \in [0, 30, (0, 30 + 0, 25)) \\ \text{s}_2 & \text{para } U \in [(0, 30 + 0, 25), 1] \end{cases} \tag{3.4}$$

Tendo como base a expressão 3.4 e assumindo a variável aleatória retornada da função U igual a $0, 40$, a transição ocorrerá do estado s_2 para o estado s_1. Isto acontece pois a mesmo encontra-se entre o intervalo $0, 30$ e $0, 55$ $(0, 30 + 0, 25)$ indicado na expressão.

3.3 Técnicas de Simulação

Definida a função de transição, três técnicas de simulação são empregadas neste trabalho para a resolução de modelos. Dentre elas estão a *Forward Simulation*, a *Backward Coupling Simulation* e a *Permanence Time Simulation*. Estas técnicas são descritas a seguir.

3.3.1 *Forward Simulation*

Nesta técnica de simulação é considerado um estado inicial arbitrário e em seguida são disparadas transições entre os estados do modelo por uma quantidade de vezes pré-estabelecida, de forma a realizar uma trajetória na Cadeia de Markov [21]. O tamanho ideal desta trajetória ainda é desconhecido, pois pode variar dependendo da cardinalidade do espaço de estados, que corresponde a dimensão das matrizes de transição. Entretanto esta trajetória deve ser grande o suficiente para que todos os estados da Cadeia de Markov sejam atingidos. Após várias transições, o estado final da trajetória é coletado como uma amostra da simulação. A quantidade ideal de amostras a serem coletadas é um valor ainda desconhecido, no entanto quanto maior esse número melhor tende a ser a precisão dos resultados.

O vetor de probabilidades é calculado contabilizando a proporção das amostras coletadas no final da simulação. A Figura 3.1 ilustra uma trajetória, disparada para a coleta de uma amostra (representada pelo estado circulado com traço contínuo) dada uma função de transição estabelecida $\phi(s_i, U)$, sendo o estado s_i, da Cadeia de Markov, representado no eixo vertical da figura. É importante notar que nesta técnica a trajetória ocorre a partir do tempo $zero$, parando em um tempo t pré-definido. O tempo, no processo de simulação, corresponde ao número de transições (passos) realizadas na Cadeia de Markov, uma vez que, a matriz de taxas de transição é discretizada [38]. Desta forma, cada unidade de tempo t corresponde a um passo/transição na Cadeia de Markov. Vale destacar que a trajetória realizada na Figura 3.1 baseia-se na matriz de probabilidades da Tabela 3.1, respeitando os intervalos das equações 3.2, 3.3 e 3.4 exemplificadas anteriormente.

Nas simulações realizadas neste trabalho, o estado inicial foi definido arbitrariamente como 0, conforme pode ser observado na linha 3 do Algoritmo 3.1 e o tamanho da trajetória foi fixado, também arbitrariamente, em 10.000 transições. Ao final da trajetória a amostra é então coletada e contabilizada, conforme visto na linha 8 do algoritmo.

Terminada a simulação, o vetor de probabilidades é então calculado dividindo o número de amostras coletadas em cada um dos i estados do modelo pelo quantidade total de amostras.

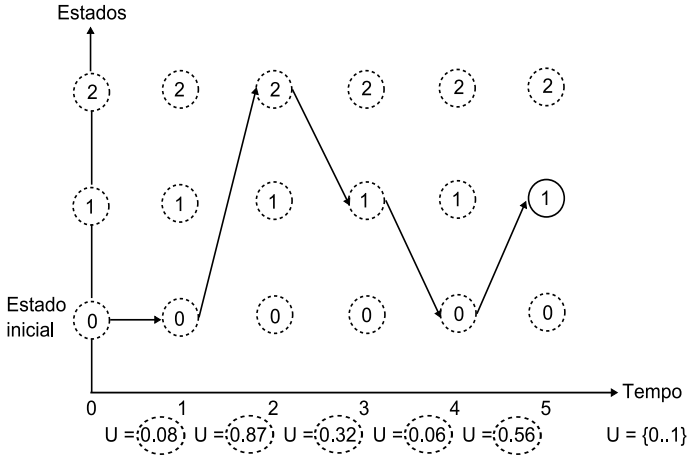


Figura 3.1: Trajetória da técnica *Forward Simulation*

Algorithm 3.1: Representação do algoritmo para a técnica *Forward Simulation*

1: $\pi \leftarrow 0$ $\{inicializa\ todas\ as\ posições\ do\ vetor\ de\ probabilidades\ \pi\}$
2: **while** $critério\ de\ parada$ $\{número\ pré-definido\ de\ amostras\}$ **do**
3: $\quad i \leftarrow 0$ $\{escolha\ de\ um\ estado\ inicial\}$
4: $\quad$ **while** $critério\ de\ parada$ $\{número\ pré-definido\ de\ passos/transições\}$ **do**
5: $\quad\quad \delta \leftarrow U(0..1)$ $\{geração\ de\ um\ número\ aléatório\ entre\ 0\ e\ 1\}$
6: $\quad\quad s_i \leftarrow \phi(s_i, \delta)$ $\{função\ que\ define\ a\ próxima\ transição\}$
7: $\quad$ **end while**
8: $\quad \pi[s_i] = \pi[s_i] + 1$ $\{contabiliza\ amostra\ gerada\}$
9: **end while**
10: **return** π

3.3.2 *Backward Coupling Simulation*

Esta técnica é baseada no algoritmo *Coupling from the Past* (CFTP) proposto por Propp e Wilson em 1996 [33]. A técnica foi revolucionária na época dentro do campo de aplicação na física para resolver alguns dos problemas de simulação. Nesta técnica não é necessário escolher um estado inicial uma vez que várias trajetórias são disparadas em paralelo partindo uma de cada estado do modelo. Outra vantagem é que não é necessário definir o tamanho da trajetória, pois a amostra é coletada quando todas as trajetórias (disparadas em paralelo) se encontram em um mesmo estado no

tempo $zero$ de simulação. Isso ocorre, pois as mesmas são disparadas "do passado", observando os estados predecessores, ou seja, os estados que trouxeram ao estado atual. Esta técnica destaca-se também, por permitir a coleta de amostras perfeitamente distribuídas de acordo com a distribuição estacionária do processo Markoviano, não produzindo amostras tendenciosas. A Figura 3.2 ilustra este processo, apresentando as trajetórias disparadas a partir de cada estado, no passado, parando no tempo -5 de simulação.

O eixo horizontal, de cada sub-figura, representa o tempo que corresponde ao número de transições necessárias até que todos os estados (eixo vertical) se encontrem no tempo $zero$. As linhas contínuas apresentam as trajetórias que levam ao mesmo estado, enquanto as linhas pontilhadas correspondem às trajetórias que não se encontram no mesmo estado. O Algoritmo 3.2, ajuda a compreender este processo, sendo o vetor ω da linha 9 o responsável por armazenar o estado referente a transição de cada trajetória. Quando este vetor armazenar transições para o mesmo estado em todas as suas posições, o mesmo será então a amostra a ser coletada, conforme a linha 13 do algoritmo.

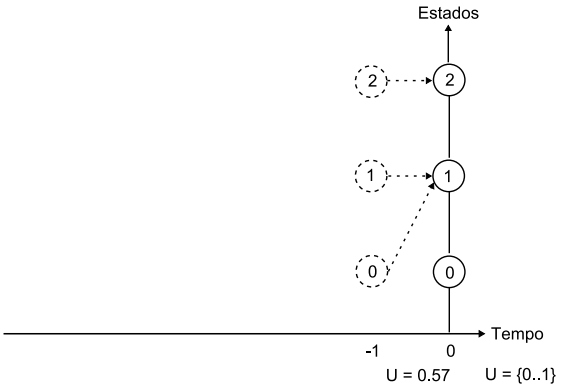
Algorithm 3.2: Representação do algoritmo para a técnica *Backward Coupling Simulation*

1: $\pi \leftarrow 0$ $\{inicializa\ todas\ as\ posições\ do\ vetor\ de\ probabilidades\ \pi\}$
2: **while** $critério\ de\ parada$ $\{número\ pré-definido\ de\ amostras\}$ **do**
3: **for all** $s_0, s_1, s_2, ..., s_i$ $\{s_i\ são\ todos\ os\ estados\ do\ modelo\}$ **do**
4: $\varpi[s_i] \leftarrow i$ $\{inicializando\ trajetórias\ em\ todos\ os\ estados\ no\ vetor\ auxiliar\ \varpi\}$
5: **repeat**
6: **for all** $s_0, s_1, s_2, ..., s_i$ $\{s_i\ são\ todos\ os\ estados\ do\ modelo\}$ **do**
7: $\omega[s_i] \leftarrow i$ $\{inicializando\ trajetórias\ em\ todos\ os\ estados\ no\ vetor\ \omega\}$
8: $\delta \leftarrow U(0..1)$ $\{geração\ de\ um\ número\ aléatório\ entre\ 0\ e\ 1\}$
9: $\omega[s_i] \leftarrow \varpi[\phi(s_i, \delta)]$ $\{função\ que\ define\ as\ transições\ de\ todos\ os\ estados\}$
10: **for all** $s_0, s_1, s_2, ..., s_i$ **do**
11: $\varpi[s_i] \leftarrow \omega[s_i]$ $\{salvando\ os\ estados\ de\ cada\ trajetória\ no\ vetor\ auxiliar\}$
12: **until** $critério\ de\ parada$ $\{todas\ as\ trajetórias\ se\ encontram\ em\ um\ único\ estado\}$
13: $\pi[\omega[s_0]] = \pi[\omega[s_a]] + 1$ $\{amostra\ gerada\}$
14: **end while**
15: **return** π

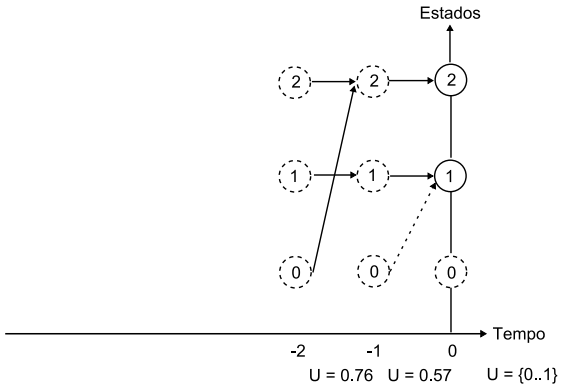
3.3.3 *Permanence Time Simulation*

Esta técnica, ao contrário das anteriores, consiste apenas na execução de uma única trajetória. Nela, as amostras de estados são coletadas após cada transição, sendo assim é contabilizada a quantidade de vezes que um estado é visitado na Cadeia de Markov. Observando a Figura 3.1, o número de transições para o estado 1 ocorre duas vezes, enquanto que para o estado 2 ocorre apenas uma vez, por exemplo.

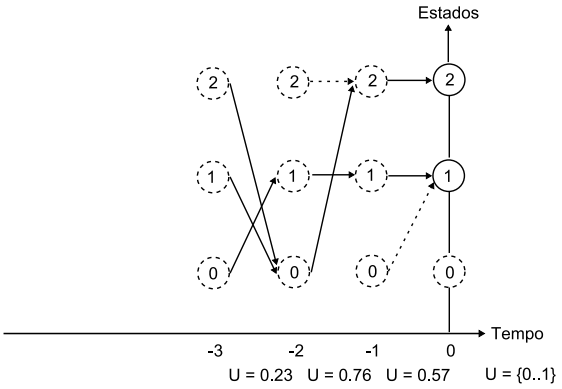
Da mesma forma que nas soluções anteriores, para que uma solução estacionária seja aproximada, o número de amostras deve ser significativamente grande para que todos os estados sejam visitados uma grande quantidade de vezes para que as proporções possam ser calculadas. O Algoritmo 3.3



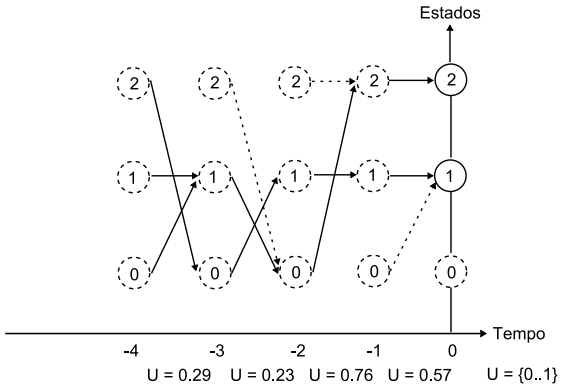
(a) Tempo -1



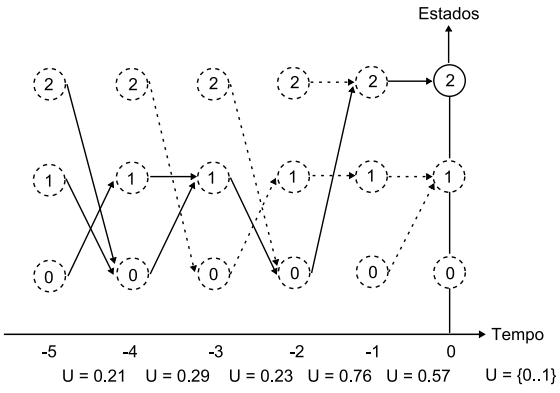
(b) Tempo -2



(c) Tempo -3



(d) Tempo -4



(e) Tempo -5

Figura 3.2: Trajetórias da técnica *Backward Coupling Simulation* encontrando-se no tempo 0 após 5 passos no passado

facilita essa análise, sendo que na linha 6 do algoritmo, o estado visitado é incrementado na posição correspondente no vetor π, após cada transição, da linha 5. Cada estado visitado, corresponde então a uma amostra da simulação.

Algorithm 3.3: Representação do algoritmo para a técnica *Permanence Time Simulation*

1: $\pi \leftarrow 0$ $\{inicializa\ todas\ as\ posições\ do\ vetor\ de\ probabilidades\ \pi\}$
2: $i \leftarrow 0$ $\{escolha\ de\ um\ estado\ inicial\}$
3: **while** $critério\ de\ parada$ $\{número\ pré-definido\ de\ passos/transições\}$ **do**
4: $\quad \delta \leftarrow U(0..1)$ $\{geração\ de\ um\ número\ aléatório\ entre\ 0\ e\ 1\}$
5: $\quad s_i \leftarrow \phi(s_i, \delta)$ $\{função\ que\ define\ a\ próxima\ transição\}$
6: $\quad \pi[s_i] = \pi[s_i] + 1$ $\{estado\ visitado\ incrementado\ na\ posição\ correspondente\ de\ \pi\}$
7: **end while**
8: **return** $\pi\{vetor\ com\ as\ amostra\ geradas\}$

Capítulo 4

Testes Realizados e Resultados Obtidos

Para avaliar a precisão da simulação, desenvolveu-se um simulador para as três técnicas mencionadas na Seção 3.3. O simulador recebe como entrada uma matriz de transição e armazena em um arquivo de saída o vetor de probabilidades resultante. Esta matriz de transição corresponde à Cadeia de Markov equivalente aos modelos SAN apresentados na Seção 2.2.3. Neste trabalho, para o modelo ASP foi utilizado $K = 2$ e $P = 2$, ou seja, fila com capacidade de dois clientes e dois padrões de serviço, resultando em um PSS e RSS de $(2+1)^4 \times 2 = 162$ estados. Para o modelo FAS foi utilizado o número de servidores N igual a 9, o que resulta em um PSS e RSS de $2^9 = 512$ estados. Para o modelo RS utilizou-se 10 processos P e 5 recursos R, sendo seu PSS igual a $2^{10} \times (5 + 1) = 6.144$ e o RSS igual a 638.

Assim como a obtenção do RSS, as conversões dos modelos SAN em Cadeias de Markov foram realizadas utilizando a ferramenta PEPS, cujos arquivos de entrada de cada modelo podem ser observados no Anexo A. Esta ferramenta disponibiliza a matriz equivalente em um formato HBF (*Harwell-Boeing Format*) [37], o qual armazena apenas os valores não nulos com seus respectivos índices na matriz. Este formato é utilizado para reduzir o tamanho do arquivo gerado, já que normalmente a matriz de transição é bastante esparsa, sendo desnecessário armazenar valores nulos.

Além da matriz de transição obtida a partir dos modelos descritos anteriormente é passado ao simulador, como argumento, o número de amostras desejadas e a técnica de simulação a ser utilizada. Após a execução é gerado um arquivo de saída contendo o vetor de probabilidades de permanência em cada estado do modelo.

A análise dos resultados é feita então através da comparação do vetor resultante das simulações com o vetor da solução analítica obtido com o Método da Potência [9, 37], implementado na ferramenta PEPS. Os testes foram realizados resolvendo o mesmo modelo em ambas as soluções para que as comparações pudessem ser realizadas. Estes testes e seus respectivos resultados são apresentados a seguir.

4.1 Erro Relativo Variando o Número de Amostras

Este teste varia a quantidade de amostras coletadas ao aplicar cada técnica de simulação, e posteriormente compara seus resultados com os resultados obtidos com o Método da Potência, para os modelos ASP, FAS e RS, já descritos.

De posse desses resultados foi calculado o erro relativo de cada uma das probabilidades contidas no vetor $\pi_i^{(s)}$ resultante da simulação, com a respectiva probabilidade do vetor $\pi_i^{(a)}$ do método iterativo (Método da Potência), as quais correspondem a probabilidade de permanência em cada estado do modelo. O erro relativo γ_i foi então calculado de acordo com a equação 4.1, sendo i cada um desses estados.

$$\gamma_i = \left| \frac{\pi_i^{(a)} - \pi_i^{(s)}}{\pi_i^{(a)}} \right| \tag{4.1}$$

Feito isso, foi efetuada a média aritmética $\tilde{x}$ entre todos estes erros relativos (γ_i) conforme a expressão 4.2 (sendo $\tilde{x} \times 100$ o erro percentual) e destacado também o erro relativo máximo γ_{max} entre eles.

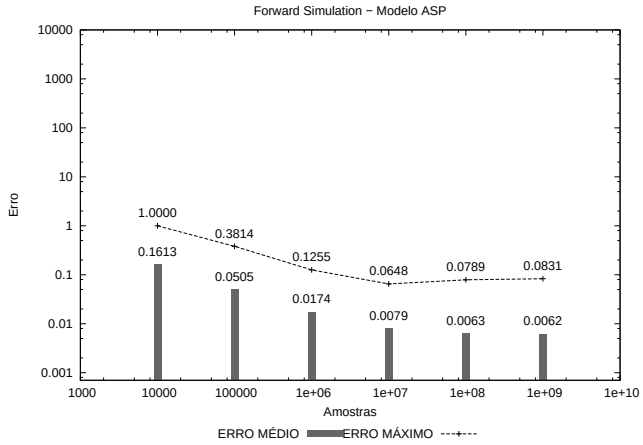
$$\tilde{x} = \frac{\gamma_1 + \gamma_2 + \gamma_i + \cdots + \gamma_n}{n} = \frac{1}{n} \sum_{i=1}^{n} \gamma_i \tag{4.2}$$

Esta operação foi feita com o resultado de todas as técnicas de simulação implementadas e foi repetida com os três modelos SAN mencionados (ASP, FAS e RS). Os resultados são apresentados nas Figuras 4.1, 4.2 e 4.3, sendo que o eixo x de cada gráfico corresponde a quantidade de amostras coletadas em cada execução e o eixo y o erro relativo. A linha pontilhada representa o erro relativo máximo dentre os estados e as barras correspondem ao erro relativo médio entre eles.

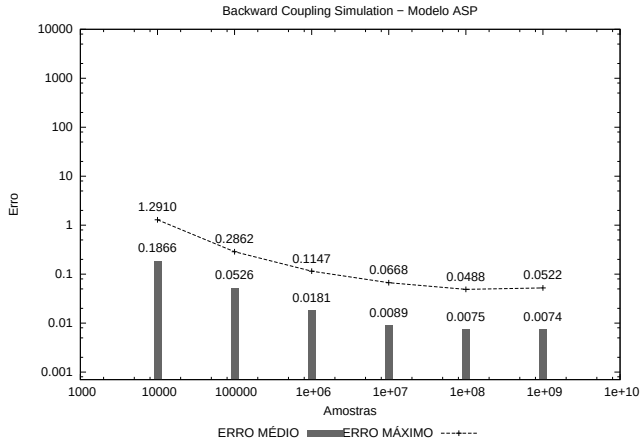
Note que conforme aumenta o número de amostras (eixo x) coletadas na resolução dos modelos, diminui o erro relativo médio e máximo dos resultados da simulação, praticamente em todas as técnicas apresentadas. Quanto à precisão das mesmas, pode-se dizer que apresentam melhor eficiência a partir dos resultados com $10.000.000$ ($1e+07$) de amostras. Isso pode ser notado uma vez que o erro relativo percentual do modelo ASP, considerando todas as técnicas, varia entre $0,79\%$ (técnica *Forward Simulation*) e $1,10\%$ (técnica *Permanence Time Simulation*), enquanto que o erro relativo máximo varia entre $6,4\%$ (técnica *Forward Simulation*) e $10,3\%$ (técnica *Permanence Time Simulation*).

Um comportamento importante a ser destacado neste modelo é que as técnicas de simulação *Backward*, *Forward* e *Permanence Time* apresentam pouca redução do erro para resultados com mais de $10.000.000$ ($1e+07$) de amostras, ou seja, a redução da média do erro relativo é praticamente nula quando a ordem de grandeza das amostras coletadas aumenta mais do que isso.

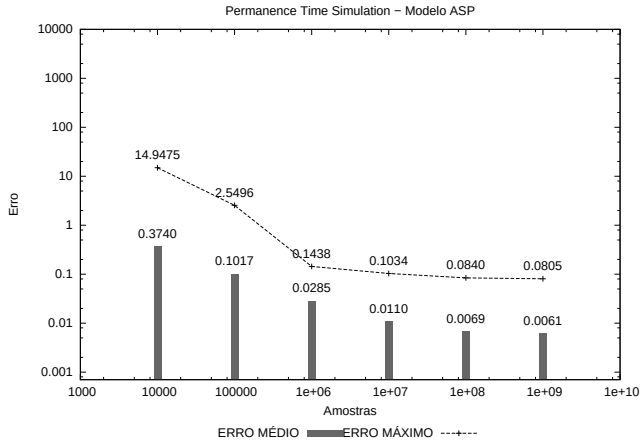
Devido ao comportamento observado anteriormente, nota-se que aumentar a quantidade de amostras coletadas na ordem de grandeza, pode não compensar o custo correspondente ao tempo de pro-



(a) Forward

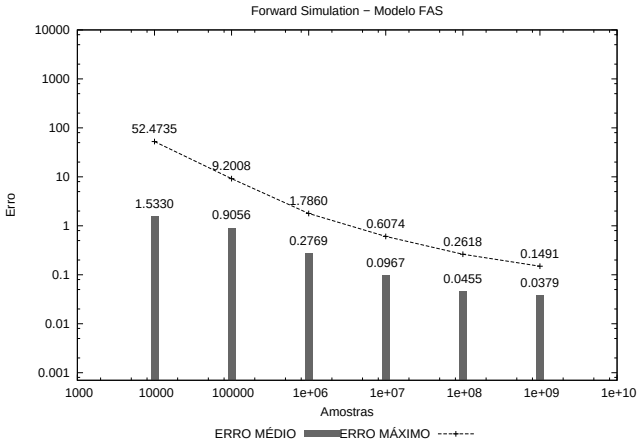


(b) Backward

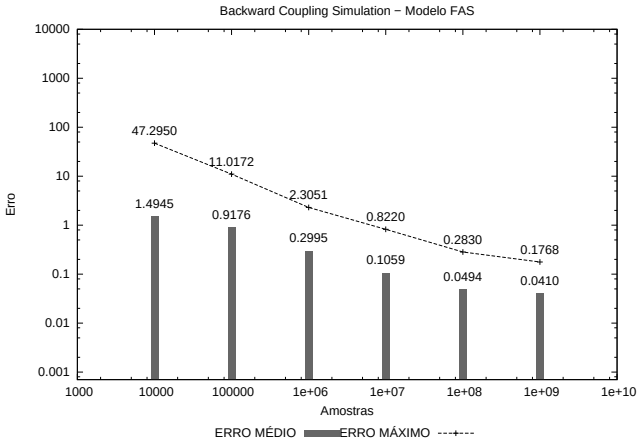


(c) Permanence Time

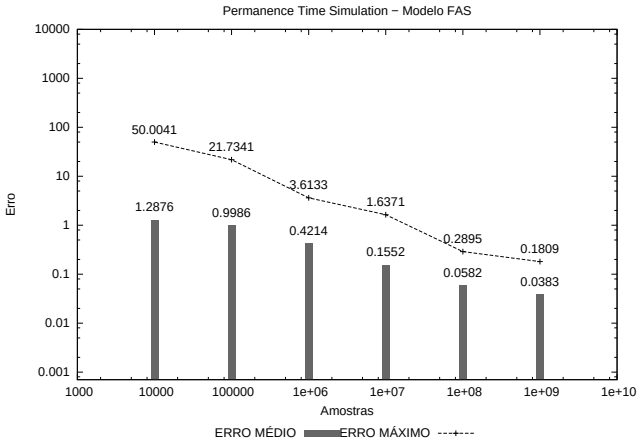
Figura 4.1: Erros relativos médios e máximos entre os resultados da aplicação das técnicas de simulação e o resultado da solução analítica para o modelo ASP com $K = 2$ e $P = 2$



(a) Forward

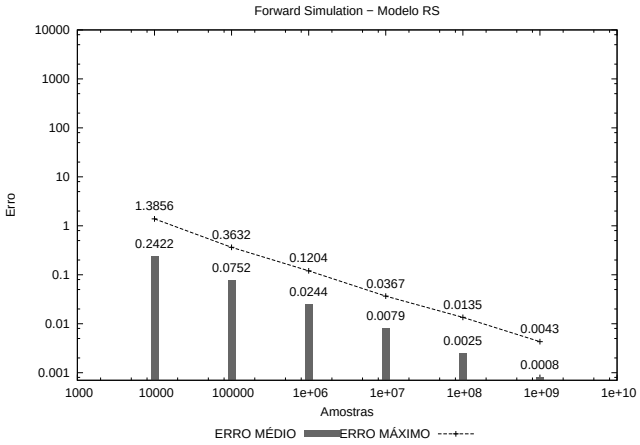


(b) Backward

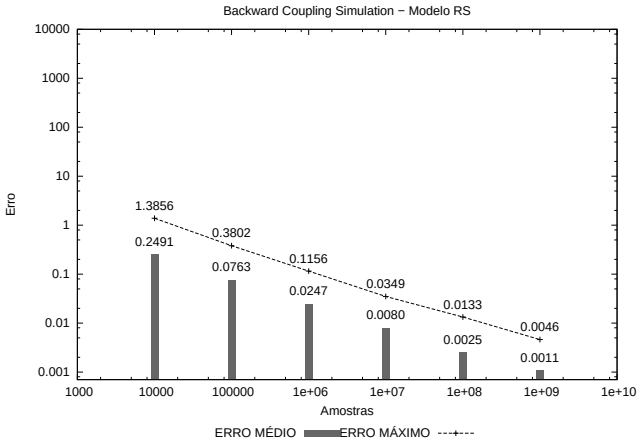


(c) Permanence Time

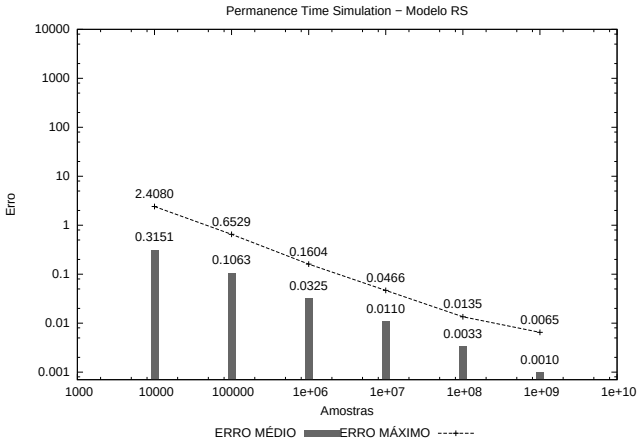
Figura 4.2: Erros relativos médios e máximos entre os resultados da aplicação das técnicas de simulação e o resultado da solução analítica para o modelo FAS com 9 servidores



(a) Forward



(b) Backward



(c) Permanence Time

Figura 4.3: Erros relativos médios e máximos entre os resultados da aplicação das técnicas de simulação e o resultado da solução analítica para o modelo RS com 10 processos e 5 recursos

cessamento que normalmente é despendido, visto que para algumas técnicas esse tempo pode ser bastante grande e a redução do erro pouco significativa. Os tempos de processamento das técnicas de simulação serão apresentados e discutidos na Seção 4.5.

Já para o segundo modelo analisado (FAS) pode ser observado um comportamento equilibrado em relação a todas as técnicas aplicadas. No entanto os erros relativos médios e máximos do modelo FAS são maiores que no modelo ASP, isso pode ser justificado, pois o RSS do modelo FAS é de 512 estados, contra 162 do modelo ASP. Acredita-se que devido a esta diferença no número de estados seria necessário uma quantidade maior de amostras do modelo FAS para atingir a mesma precisão obtida com o modelo ASP.

Com relação ao modelo RS, mesmo este apresentando um RSS ligeiramente maior que o do modelo FAS, de 636 estados (contra 512), o mesmo apresenta erros relativos médios e máximos bem menores. Um dos possíveis motivos que podem ter acarretado neste resultado é a influência das taxas de transição que os compõe. Neste sentido, a ocorrência de eventos raros, ou com probabilidades muito baixas de ocorrência, pode ser a causa de discrepâncias mais acentuadas nos resultados de alguns modelos. A simulação de modelos com eventos de ocorrência rara é um tema de pesquisa em aberto, com poucos trabalhos relacionados [24]. No entanto, mesmo esta característica tendo relevância nos resultados, a análise da mesma será proposta como um estudo futuro, sendo que não foram realizados experimentos que avaliem esta propriedade no momento.

Uma característica bastante importante que não pode deixar de ser mencionada é a precisão da técnica *Backward Coupling Simulation*. Esta técnica de simulação é considerada uma das mais precisas, por coletar amostras mais perfeitamente distribuídas. Porém, os resultados mostraram que a mesma apresenta ganhos pouco significativos comparada com as demais, onde em praticamente todos os casos apresenta erro relativo médio maior que a técnica *Forward Simulation*, com exceção do primeiro ponto ($1,5330$ contra $1,4945$) do gráfico do modelo FAS (Figura 4.2) e do quinto ponto (ambos com erro médio igual a $0,0025$) do gráfico do modelo RS (Figura 4.3). Vale lembrar que esta última necessita a escolha de um estado inicial e a definição do tamanho da trajetória, ao contrário da *Backward Coupling Simulation*. Além disso, se compararmos o consumo de memória, a técnica *Backward Coupling Simulation* necessita armazenar trajetórias partidas de todos os estados do modelo, fato que acarreta um consumo elevado, quando comparada com as outras duas. Alguns estudos recentes vêm sendo realizados para reduzir esse consumo de memória e acelerar a coleta das amostras através de propriedades de monotonicidade, conforme visto em Vincent [39]. Porém, essa propriedade só é aplicada para uma determinada classe de modelos, através de técnicas não triviais.

Destaca-se também que os erros apresentados nos gráficos foram calculados a partir de uma única execução, sendo que na simulação este resultado pode apresentar determinada variância quando executado outras vezes. Isso ocorre devido a influência na escolha da semente utilizada no gerador de números aleatórios. Para verificar esta variação, foi realizado o intervalo de confiança com a aplicação da técnica *Permanence Time Simulation* (escolhida por apresentar menor tempo de processamento), para várias execuções, conforme será apresentado a seguir.

4.2 Intervalos de Confiança

O Intervalo de confiança [9] é utilizado para verificar a variação de cada execução da simulação, de forma a observar o quanto o resultado de uma execução difere de outro. Em outras palavras, o intervalo de confiança dá a margem de erro da simulação, necessitando para isso a realização de várias execuções.

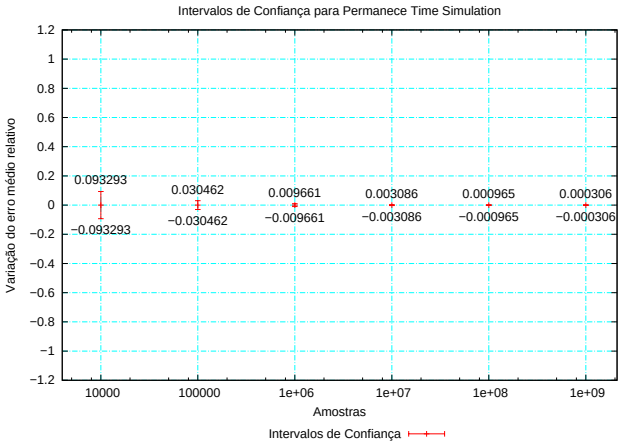
A variação entre os resultados ocorre, pois na simulação as variáveis que disparam as transições entre um estado e outro são variáveis aleatórias definidas por distribuições de probabilidades, conforme visto na Seção 3.1. Computacionalmente, estas variáveis são definidas por funções que recebem como parâmetro uma semente, que corresponde a um determinado valor real passado como parâmetro para a função. O uso de diferentes sementes faz com que haja variações entre um resultado e outro da simulação, pois serve como ponto inicial para a geração das variáveis aleatórias. Por este motivo, estas variáveis são na verdade chamadas de pseudoaleatórias [26], visto que não são totalmente aleatórias, justamente por serem geradas através de uma função, pelo computador.

Para mostrar este comportamento, foram realizadas 50 execuções dos modelos ASP, FAS e RS, utilizando a técnica *Permanence Time Simulation*. Feito isso, foi calculado o intervalo de confiança de 95% para cada estado em todos os modelos. Em cada execução foi passado como parâmetro uma semente diferente ao gerador de números aleatórios e a média aritmética dos intervalos obtidos são apresentados nos gráfico da Figura 4.4.

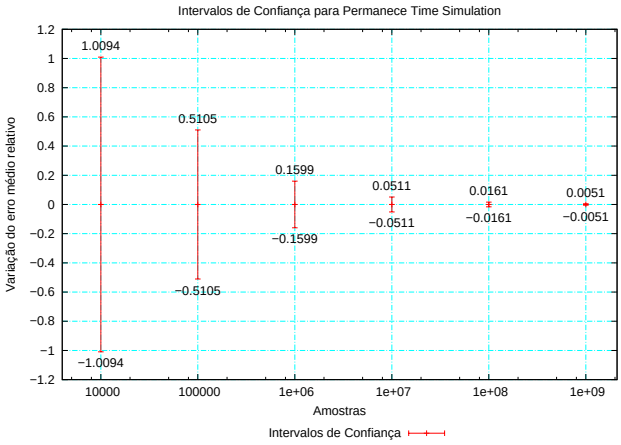
O eixo x dos gráficos apresenta a quantidade de amostras coletadas em cada uma das 50 execuções da simulação, ou seja, 50 execuções com 10.000 amostras, 50 execuções com 100.000 amostras e assim por diante, para cada modelo. O eixo y mostra a variação do erro médio relativo entre as execuções, onde nota-se a significativa redução do mesmo à medida que a quantidade de amostras aumenta no eixo x, principalmente para o modelo FAS. Constata-se com isso que a média dos intervalos de confiança realizada entre os estados de cada modelo, torna-se mais precisa, conforme essas amostras aumentam, visto que um intervalo de confiança menor corresponde a pouca variação entre os resultados de diferentes execuções. Este fato aumenta a credibilidade das simulações conforme aumenta o número de amostras coletadas uma vez que a semente utilizada no gerador de números aleatórios passa a influenciar cada vez menos no resultado.

Como a mesma quantidade de amostras foi coletada para as demais técnicas, acredita-se que a variação apresentada nos gráficos anteriores seja a mesma para todas elas, não sendo necessário repetir o teste para cada uma novamente. Além disso, isso seria inviável neste momento, visto que as técnicas *Backward Coupling Simulation* e *Forward Simulation* apresentam um tempo de processamento significativamente superior à técnica *Permanence Time Simulation*, conforme será apresentado na Seção 4.5.

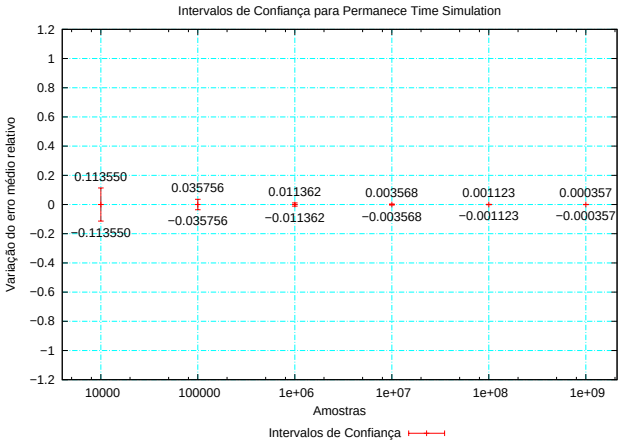
Além da semente utilizada na simulação, fatores como a escolha de um estado inicial, e a definição de diferentes tamanhos de trajetória, como no caso da técnica *Forward Simulation*, podem também apresentar perda de precisão. Para avaliar a influência na escolha do estado inicial e no tamanho da



(a) ASP com K = 2 e P = 2



(b) FAS com N = 9



(c) RS com P = 10 e R = 5

Figura 4.4: Intervalos de confiança com 50 execuções da simulação dos modelos ASP, FAS e RS com a técnica *Permanence Time Simulation*

trajetória desta técnica e também pelo fato da mesma ter se mostrado ligeiramente melhtécnicaor que a técnica *Backward Coupling Simulation* em determinados pontos dos gráficos, foram feitos alguns testes, descritos na próxima seção.

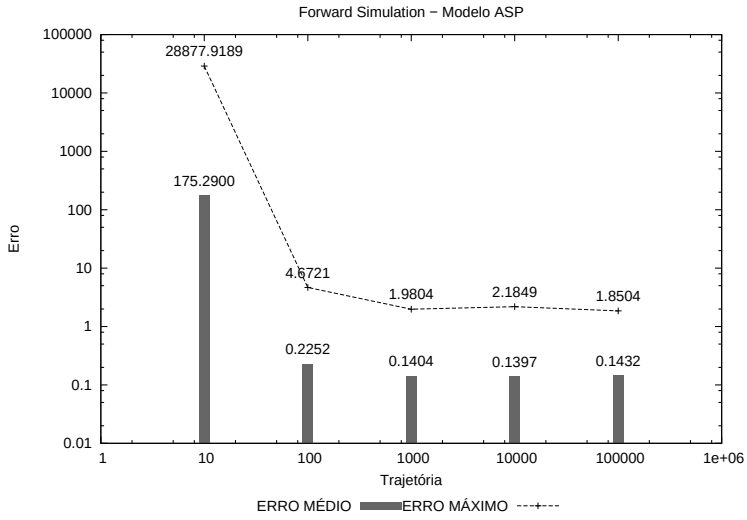
4.3 Tamanho da Trajetória da Técnica Forward Simulation

Uma das grandes desvantagens da técnica *Forward Simulation* é ter que definir o tamanho da trajetória, conforme visto na Seção 3.3.1 e também ter que definir o estado inicial para o disparo da mesma. Assim sendo, a escolha de uma trajetória muito grande aumenta a carga computacional acarretando em um tempo de processamento elevado, enquanto que uma trajetória muito pequena, pode comprometer a precisão dos resultados. Para avaliar este impacto, foram realizados experimentos variando o tamanho da mesma e também seu estado inicial, na coleta de $1.000.000$ de amostras do modelo ASP com $K = 4$ e $P = 4$, totalizando um RSS de 2.500 estados.

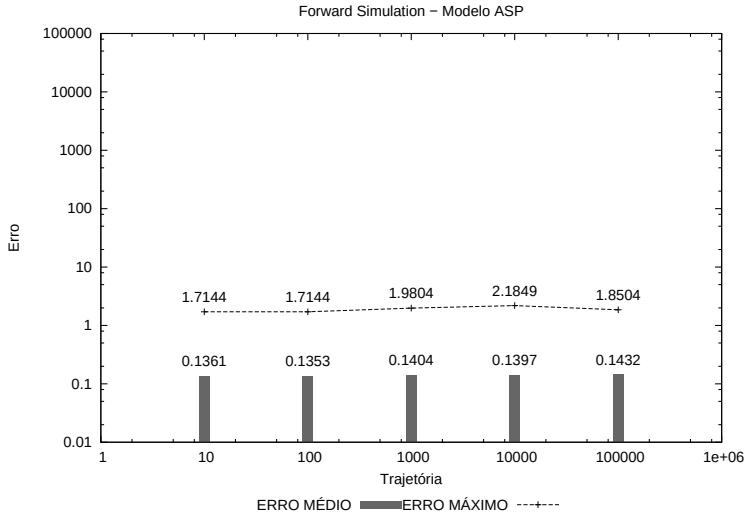
Os gráficos da Figura 4.5 mostram o erro relativo médio (eixo y), variando o tamanho das trajetórias de acordo com o eixo x, sendo que o gráfico (a) da figura mostra os resultados com as trajetórias iniciando todas no estado 0. Neste gráfico pode-se dizer que o estado inicial fixo apresenta influência apenas para as trajetórias de tamanho 10 e 100, visto que o erro estabiliza conforme ela cresce, caracterizando certa estacionariedade. Este comportamento permite deduzir que o estado inicial torna-se irrelevante conforme o tamanho da trajetória aumenta, assim como o ganho em relação a precisão mantêm-se praticamente nulo após aumentá-la mais do que 1.000.

Pensando em uma forma de reduzir este erro foram realizados experimentos com diferentes estados iniciais. Para definí-los foi disparada a primeira trajetória partindo do estado 0 e a trajetória seguinte utilizou a amostra coletada pela primeira (que corresponde ao último estado visitado) como sendo seu estado inicial e assim sucessivamente. O resultado é mostrado no gráfico (b) da Figura 4.5 e apresenta um erro relativo médio significativamente menor que o apresentado no gráfico (a) para trajetórias menores que 1.000. Embora este erro reduza rapidamente conforme a trajetória aumenta (mais do que 100), esta primitiva pode minimizar o erro quando a mesma for fixada em um tamanho muito pequeno, erroneamente. Sugere-se, no entanto, que seu tamanho seja no mínimo maior que o RSS do modelo, caso contrário se todas as trajetórias partirem do estado $zero$, como no caso do gráfico (a) da figura, alguns estados podem nunca ser visitados (atingidos).

Outra questão que permanece em aberto tanto para a técnica *Forward Simulation*, quanto para as demais, é a quantidade de amostras necessárias para se obter uma determinada precisão. Esta questão é difícil de ser respondida, porém, algumas primitivas podem ser utilizadas para auxiliar nesta inferência, mesmo que superficialmente. Uma delas é analisando o intervalo de confiança apresentado na Seção 4.2, sendo que um número de amostras que apresente um intervalo de confiança pequeno pode ser um bom palpite para a obtenção de um resultado preciso, definido de acordo com a necessidade do usuário. Outra primitiva que permite auxiliar na definição da quantidade de amostras é a distribuição do erro, conforme será apresentado a seguir.



(a) Estado inicial fixo



(b) Estado inicial variável

Figura 4.5: Erro relativo variando o tamanho da trajetória da técnica *Forward Simulation* para a coleta de $1.000.000$ de amostras do modelo ASP com $k = 4$ e $P = 4$

4.4 Distribuição do Erro Relativo

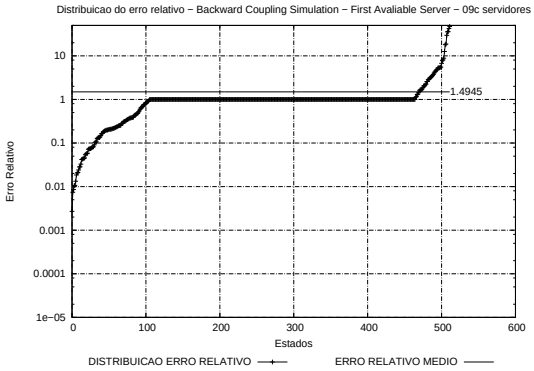
Os testes realizados nesta seção também demonstram a influência na quantidade de amostras coletadas na simulação, apresentado a distribuição do erro relativo com os estado do modelo simulado. Os gráficos da Figura 4.6, mostram essa distribuição para o modelo FAS com $N = 9$, aplicando a técnica *Backward Coupling Simulation* escolhido por apresentar erros relativos médios maiores que os demais modelos, conforme visto na Seção 4.1.

Cada gráfico da figura apresenta a distribuição do erro relativo com a coleta de uma quantidade diferente de amostras, sendo o eixo x os estados do modelo e o eixo y o erro relativo em escala logarítmica ordenados do menor para o maior para melhorar a visualização. Os resultados permitem observar uma característica bastante interessante aproximadamente entre os estados 100 e 470 no gráfico (a) e entre os estados 240 a 460 do gráfico (b) à respeito do erro que é exatamente igual a 1, ou seja, 100% enquanto que no gráfico (c), (d), (e) e (f), isso não acontece. Esse comportamento se explica, pois a pequena quantidade de amostras coletadas nos gráficos (a) e (b) de 10.000 e 100.000, respectivamente, não é suficiente para que amostras em todos os estados do modelo sejam coletadas. Este fato faz com que os mesmos sejam considerados, erroneamente, como inatingíveis de forma que a probabilidade contida no vetor resultante seja igual a $zero$ acarretando no erro de 100% caracterizado pela linha constante observada nos gráficos (a) e (b) da Figura 4.6. Esta análise permite concluir que 10.000 ou 100.000 amostras para este modelo é um número muito pequeno para atingir uma precisão próxima da solução analítica.

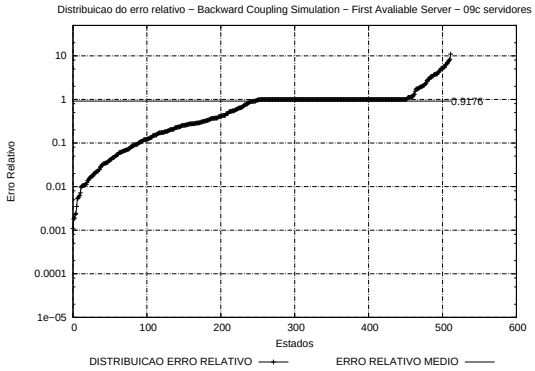
Um fator importante que merece ser analisado também é o tempo gasto para se conseguir um resultado preciso. Conforme observamos até o momento, o aumento do número de amostras coletadas reduz cada vez mais o erro da simulação, até que se atinja uma solução próxima da estacionária. Esta redução pode ser notada analisando o erro relativo médio do gráfico (a), representado pela linha contínua horizontal e analisando o erro relativo médio do gráfico (f), sendo ele reduzido na ordem de grandeza. Contudo, o custo computacional referente ao tempo de processamento pode ser bastante elevado e às vezes dependendo da necessidade do usuário, o mesmo pode não compensar, quando o erro apresenta pouca redução mesmo aumentando em ordens de grandeza a quantidade de amostras coletadas. Isso pode ser observado analisando o erro médio relativo do gráfico (g) com o do gráfico (f), onde o mesmo diminuiu $0,0494 - 0,0410 = 0,0084$, ou seja $0,84\%$. Este erro deve, entretanto, ser avaliado pelo usuário para que este julgue a relevância dessa redução ao tirar conclusões do sistema que está sendo modelado.

4.5 Tempos de Processamento

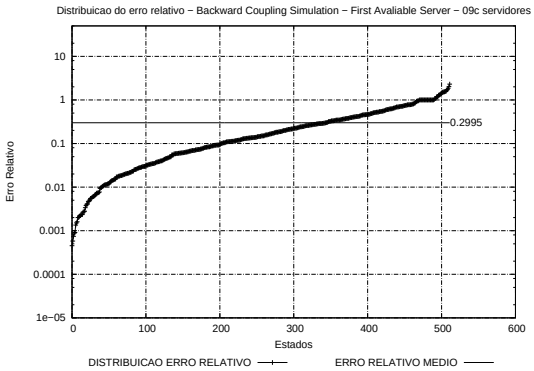
Esta seção apresenta os tempos de processamento gastos na execução das simulações para a obtenção dos resultados apresentados nos gráficos da Seção 4.1, referentes à precisão obtida com a coletada de diferentes quantidades de amostras para todas as técnicas de simulação apresentadas



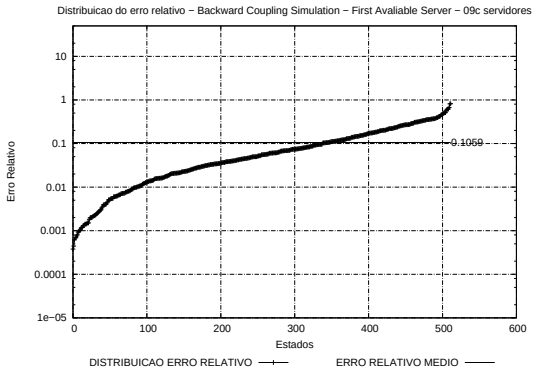
(a) 10.000 amostras



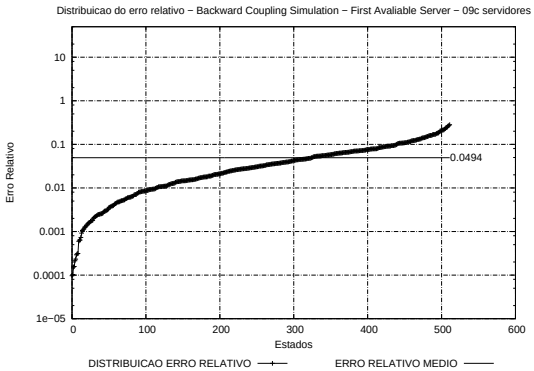
(b) 100.000 amostras



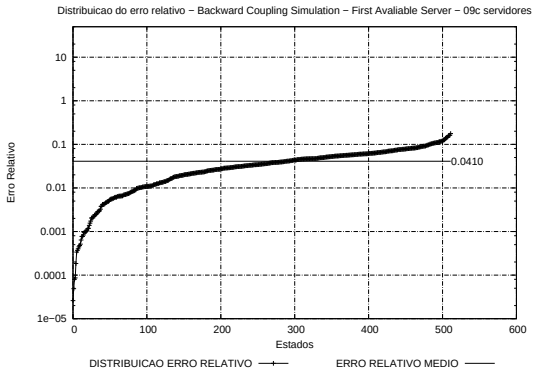
(c) 1.000.000 de amostras



(d) 10.000.000 de amostras



(e) 100.000.000 de amostras



(f) 1.000.000.000 de amostras

Figura 4.6: Distribuição do erro relativo do modelo FAS com 9 servidores com a técnica *Backward Coupling Simulation*

neste trabalho aplicadas aos modelos ASP, FAS e RS.

Antes de analisá-los é importante mencionar que foi realizado um estudo na paralelização das técnicas de simulação, utilizando a biblioteca *Message Passing Interface (MPI)* e o paradigma mestre-escravo [25], bastante comum em sistemas distribuídos. Neste paradigma, existe um nodo (processo) que divide as tarefas entre todos os nodos escravos e por fim recebe e processa os resultados retornados pelos mesmos, através de troca de mensagens. Na simulação, este nodo mestre divide as amostras a serem coletadas entre cada nodo escravo, sendo que cada um deles processa por um tempo proporcionalmente distribuído, juntamente com o nodo mestre. Por fim, cada nodo escravo envia um vetor contendo os resultados obtidos novamente ao nodo mestre, onde este, por sua vez, acumula o resultado em um único vetor e calcula as probabilidades do modelo. Esta característica permite com que a paralelização se torne bastante vantajosa pois o fato de haver pouca comunicação entre os nodos (escravos com o mestre e vice-versa) torna, consequentemente, o tempo de comunicação bastante pequeno, uma vez que são transmitidos na rede apenas os vetores com as amostras de estados por cada nodo escravo. Este fato também garante um *speedup* muito próximo do ideal, conforme será apresentado mais adiante.

Estes testes foram realizados para avaliar o comportamento paralelo em cada uma das técnicas propostas a fim de acelerar a coleta dos resultados utilizados nas análises deste trabalho. O gráfico (a) da Figura 4.7 mostra o *speedup*, definido pelo quociente da divisão entre o tempo sequencial e o tempo paralelo, de acordo com a equação 4.3, sendo p o número de processadores, T_1 o tempo de execução do algoritmo sequencial e T_p o tempo de execução do algoritmo paralelo com p processadores.

$$S_p = \frac{T_1}{T_p} \tag{4.3}$$

Assim sendo, o eixo x do gráfico (a) corresponde ao número de processadores e o eixo y corresponde ao *speedup* resultante. O gráfico (b) da figura, por sua vez, mostra o tempo de processamento de cada técnica, com os respectivos p processadores no eixo x. Vale destacar que estes tempos foram adquiridos com a coleta de $1.000.000$ de amostras com cada técnica e as simulações foram realizadas no cluster Cerrado da PUCRS. Este cluster [12] é composto por cinco nodos bi-processados (10 processadores), com tecnologia *Intel Itanium* com frequência de $1.5GHz$ e $2GB$ de memória.

Conforme pode ser observado nos gráficos, o *speedup* apresentou resultados ideais, resultando em um ótimo desempenho com a paralelização de todas as técnicas apresentadas. A redução do *speedup*, no entanto, para as técnicas *Permanence Time Simulation* conforme pode ser observado com a utilização de mais de 6 processadores, ocorre pois a mesma é muito rápida, para a coleta de apenas $1.000.000$ de amostras. Número este, que para esta técnica é de baixa carga computacional, visto que o tempo de processamento foi praticamente desprezível ($0,04$ a $0,13$ segundos), ainda mais quando comparados com as técnicas *Forward Simulation* e *Backward Coupling Simulation*, as quais apresentam um tempo de várias minutos. Isso pode ser constatado observando o tempo despendido com dois processadores, visto no gráfico (b) para a técnica *Forward Simulation* o qual foi de 1.020

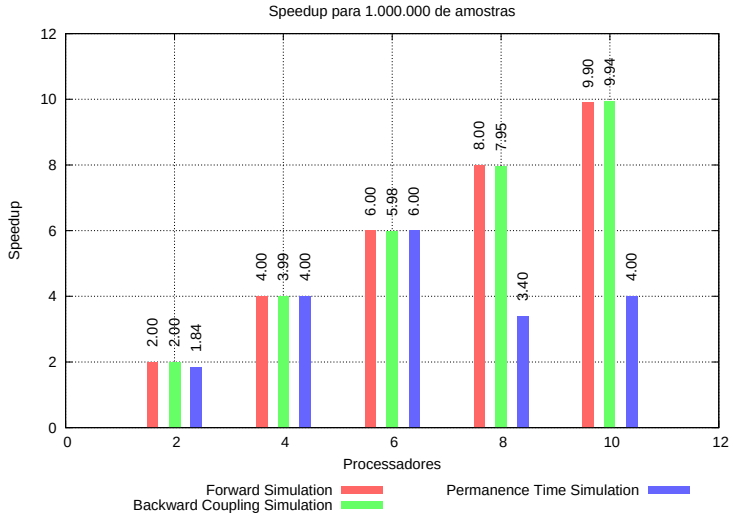
segundos, ou seja, aproximadamente 17 minutos de processamento.

Os gráficos seguintes, referentes às Figuras 4.8, 4.9 e 4.10 apresentam um comparativo entre as três técnicas de simulação em relação ao tempo de processamento, utilizando 10 processadores durante a execução dos modelos ASP, FAS e RS respectivamente.

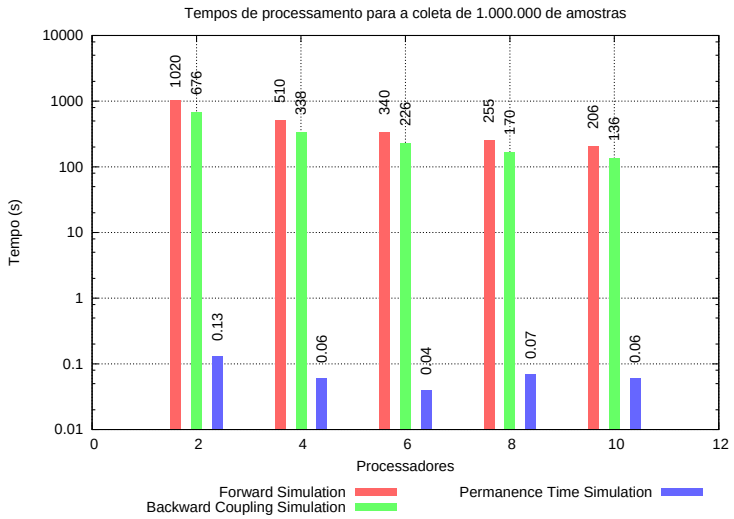
Um fator importante observado nos três modelos, que merece ser destacado, é que a técnica *Permanence Time Simulation*, por realizar apenas uma trajetória e coletar como amostra o número de transições para cada estado, apresenta-se como a mais eficiente em tempo de processamento, uma vez que nenhuma execução demorou mais do que 25 segundos. Em contrapartida a técnica *Forward Simulation* mostrou-se a mais custosa, o que é justificável pelo grande tamanho das trajetórias que são realizadas no modelo (10.000 transições). O tempo também é bastante grande na técnica *Backward Coupling Simulation*, comparado com a *Permanence Time Simulation*. Ao observarmos, por exemplo, o tempo gasto na coleta de $1.000.000.000$ ($1e + 09$) de amostras para o modelo ASP do gráfico da Figura 4.8, obteve-se para a técnica *Forward* 165.200 segundos (aproximadamente 46 horas) e para a técnica backward 116.912 segundos (cerca de $32,5$ horas) enquanto que para a técnica *Permanence Time* este tempo foi cerca de 21 segundos.

O comportamento dos gráficos permite prever também o tempo necessário para a coleta de amostras com uma ordem de grandeza maior para qualquer modelo apresentado. Isso é fácil de ser inferido, pois o mesmo também aumenta em uma ordem de grandeza, que no caso do modelo ASP aplicando à técnica *Backward* seria de aproximadamente $1.169.120$ segundos ($324,75$ horas ou $13,53$ dias) para coletar $10.000.000.000$ de amostras.

O tempo de processamento das técnicas mencionadas ainda é uma área que pode ser amplamente explorada uma vez que, além da paralelização, outros métodos podem ser acrescentados a cada uma delas para agilizar a coleta das amostras. Dentre eles pode ser mencionado o método *Aliasing* [40], caracterizado por definir o próximo estado de forma mais eficiente.

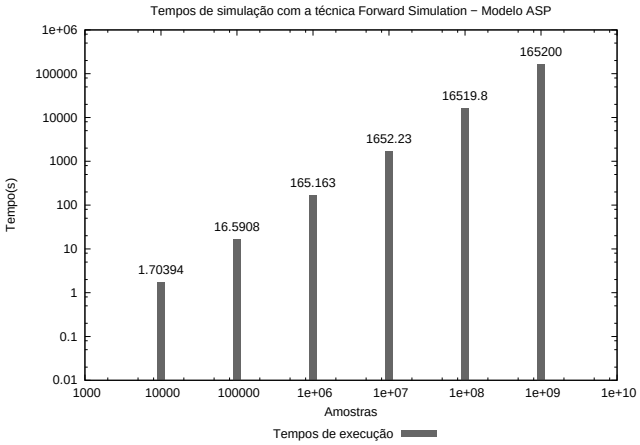


(a) Speedup

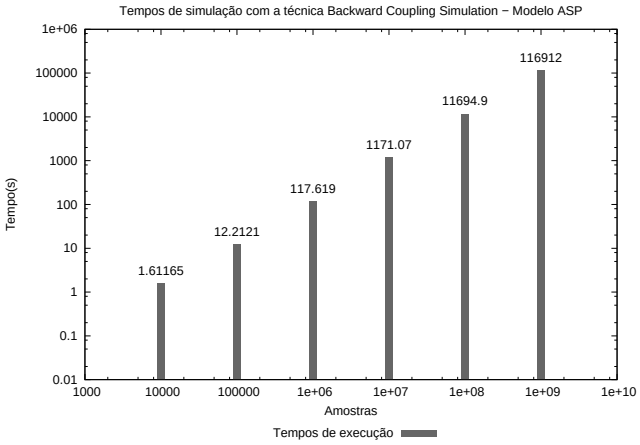


(b) Tempos Paralelos

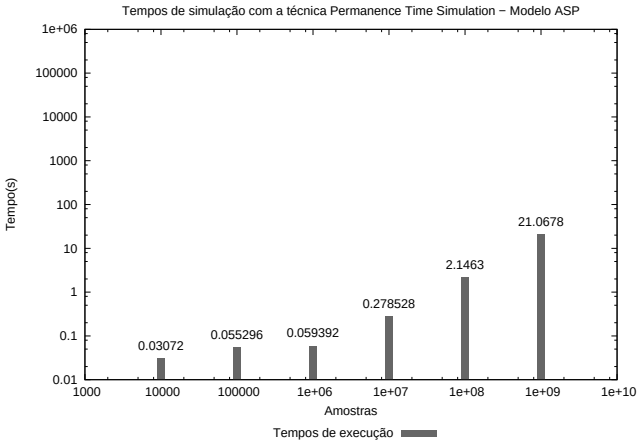
Figura 4.7: *Speedup* e tempos de processamento para o modelo RS com 10 recursos e 5 processos na coleta de 1.000.000 de amostras



(a) Forward

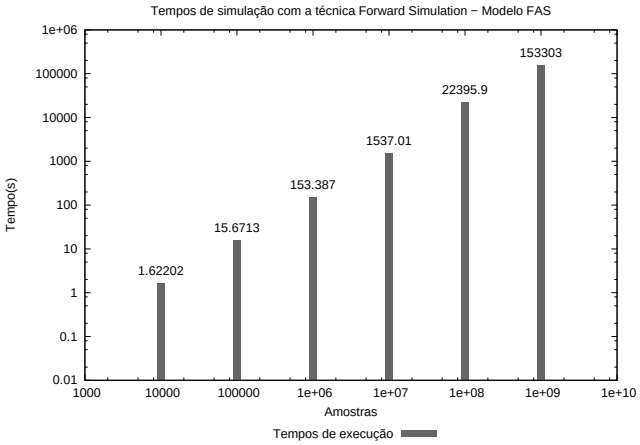


(b) Backward

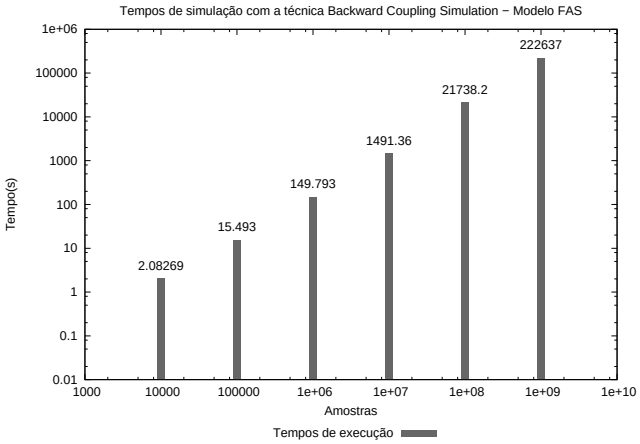


(c) Permanence Time

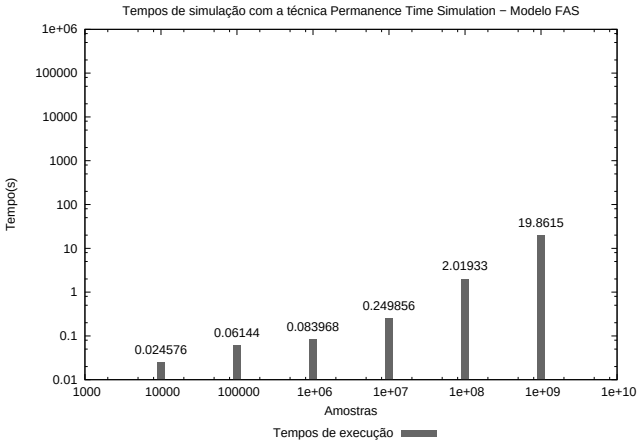
Figura 4.8: Tempos de execução das técnicas de simulação para o modelo ASP com $K = 2$ e $P = 2$



(a) Forward

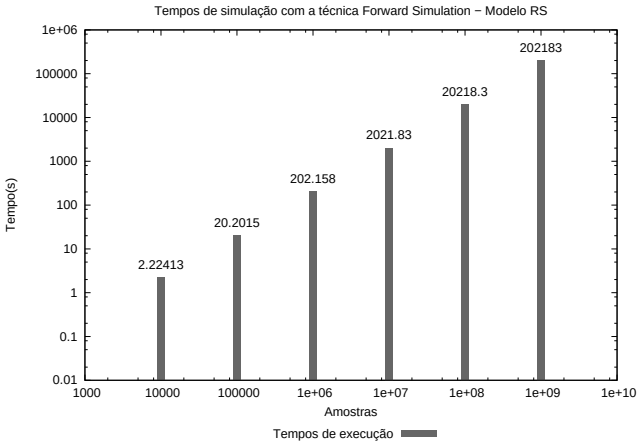


(b) Backward

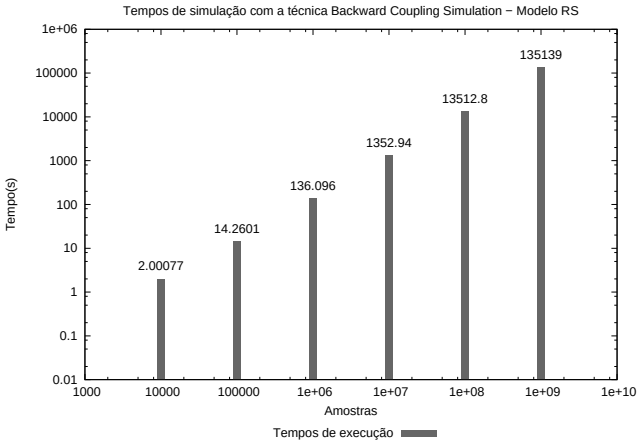


(c) Permanence Time

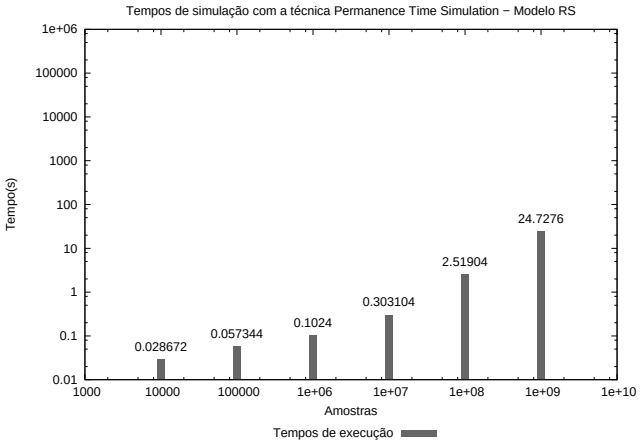
Figura 4.9: Tempos de execução das técnicas de simulação para o modelo FAS com 9 servidores



(a) Forward



(b) Backward



(c) Permanence Time

Figura 4.10: Tempos de execução das técnicas de simulação para o modelo RS com 10 recursos e 5 processos

Capítulo 5

Técnica Proposta

Conforme observado anteriormente, mesmo aumentando em ordens de grandeza o número de amostras coletadas durante o processo de simulação, obtêm-se ganhos pouco significativo em termos de precisão, após determinada quantidade das mesmas. Em virtude disso e na busca de resultados mais precisos, foi realizado um estudo no *Bootstrap Method*, proposto por Efron [18]. Este método consiste em uma técnica utilizada para resolver uma grande variedade de problemas de estimativa em diversas áreas, sendo aplicada no campo de estatística para derivar estimativas de erro padrão e intervalo de confiança para estimadores de parâmetros complexos da distribuição. A essência dessa técnica é que na ausência de qualquer conhecimento prévio sobre uma determinada população $\tilde{\Lambda}$ de tamanho infinito, a distribuição dos valores encontrados em uma amostra aleatória Λ de tamanho n (extraída dessa população, $i.e.$, $\Lambda \subset \tilde{\Lambda}$) é a melhor abordagem para obter a distribuição da mesma [28]. Em outras palavras, a população infinita observada em apenas n valores da amostra Λ, cada um com probabilidade $\frac{1}{n}$, é usada para modelar a população desconhecida real.

O conjunto de valores de uma nova amostra K, também de tamanho n ($|K| = |\Lambda|$), é obtido exclusivamente a partir da primeira amostra Λ, sem que hajam novos valores da população $\tilde{\Lambda}$, conforme pode ser observado na Figura 5.1. A principal característica da técnica *Bootstrap* é a amostragem com reposição, $i.e.$, os valores que compõem a amostra podem repetir-se durante tiragens de valores.

Para exemplificar o uso dessa técnica considere que se deseja encontrar a média de altura da população mundial. Como seria inviável obter esses valores para toda a população (conjunto $\tilde{\Lambda}$), apenas uma amostra desta população é considerada (subconjunto Λ). De posse de Λ são realizadas z reamostragens, que correspondem então ao número de *bootstraps*, conforme visto na Figura 5.1, onde $K_i = \{x \in K_i | x \in \Lambda\}$ e $i \in [1..z]$. Cada *bootstrap* K_i conterá um conjunto de n valores obtidos de Λ, através de tiragens pseudo-aleatórias podendo haver repetições de valores ($i.e.$, $x_1 \in K_i, x_2 \in K_i | x_1 = x_2$). A média $\bar{x}_K$ da altura da população é calculada pelas médias $\bar{x}_{K_1}$, $\bar{x}_{K_2}$, $\ldots$, $\bar{x}_{K_z}$ de cada *bootstrap*. A técnica *Bootstrap* objetiva uma melhor precisão para a média $\bar{x}_K$ do que a média $\bar{x}_\Lambda$. Além disso, o aumento do número de *bootstraps* pode reduzir os efeitos causados por erros (ruídos) dos geradores de números pseudo-aleatórios que os compõem.

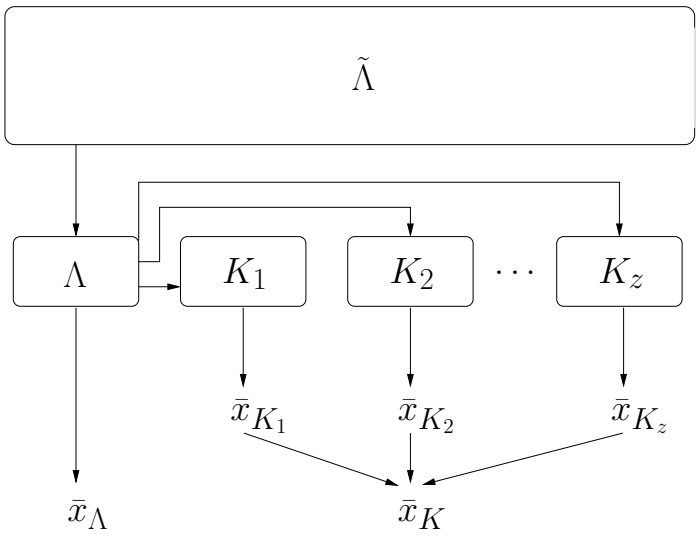


Figura 5.1: Técnica *Bootstrap*

5.1 *Bootstrap Simulation*

A proposta de utilização desta técnica na simulação de modelos Markovianos foi adaptada de forma que *bootstraps* fossem integrados à função de transição ϕ. Para isso, gera-se um valor real, uniformemente distribuído entre 0 e 1, e a função de transição $\phi(s_i, U)$ determina o próximo estado do modelo a ser visitado. A sequência de estados visitados compõe a trajetória da simulação, a qual corresponde à amostra Λ definida anteriormente. A Figura 5.2 ilustra essa trajetória, sendo que a cada estado visitado são realizadas tiragens de valores pseudo-aleatórios que irão definir os estados que serão coletados em cada *bootstrap* K_i. O número de tiragens é igual a quantidade de valores da amostra Λ, ou seja, n valores, que por sua vez é também igual ao tamanho da trajetória da simulação. As tiragens são realizadas de acordo com a geração de valores inteiros uniformemente distribuídos entre 0 e $n-1$ através do gerador U para cada *bootstrap*, ou seja, z vezes. Na Figura 5.2, T_i (onde $i \in [1..z]$) representa as n tiragens para cada *bootstrap* K_i. Logo, as $n \times z$ tiragens ocorrem a cada passo/transição da trajetória (no modelo) e os valores pseudo-aleatórios, retornado pelo gerador U, são comparados com uma constante α qualquer escolhida arbitrariamente também entre 0 e $n-1$. Se o valor retornado pelo gerador for igual a esta constante α, o estado corrente na trajetória é coletado no *bootstrap* correspondente (K_b). Este procedimento se repete para todos os *bootstraps* até que o tamanho (n) pré-estabelecido da trajetória seja atingido. Na Figura 5.2, $|\underline{\bar{b}}|$ representa este procedimento a cada transição realizada na simulação. Ao final da simulação ($i.e.$, quando toda a trajetória for percorrida), o vetor de probabilidades π, o qual corresponde à solução do modelo com suas respectivas probabilidades de permanência em cada estado, é calculado através das médias das probabilidades de permanência encontrada para os estados em cada *bootstrap*.

O Algoritmo 5.1 apresenta a aplicação da técnica *Bootstrap* empregada no contexto de simulação. Um fato importante a ser notado na utilização desta técnica é que realizar tiragens por um número

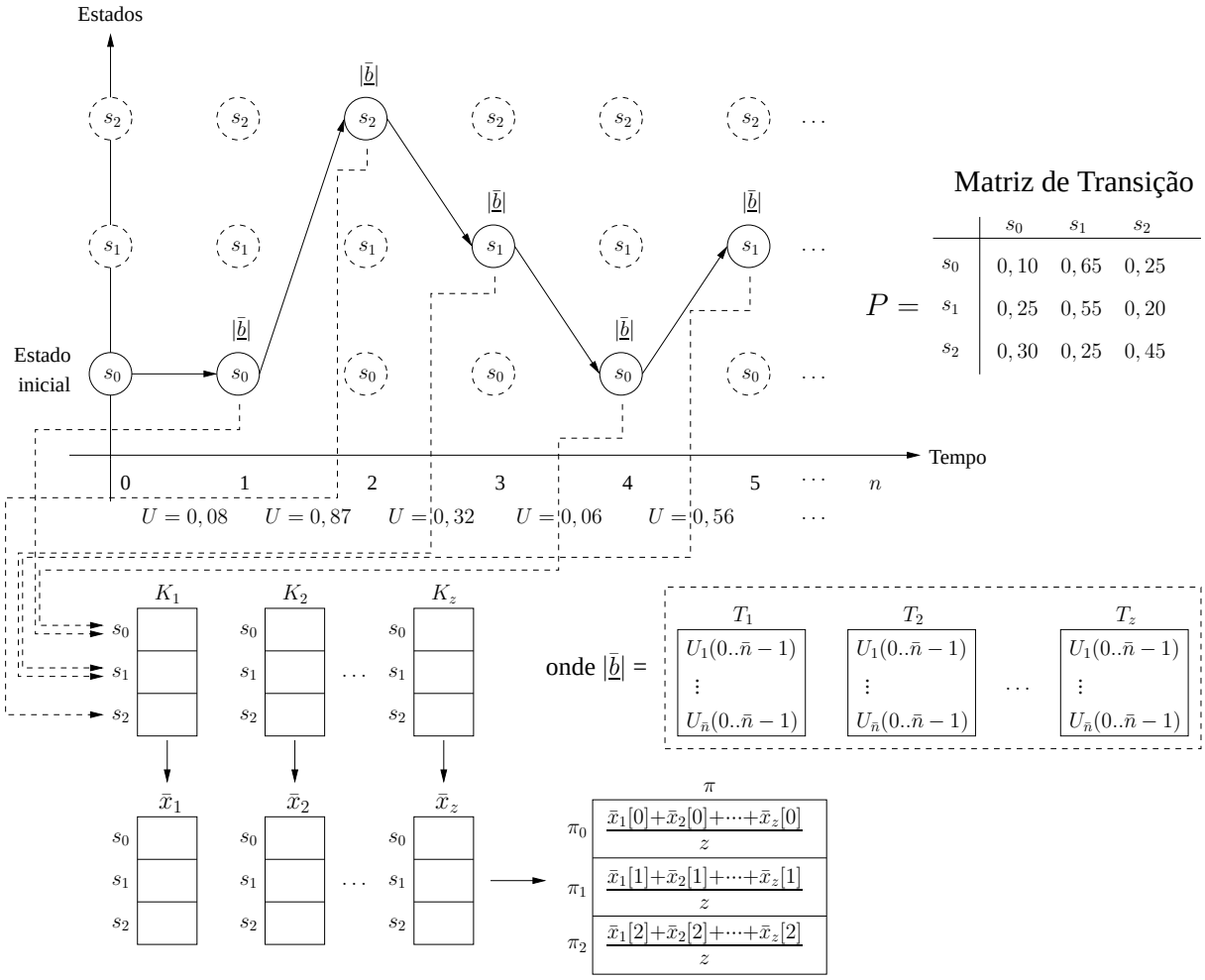


Figura 5.2: Simulação *Bootstrap*

de vezes igual ao tamanho n da trajetória pode tornar-se impraticável computacionalmente. Devido a este fato, ao invés de realizar n tiragens para cada *bootstrap*, serão realizadas $\bar{n}$ tiragens, onde $\bar{n} \ll n$. Para avaliar a influência na redução de n para $\bar{n}$ tiragens, foi calculada a probabilidade de um mesmo valor ser coletado $\bar{n}+1$ vezes durante n tiragens. Se essa probabilidade for significativa, significa que o número de tiragens $\bar{n}$ não é suficiente, visto que uma chance alta de um mesmo valor repetir-se mais de $\bar{n}$ vezes, por exemplo, irá afetar a precisão dos resultados. Este cálculo é realizado baseado no problema clássico conhecido como *The Birthday Problem* [1] - Equação (5.1) - o qual consiste em descobrir as chances de em um conjunto de pessoas, escolhidas aleatoriamente, existirem pares que façam aniversário na mesma data.

$$\text{Probabilidade} = 1 - \left[\frac{n!}{(n-\bar{n}+1)!} \times \left(\frac{1}{n}\right)^{\bar{n}+1} \right] \tag{5.1}$$

A fim de determinar um valor satisfatório para $\bar{n}$, definiu-se $\bar{n} = 20$ para o cálculo das probabilidades dada pela equação (5.1). Neste caso, quando $n \geq 10^4$, a probabilidade de um mesmo valor ser obtido $20{+}1$ vezes com n tiragens é inferior a 2%, conforme pode ser observado na Tabela 5.1. É

importante mencionar também que as tiragens de valores, que eram realizadas uniformemente entre 0 e $n-1$, são agora realizadas entre 0 e $\bar{n}-1$ para manter a probabilidade de $\frac{1}{\bar{n}}$.

Tabela 5.1: Probabilidade de 20 ou mais tiragens obterem o mesmo valor

Tamanho da amostra n	Probabilidade (Equação 5.1)
10.000	0.02079510 ($2\times10^{0}\%$)
100.000	0.00209793 ($2\times10^{-1}\%$)
1.000.000	0.00020997 ($2\times10^{-2}\%$)
10.000.000	0.00002099 ($2\times10^{-3}\%$)
100.000.000	0.00000209 ($2\times10^{-4}\%$)
1.000.000.000	0.00000021 ($2\times10^{-5}\%$)

Algorithm 5.1: Representação do algoritmo para a técnica *Bootstrap Simulation*

1: $\alpha \leftarrow U(0..\bar{n}-1)$ {*inicialização da constante* α *com um valor pseudo-aleatório entre* 0 *e* $n-1$}
2: $\pi \leftarrow 0$ {*inicializa todas as posições do vetor de probabilidades* π}
3: $K \leftarrow 0$ {*inicializa todos os* z *bootstraps* K}
4: $s_c \leftarrow s_0$ {*escolha arbitrária de um estado corrente* s_c *inicial*}
5: { *percorre a trajetória de tamanho* n }
6: **for** $t = 1$ to n **do**
7: $\quad s_d \leftarrow \phi(s_c, U(0..1))$ {*determina o estado destino* s_d *a partir de* s_c *dado o resultado de* $U(0..1)$}
8: $\quad$ **for** $b = 1$ to z **do**
9: $\quad\quad$ **for** $c = 1$ to $\bar{n}$ **do**
10: $\quad\quad\quad$ **if** $(U(0..\bar{n}-1) == \alpha)$ **then**
11: $\quad\quad\quad\quad K_b[s_d] \leftarrow K_b[s_d] + 1$ {*incrementa* $K_b[s_d]$ *toda vez que o valor da tiragem for igual a* α}
12: $\quad s_c \leftarrow s_d$ {*atualiza o estado corrente* s_c}
13: **for** $b = 1$ to z **do**
14: $\quad \omega \leftarrow 0$
15: $\quad$ **for** $i = 1$ to $|RSS|$ **do**
16: $\quad\quad \omega \leftarrow \omega + K_b[i]$ {*calcula em* ω *a soma total dos valores acumulados no bootstrap* K_b}
17: $\quad$ **for** $i = 1$ to $|RSS|$ **do**
18: $\quad\quad \bar{x}_b[i] \leftarrow \frac{K_b[i]}{\omega}$ {*calcula em* $\bar{x}_b$ *as probabilidades do estado* i *para o bootstrap* K_b}
19: **for** $i = 1$ to $|RSS|$ **do**
20: $\quad$ **for** $b = 1$ to z **do**
21: $\quad\quad \pi[i] \leftarrow \pi[i] + \bar{x}_b[i]$
22: $\quad \pi[i] \leftarrow \frac{\pi[i]}{z}$ {*calcula em* π *a média dos bootstraps*}

No Algoritmo 5.1, note que entre as linhas 1 e 4 são realizadas as inicializações das variáveis e vetores utilizados no emprego da técnica no contexto de simulação. As tiragens armazenadas nos *bootstraps* percorrendo toda a trajetória de tamanho n ocorrem da linha 6 até a linha 12. Entre as linhas 13 e 18, são calculadas as probabilidades dos estados do modelo em cada *bootstrap*. E, finalmente, entre as linhas 19 e 22, são calculadas as médias das probabilidades de cada estado no vetor π a partir dos *bootstraps*.

5.2 Resultados Obtidos

Após a adaptação do método *bootstrap* como uma nova técnica de simulação, foram repetidos alguns dos experimentos apresentados no Capítulo 4 para avaliar o comportamento da mesma em relação à precisão e ao tempo de processamento. Os resultados são apresentados a seguir.

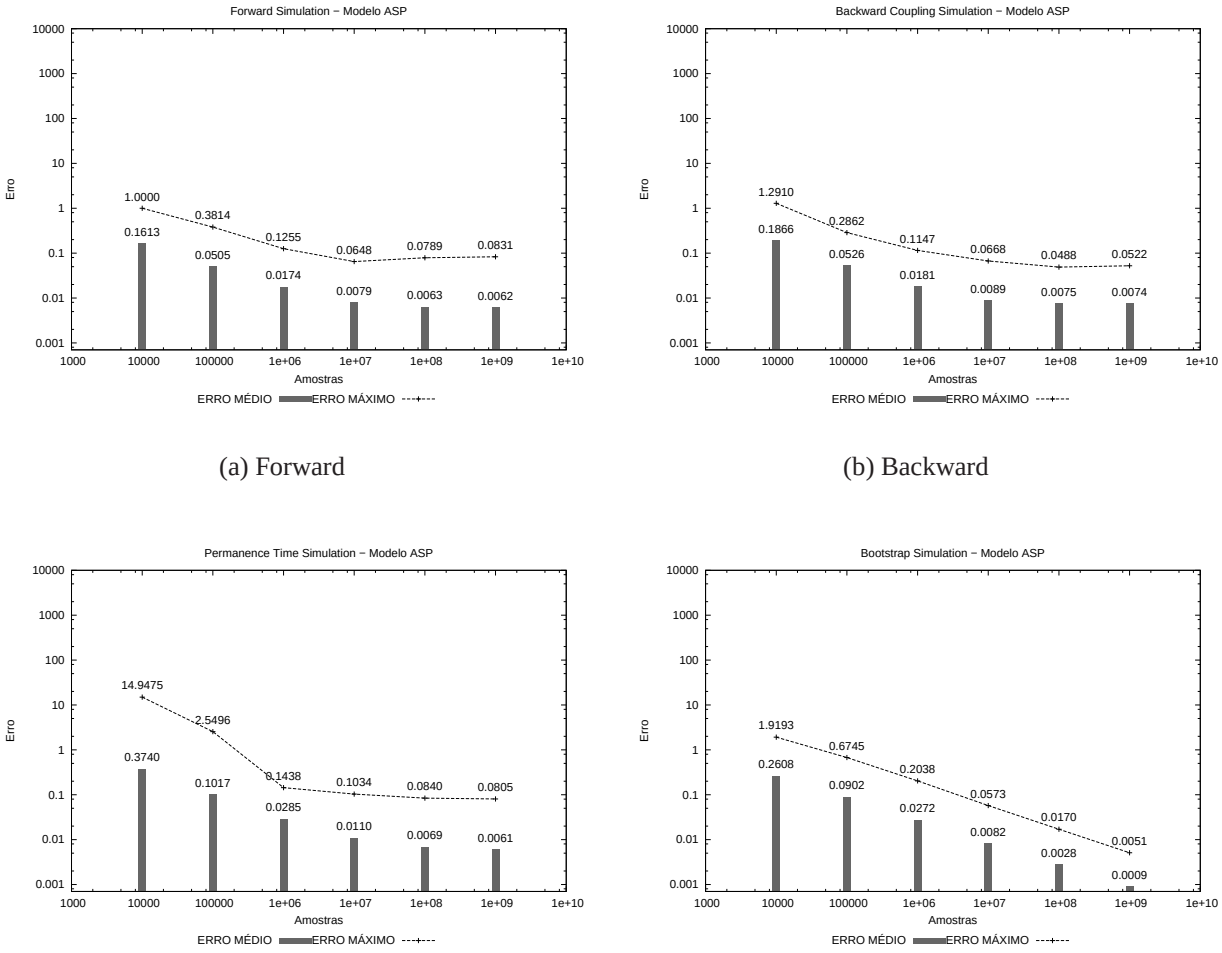
5.2.1 Erro Relativo da Técnica *Bootstrap Simulation*

Para verificar o comportamento da técnica *Bootstrap Simulation* aplicada à resolução de modelos Markovianos, a mesma foi utilizada na simulação dos modelos ASP, FAS e RS. Uma questão ainda em aberto para a utilização desta técnica é definir um número adequado de *bootstraps*, no entanto para os gráficos apresentados nesta seção, este número foi fixado em 10 de forma arbitrária.

Os resultados com a utilização desta nova técnica podem ser analisados observando as Figuras 5.3, 5.4 e 5.5 a seguir, onde foram repetidos os gráficos das técnicas anteriores para facilitar a análise e as comparações com a nova técnica proposta. Observando os gráficos da primeira figura, referente ao modelo ASP, nota-se que os resultados demostraram maior precisão com a redução do erro médio a partir da coleta de $10.000.000$ ($1e+07$) de amostras. Paralelo a ela as demais técnicas apresentaram reduções muito pequenas no erro relativo médio e máximo, apresentando certa estacionariedade, enquanto que a técnica proposta continua obtendo ganhos significativos. Se observarmos o último ponto do gráfico (c), referente a técnica *permanence time* e o gráfico (d), referente a técnica *bootstrap*, o erro que era de $0,61\%$ cai para $0,09\%$, sendo que a última apresenta uma linha decrescente constante em relação a redução tanto do erro relativo médio como do erro relativo máximo, ao contrário das demais.

Comparando os resultados obtidos para o modelo FAS (Figura 5.4), obteve-se erros relativos menores com a técnica *Bootstrap Simulation*, principalmente para os dois últimos pontos do gráfico (d) ($1e+08$ e $1e+09$ amostras). Já para o primeiro ponto (10.000) os resultados apresentaram um erro máximo maior, no entanto conforme visto na Seção 4.2, este valor pode apresentar variações significativas, devido a influência na semente utilizada no gerador de números aleatórios, uma vez que a quantidade de amostras é bastante pequena.

Com relação ao modelo RS, a técnica *Bootstrap Simulation* apresentou resultados bastante parecidos com as demais, no entanto, mesmo não obtendo ganhos a mesma demonstrou equivalência em relação ao erro obtido entre todas elas. Desta forma pode-se dizer que a técnica *Bootstrap Simulation* apresenta em geral boa precisão, uma vez que os resultados obtidos com o maior número de amostras utilizadas foram mais precisos que as demais técnicas, sendo que no pior caso a mesma obteve precisão praticamente equivalente, como no caso do modelo RS. Outro fator que merece destaque ao utilizá-la é em relação a escolha do número de *bootstraps* e ao tempo de processamento, conforme será apresentado a seguir.



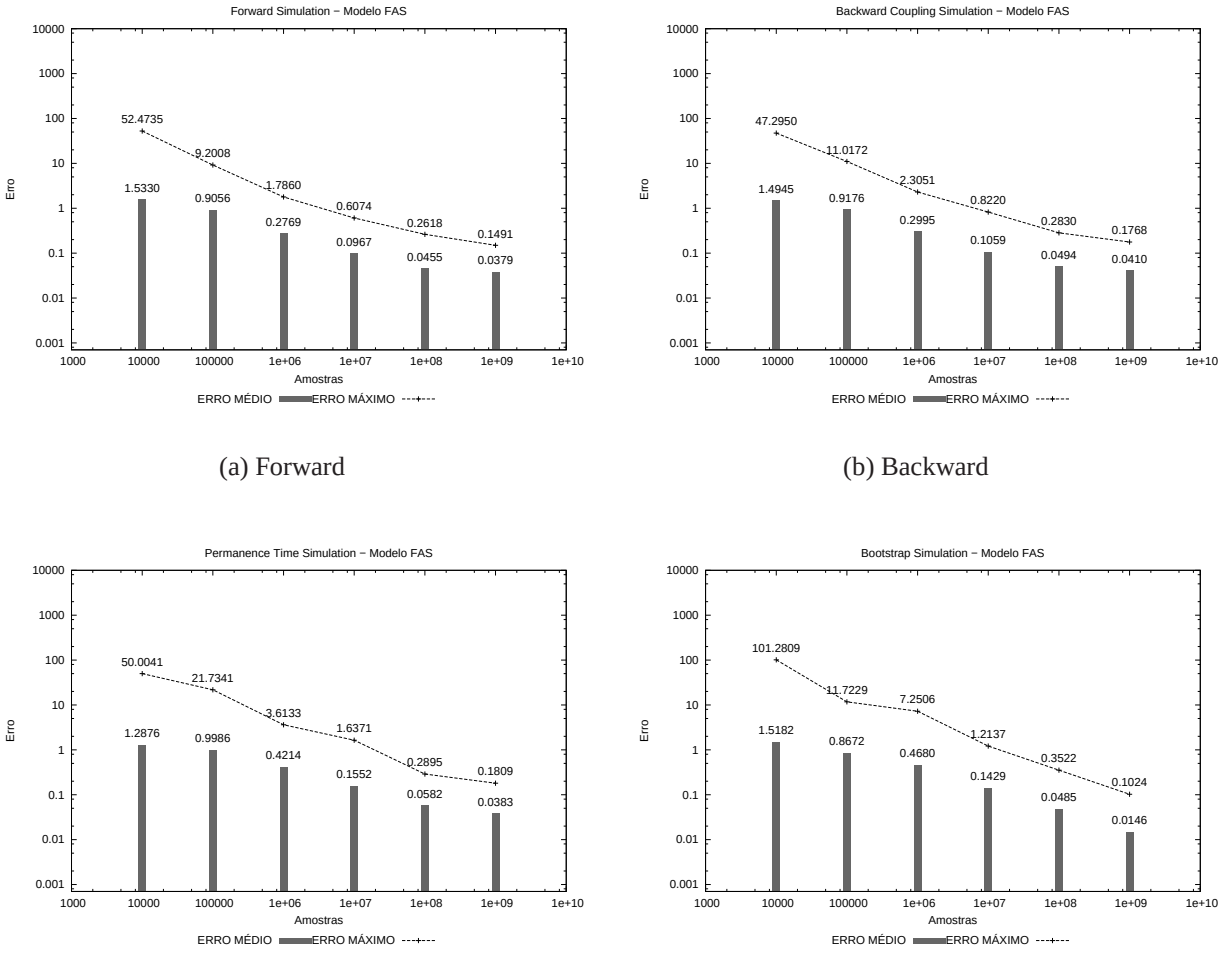
(a) Forward

(b) Backward

(c) Permanence Time

(d) Bootstrap

Figura 5.3: Erros relativos médios e máximos entre os resultados da aplicação das técnicas de simulação e o resultado da solução analítica para o modelo ASP com $K = 2$ e $P = 2$



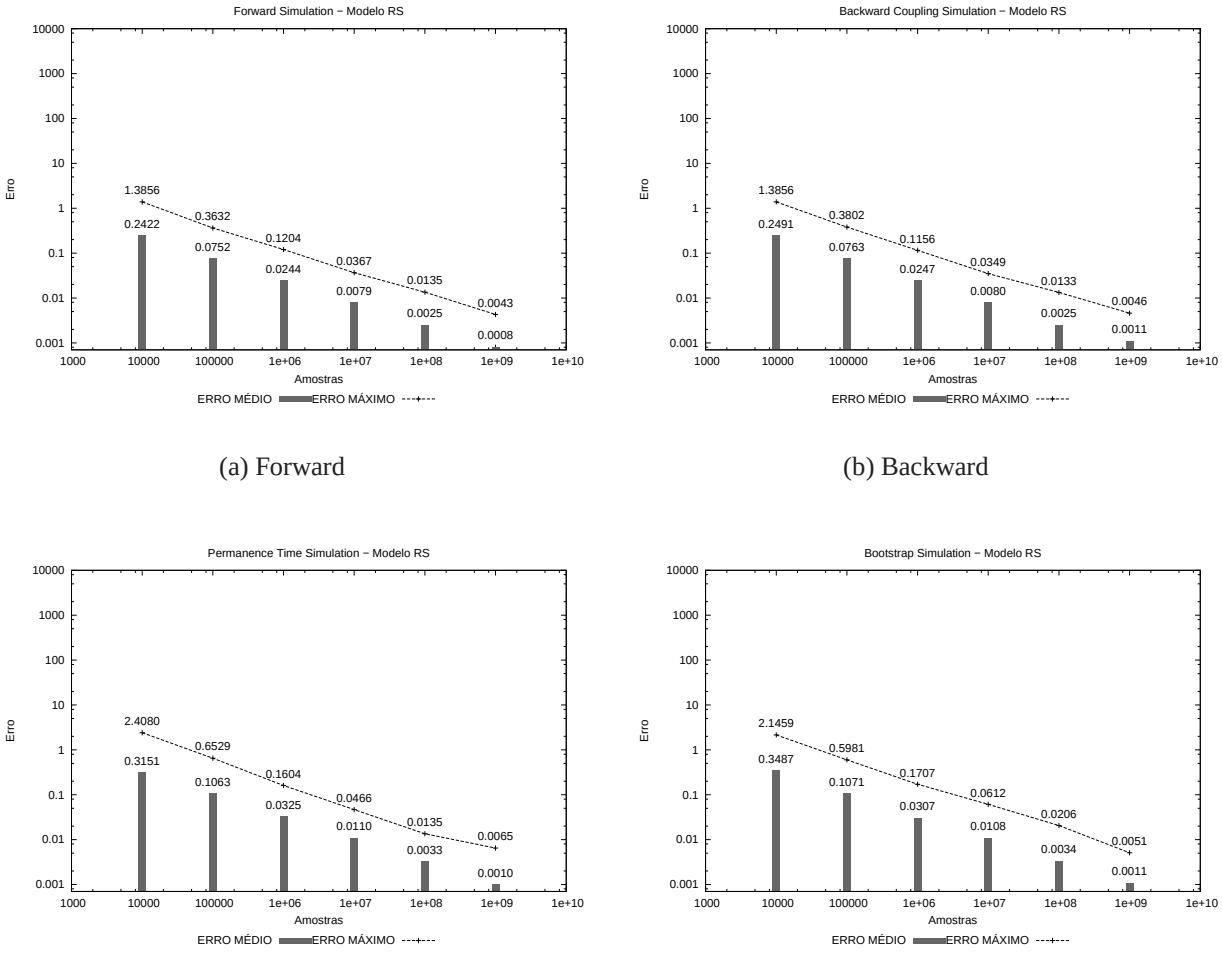
(a) Forward

(b) Backward

(c) Permanence Time

(d) Bootstrap

Figura 5.4: Erros relativos médios e máximos entre os resultados da aplicação das técnicas de simulação e o resultado da solução analítica para o modelo FAS com 9 servidores



(a) Forward

(b) Backward

(c) Permanence Time

(d) Bootstrap

Figura 5.5: Erros relativos médios e máximos entre os resultados da aplicação das técnicas de simulação e o resultado da solução analítica para o modelo RS com 10 processos e 5 recursos

5.2.2 Variação do Número de *Bootstraps*

A fim de avaliar o impacto no número de *bootstraps*, utilizados na aplicação da técnica, foram realizados testes variando este número entre 4, 10 e 50. Os resultados obtidos são apresentados nas Figuras 5.6, 5.7 e 5.8, para os modelos ASP, FAS e RS respectivamente. Observando por exemplo, o primeiro ponto (10.000) dos gráficos do modelo FAS pode-se notar uma variação maior entre os gráficos (a), (b) e (c) deste modelo, porém, isso ocorre apenas para os primeiros pontos, sendo que conforme o tamanho das amostras aumenta, este erro fica bastante próximo em todos eles. Esta discrepância ocorre devido a um maior intervalo de confiança neste ponto, o que faz com que a influência da semente do gerador de números aleatórios seja maior para amostras menores, conforme avaliado na Seção 4.2.

Se observarmos os resultados dos demais modelos, não nota-se nenhuma diferença expressiva, uma vez que há pouca variação entre os mesmos. Acredita-se que caso o gerador de números aleatórios não esteja bem calibrado a variação dos *bootstraps* seja mais significativa.

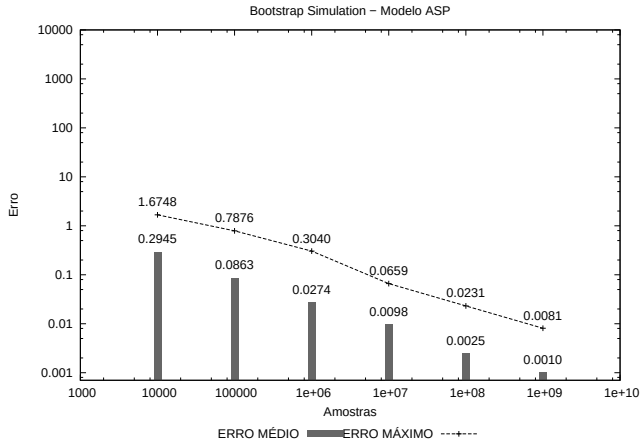
Vale destacar que quanto maior a quantidade de *bootstraps* utilizado na simulação desta técnica, maior é o tempo de processamento da mesma. Neste sentido vale a pena avaliar a influência deste valor uma vez que um número muito de grande de *bootstraps* pode acarretar apenas em desperdício de tempo, sem trazer ganhos nos resultados. Estes tempos serão abordados com mais detalhes na próxima seção.

5.2.3 Tempos de Processamento

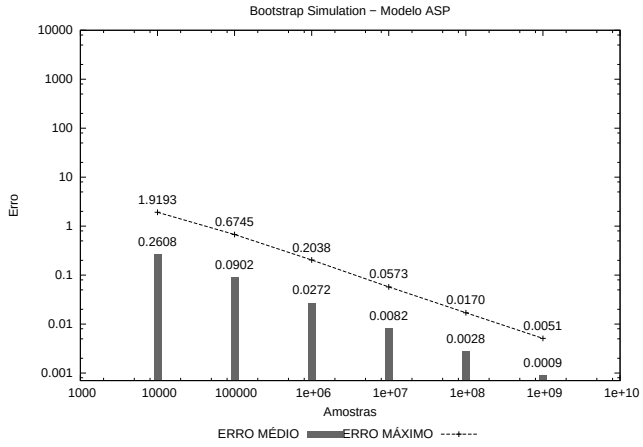
Para analisar os tempos de processamento com diferentes *bootstraps*, conforme visto anteriormente, são apresentados os gráficos da Figura 5.9, sendo o eixo x o tamanho das amostras e o eixo y o tempo, em segundos. Note que o tempo de processamento com 50 *bootstraps* é significativamente maior, aumentando em uma ordem de grandeza em relação a 4 *bootstraps*. Este resultado reforça a importância dessa escolha uma vez que um número excedente de *bootstraps* pode acarretar em um custo computacional muitas vezes maior.

É importante mencionar que os testes realizados com a técnica *Bootstrap Simulation* foram processados de forma sequencial, ou seja, com apenas um processador ao contrário das demais que foram paralelizadas e executadas com 10 processadores. Mesmo assim, acredita-se que a paralelização da técnica *bootstrap* também obtenha ganhos ideais de *speedup*, uma vez que a mesma pode ser paralelizada dividindo o tamanho da amostra n ou até mesmo o número de tiragens realizadas, entre os processadores. De posse então do tempo sequencial da mesma, foi inferido o tempo paralelo dividindo-o por 10, que corresponde ao número de processadores utilizados com as demais técnicas. Isso foi feito para que a técnica *Bootstrap* pudesse ser comparada com as demais, sendo os mesmos apresentados na Tabela 5.2, com o tempo sequencial real da técnica *Bootstrap* entre parênteses. Um estudo aprofundado na paralelização da mesma será proposto em um trabalho futuro.

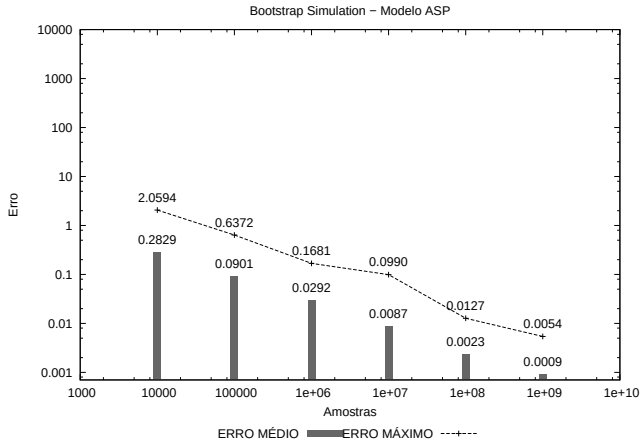
Note que os tempos de processamento para a técnica *Bootstrap Simulation* são significativamente



(a) ASP 4 bootstraps

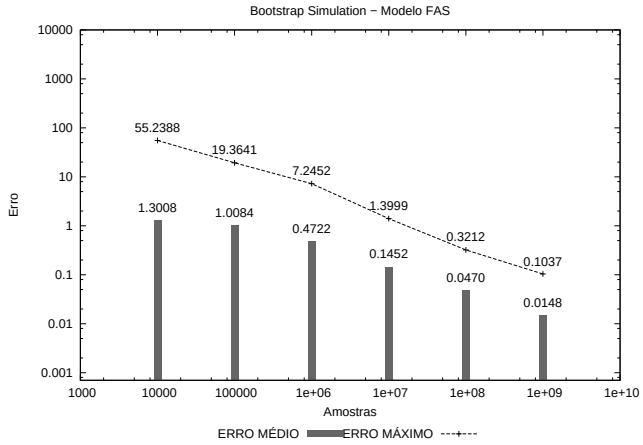


(b) ASP 10 bootstraps

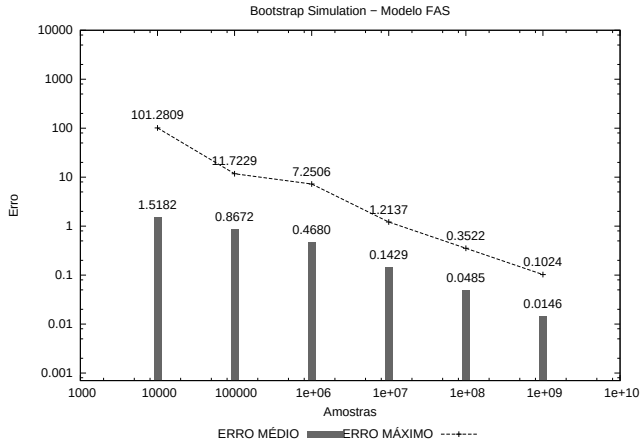


(c) ASP 50 bootstraps

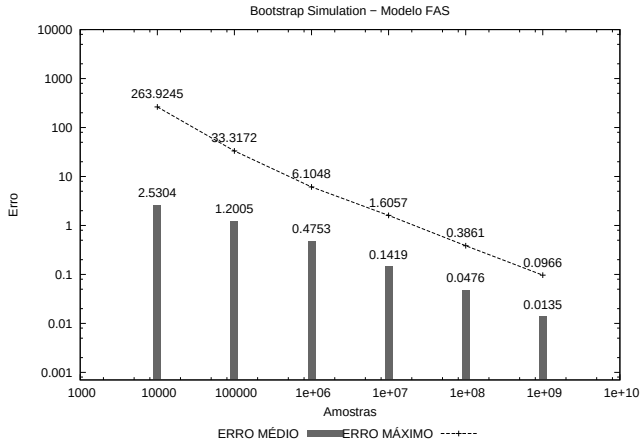
Figura 5.6: Erro Relativo variando o número de *bootstrap* para o modelo ASP com $K = 2$ e $P = 2$



(a) FAS 4 bootstraps

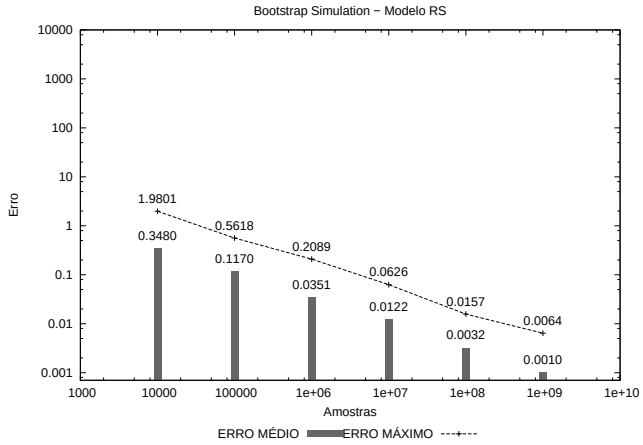


(b) FAS 10 bootstraps

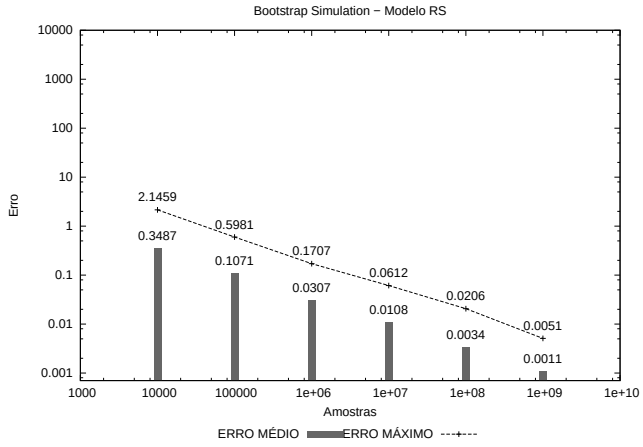


(c) FAS 50 bootstraps

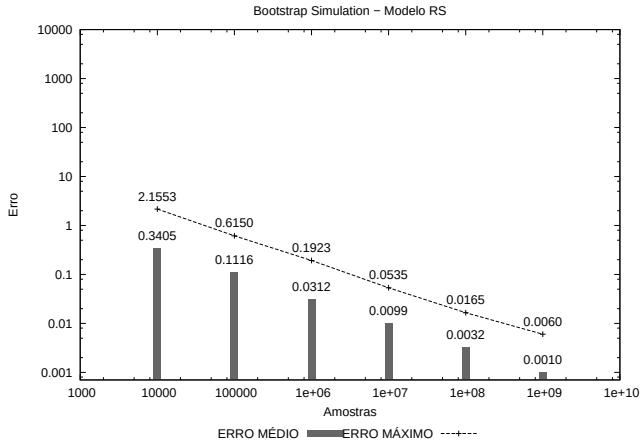
Figura 5.7: Erro Relativo variando o número de *bootstrap* para o modelo FAS com 9 servidores



(a) RS 4 bootstraps

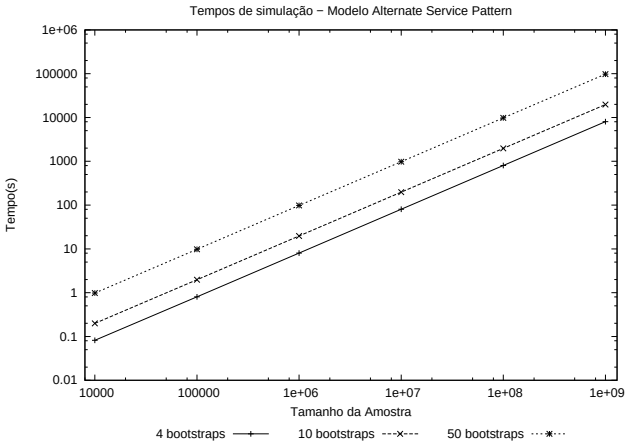


(b) RS 10 bootstraps

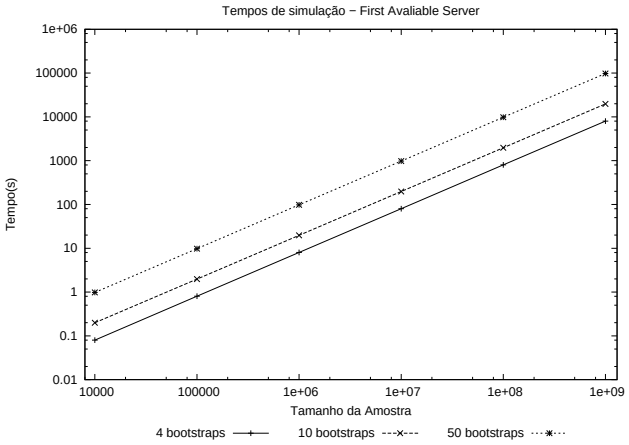


(c) RS 50 bootstraps

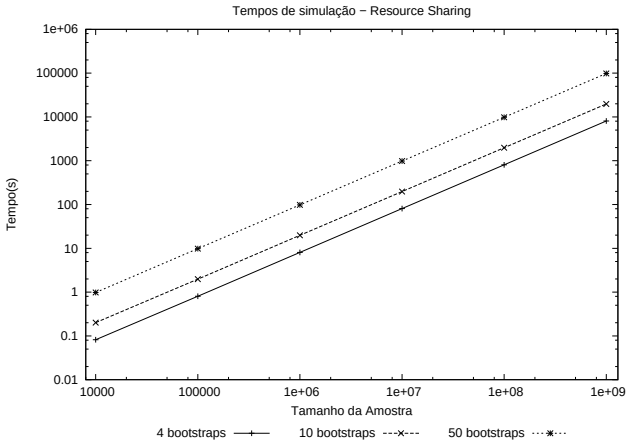
Figura 5.8: Erro Relativo variando o número de *bootstrap* para o modelo RS com 10 processos e 5 recursos



(a) Modelo ASP



(b) Modelo FAS



(c) Modelo RS

Figura 5.9: Tempos de execução da técnica *Bootstrap Simulation* variando o número de *bootstraps*

menores do que com as técnicas *Backward Coupling Simulation* e *Forward Simulation*, porém maiores para que técnica *Permanence Time Simulation*. É importante salientar que o tempo de processamento da técnica proposta foi coletado com a utilização de 10 *bootstraps*, visto que o mesmo varia conforme este número muda. Mesmo assim esta última apresenta bom desempenho em relação ao tempo de processamento, sendo poucas vezes maior que a técnica *Permanence Time* e muitas vezes menor que as demais, uma vez que o próprio tempo sequencial desta técnica é menor que o tempo paralelo das técnicas *Backward Simulation* e *Forward Simulation*.

Tabela 5.2: Tempos parelelos (em segundos) com a aplicação das técnicas de simulação

	ASP			
Número de Amostras	Forw.	Backw.	Perm. T.	Bootstrap
1e+04	1,70394	1,61165	0,0307	0,019 (0,19865)
1e+05	16,5908	12,2121	0,0552	0,019 (1,97427)
1e+06	165,163	117,619	0,0593	0,197 (19,7673)
1e+07	1.652,23	1.171,07	0,2785	1,976 (197,676)
1e+08	16.519,8	11.694,9	2,1463	19,76 (1976,69)
1e+09	165.200	116.912	21,067	197,6 (19.766,6)

	FAS			
Número de Amostras	Forw.	Backw.	Perm. T.	Bootstrap
1e+04	1,62202	2,08269	0,02457	0,019 (0,19865)
1e+05	15,6713	15,4930	0,06144	0,019 (1,97632)
1e+06	153,387	149,793	0,08396	0,197 (19,7673)
1e+07	1.537,01	1.491,36	0,24985	1,976 (197,667)
1e+08	22.395,9	21.738,2	2,01933	19,75 (1975,54)
1e+09	153.303	222.637	19,8615	197,6 (19.760,1)

	RS			
Número de Amostras	Forw.	Backw.	Perm. T.	Bootstrap
1e+04	2,22413	2,00077	0,02867	0,020 (0,20070)
1e+05	20,2015	14,2601	0,05734	0,019 (1,98042)
1e+06	202,158	136,096	0,10240	0,198 (19,8124)
1e+07	2.021,83	1.352,94	0,30310	1,981 (198,118)
1e+08	20.218,3	13.512,8	2,51904	19,81 (1.981,28)
1e+09	202.183	135.139	24,7276	198,1 (19.811,4)

Capítulo 6

Conclusão

Este trabalho apresentou o desempenho, em termos de precisão, de diferentes técnicas de simulação conhecidas aplicadas à resolução de modelos Markovianos e propôs a utilização do método *Bootstrap* adaptado a uma nova técnica de simulação, com o objetivo de melhorar a precisão dos resultados.

Durante a realização dos testes, foi observado que a simulação é uma solução bastante onerosa em termos de tempo de processamento, sendo a mesma alvo constante de aperfeiçoamentos para que resultados com grandes modelos possam ser obtidos em um tempo hábil. Utilizar-se de paradigmas de paralelização ao implementar as técnicas de simulação é uma alternativa claramente viável, pois reduz significativamente este tempo conforme os gráficos de *speedup* apresentados neste trabalho, uma vez que as amostras podem ser coletadas de forma independente. Mesmo assim, outras primitivas podem ser empregadas, como a utilização do método *Aliasing* [40] para acelerar a escolha das transições no modelo. Além disso, o estudo de propriedades de monotonicidade podem auxiliar na redução das trajetórias disparadas durante a coleta das amostras, para a técnica *Backward Coupling Simulation*.

Um fator que merece destaque é o tempo de processamento das técnicas *Permanence Time Simulation* e *Bootstrap Simulation*, por propiciarem melhor desempenho em relação às demais. Foi mostrado também que de posse do tempo despendido com a coleta de um determinado número de amostras é possível inferir o tempo necessário para a coleta de quantidades maiores das mesmas, uma vez que este tempo aumenta nas mesmas proporções.

Quanto a precisão dos resultados obtidos com a aplicação das diferentes técnicas de simulação, que corresponde ao foco principal deste trabalho, conclui-se que os mesmos apresentam melhor precisão com a técnica *Bootstrap Simulation*, levando em consideração principalmente os índices com a coleta de amostras maiores do que $1.000.000$ para os modelos analisados. Com relação a técnica *Backward*, acredita-se que ela demonstre resultados mais precisos para a coleta de uma quantidade pequena de amostras (10.000), por ser caracterizada como não tendenciosa, embora para amostras maiores não tenha superado as demais, nos experimentos deste trabalho.

Com relação a técnica *Forward*, a necessidade de ter que definir o tamanho de suas trajetórias, torna difícil sua calibragem. Além disso, ter que escolher um estado inicial é outro problema apresentado por ela, porém, constatou-se que utilizando estados iniciais variáveis, este problema é amenizado

tornando este fator pouco impactante na precisão.

Quanto a técnica *Permanence Time*, para grandes quantidades de amostras observou-se bom equilíbrio entre tempo de processamento e precisão dos resultados.

6.1 Contribuição

Este trabalho evidenciou particularidades de técnicas de simulação conhecidas apontando as principais vantagens e desvantagens das mesmas. Com isso, os resultados aqui apresentados servirão como base para o emprego de novos métodos e técnicas de aperfeiçoamento da simulação que possam melhorar ainda mais a precisão dos resultados e reduzir seu tempo de processamento. A principal contribuição deste trabalho foi propor a adaptação do método *Bootstrap* como uma nova técnica de simulação aplicada à resolução de modelos Markovianos. Esta técnica, por sua vez, apresentou resultados satisfatórios reduzindo, em alguns casos, o erro relativo médio e máximo comparado com a solução iterativa. Além disso uma abordagem paralela para as técnicas conhecidas, demonstrou sua viabilidade na implementação das mesmas.

6.2 Trabalhos Futuros

Alguns trabalhos futuros podem ser motivados com a realização deste trabalho para auxiliar na resolução de modelos Markovianos através de soluções por simulação. Alguns deles são descritos a seguir:

- Paralelização da Técnica *Bootstrap Simulation* - o fato das técnicas de simulação conhecidas terem demonstrado *speedup* lineares permite concluir que esta primitiva é uma boa prática para a implementação das mesmas. Por este motivo, o estudo na paralelização da técnica *Bootstrap Simulation* é um dos trabalhos futuro à esta dissertação;

- Modelos com Eventos Raros - outro trabalho futuro é com relação a uma análise aprofundada na simulação de modelos que apresentam eventos de ocorrência rara, determinados por frequências muito pequenas do disparo das transições. Além da variação do RSS dos modelos, acredita-se que as taxas também devem acarretar variações na precisão dos seus resultados. Desta forma, a simulação destes modelos pode demonstrar comportamentos ainda não observados até o momento;

- Categorização dos Modelos - como mencionado, o estudo de modelos com taxas que determinam transições muito raras podem apresentar comportamentos diferentes e influenciar na precisão dos resultados. Além disso, o modelo RS, ao contrário dos demais utilizados neste trabalho, não demonstrou diferenças significativas na precisão, mesmo com a utilização de diferentes técnicas de simulação. Dito isso, identificar possíveis classes de modelos para que

possa ser realizada uma categorização dos mesmos ao aplicar diferentes técnicas de simulação, ou até mesmo propor uma nova técnica que apresente melhor eficiência em termos de precisão para elas é um trabalho futuro a ser realizado;

- Simulação do Formalismo SAN - este trabalho propôs um estudo na precisão dos resultados simulando trajetórias nas Cadeias de Markov de modelos SAN. Outro trabalho futuro é propor uma solução por simulação, específico para este formalismo, para avaliar o desempenho em relação ao consumo de memória aplicado a resolução do mesmo. Além disso, poderiam ser aplicadas todas as técnicas de simulação aqui apresentadas e também empregados métodos conhecidos para agilizar o tempo de processamento e reduzir ainda mais o consumo de memória, através de outros formatos de armazenamento.

Referências Bibliográficas

[1] A. D. Aczel. *Probability 1: The Book That Proves There is Life in Outer Space.* Harvest Books, Orlando, Florida, 1998.

[2] M. Ajmone-Marsan, G. Conte, and G. Balbo. A Class of Generalized Stochastic Petri Nets for the Performance Evaluation of Multiprocessor Systems. *ACM Transactions on Computer Systems*, 2(2):93–122, 1984.

[3] D. Aldous and J. A. Fill. *Reversible Markov Chains and Random Walks on Graphs.* University of California, Berkeley, 1993.

[4] V. Amoia, G. De Micheli, and M. Santomauro. Computer-Oriented Formulation of Transition-Rate Matrices via Kronecker Algebra. *IEEE Transactions on Reliability*, R-30(2):123–132, 1981.

[5] W. E. Arnoldi. The principle of minimized iterations in the solution of the matrix eigenvalue problem. *Quarterly of Applied Mathematics*, 9:17–29, 1951.

[6] A. Benoit, L. Brenner, P. Fernandes, B. Plateau, and W. J. Stewart. The PEPS Software Tool. In *Computer Performance Evaluation (TOOLS 2003)*, volume 2794 of *LNCS*, pages 98–115, Urbana, IL, USA, 2003. Springer-Verlag Heidelberg.

[7] J. Bernoulli. On the law of large numbers. *Ars Conjectandi: Usum e Applicationem Praecedentis Doctrinae in Civilibus, Moralibus e Oeconomicis*, 1713.

[8] C. Bertolini, L. Brenner, P. Fernandes, A. Sales, and A. F. Zorzo. Structured Stochastic Modeling of Fault-Tolerant Systems. In *12th IEEE/ACM Internacional Symposium on Modelling, Analysis and Simulation on Computer and Telecommunication Systems*, pages 139–146, MASCOTS'2004, Volendam, The Netherlands, October 2004. IEEE Press.

[9] G. Bolch, S. Greiner, H. de Meer, and K. S. Trivedi. *Queueing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications.* John Wiley & Sons, 1998.

[10] L. Brenner, P. Fernandes, J.-M. Fourneau, and B. Plateau. Modelling Grid5000 point availability with SAN. *Electronic Notes in Theoretical Computer Science (ENTCS)*, 232:165–178, March 2009.

[11] L. Brenner, P. Fernandes, and A. Sales. The Need for and the Advantages of Generalized Tensor Algebra for Kronecker Structured Representations. *International Journal of Simulation: Systems, Science & Technology*, 6(3-4):52–60, February 2005.

[12] R. Buyya. *High Performance Cluster Computing: Architectures and Systems*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1999.

[13] R. Chanin, M. Corrêa, P. Fernandes, A. Sales, R. Scheer, and A. F. Zorzo. Analytical Modeling for Operating System Schedulers on NUMA Systems. In 2^{nd} *International Workshop on Practical Applications of Stochastic Modelling (PASM 2005)*, pages 1–18, Newcastle, UK (submitted), July 2005.

[14] R. M. Czekster, P. Fernandes, J. M. Vincent, and T. Webber. Split: a flexible and efficient algorithm to vector-descriptor product. In *Proceedings of the 2nd international conference on Performance evaluation methodologies and tools (ValueTools'07)*, Brussels, Belgium, Belgium, 2007. Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering (ICST).

[15] M. Davio. Kronecker Products and Shuffle Algebra. *IEEE Transactions on Computers*, C-30(2):116–125, 1981.

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

[17] F. L. Dotti, P. Fernandes, A. Sales, and O. M. Santos. Modular Analytical Performance Models for Ad Hoc Wireless Networks. In 3^{rd} *International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks*, pages 164–173, Trentino, Italy, April 2005. IEEE Press.

[18] B. Efron. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7(1):1–26, 1979.

[19] P. Fernandes. *Meťhodes numériques pour la solution de systémes Markoviens á grand espace d'états*. PhD thesis, Institut National Polytechnique de Grenoble, France, 1998.

[20] P. Fernandes, B. Plateau, and W. J. Stewart. Efficient descriptor-vector multiplication in Stochastic Automata Networks. *Journal of the ACM SIGMETRICS*, 45(3):381–414, 1998.

[21] O. Häggström. *Finite Markov Chains and Algorithmic Applications.* Cambridge University Press, 2002. Based on lecture notes.

[22] J. Hillston. *A compositional approach to performance modelling.* Cambridge University Press, New York, USA, 1996.

[23] J. Hillston and L. Kloul. An Efficient Kronecker Representation for PEPA models. In L. de Alfaro and S. Gilmore, editors, *Proceedings of the first joint PAPM-PROBMIV Workshop)*, pages 120–135, Aachen, Germany, September 2001. Springer-Verlag Heidelberg.

[24] S. Juneja and P. Shahabuddin. Fast simulation of markov chains with small transition probabilities. *Manage. Sci.*, 47(4):547–562, 2001.

[25] G. M. Karniadakis and R. M. Kirby. *Parallel Scientific Computing in C++ and MPI.* Cambridge University Press, New York, NY, USA, 2003.

[26] A. M. Law and D. W. Kelton. *Simulation Modeling and Analysis.* McGraw-Hill Higher Education, 2000.

[27] L.Brenner, P.Fernandes, B.Plateau, and I.Sbeity. PEPS2007 - Stochastic Automata Networks Software Tool. In *QEST 2007*, pages 163–164. IEEE Press, 2007.

[28] B. F. J. Manly. *Randomization, Bootstrap and Monte Carlo Methods in Biology*. Chapman & Hall/CRC, second edition, 1997.

[29] L. Mokdad, J. Ben-Othman, and A. Gueroui. Quality of Service of a Rerouting Algorithm Using Stochastic Automata Networks. In 6^{th} *IEEE Symposium on Computers and Communications*, pages 338–343, Hammamet, Tunisia, July 2001. IEEE Computer Society.

[30] PEPS. Performance Evaluation of Parallel Systems. Capturado em: http://www-id.imag.fr/Logiciels/peps/, Dezembro 2009.

[31] B. Plateau. On the stochastic structure of parallelism and synchronization models for distributed algorithms. *SIGMETRICS Performance Evaluation Review*, 13(2):147–154, 1985.

[32] B. Plateau and K. Atif. Stochastic Automata Networks for modelling parallel systems. *IEEE Transactions on Software Engineering*, 17(10):1093–1108, October 1991.

[33] J. G. Propp and D. B. Wilson. Exact Sampling with Coupled Markov Chains and Applications to Statistical Mechanics. *Random Structures and Algorithms*, 9(1–2):223–252, 1996.

[34] Y. Saad and M. H. Schultz. Gmres: a generalized minimal residual algorithm for solving non-symmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 7(3):856–869, 1986.

[35] K. Sayre. *Improved techniques for software testing based on Markov chain usage models.* PhD thesis, University of Tennessee, Knoxville, USA, 1999.

[36] R. E. Shannon. Introduction to the art and science of simulation. In *WSC '98: Proceedings of the 30th conference on Winter simulation*, pages 7–14, Los Alamitos, CA, USA, 1998. IEEE Computer Society Press.

[37] W. J. Stewart. *Introduction to the numerical solution of Markov chains.* Princeton University Press, Princeton, NJ, USA, 1994.

[38] W. J. Stewart. *Probability, Markov Chains, Queues, and Simulation: The Mathematical Basis of Performance Modeling.* Princeton University Press, Princeton, NJ, USA, 2009.

[39] J.-M. Vincent and J. Vienne. Perfect simulation of index based routing queueing networks. *ACM SIGMETRICS Performance Evaluation Review*, 34(2):24–25, 2006.

[40] A. J. Walker. An Efficient Method for Generating Discrete Random Variables with General Distributions. *ACM Trans. Math. Softw.*, 3(3):253–256, 1977.

[41] J. A. Whittaker and M. G. Thomason. A Markov chain model for statistical software testing. *IEEE Transactions on Software Engineering*, 20(10):812–824, 1994.

Anexo A

Arquivos Fontes da Ferramenta PEPS

A.1 *Alternate Service Pattern* - ASP

```
//=== ASP - Alternate Service Pattern (P=2) ===

identifiers

  // Queue Capacity (queue 1)
  C1 = 2;
  // Queue Capacity (queue 2)
  C2 = 2;
  // Queue Capacity (queue 3)
  C3 = 2;
  // Queue Capacity (queue 4)
  C4 = 2;

  // Array for the queue capacities
  K1 = [0..2];
  K2 = [0..2];
  K3 = [0..2];
  K4 = [0..2];

  // Rates
  arrive1     = 1;
  arrive2     = 1;
  service1    = 1;
  service2    = 1;
  service3_41 = 1;
  service3_42 = 1;
  service4    = 1;
  pi1         = (1/2);
  pi2         = (1/2);

events

  loc l1        (arrive1);
  loc l2        (arrive2);
  syn rot_1_3   (service1);
  syn rot_2_3   (service2);
```

```
  syn rot_3_41  (service3_41);
  syn rot_3_42  (service3_42);
  loc rot_4_out (service4);

reachability = 1;

network ASP_2c (continuous)

  aut Q1
    stt n[K1] to (++) l1
              to (--) rot_1_3

  aut Q2
    stt n[K2] to (++) l2
              to (--) rot_2_3

  aut Q3
    stt n[K3] to (++) rot_1_3 rot_2_3
              to (--) rot_3_41 rot_3_42
    from n[2] to (n[2]) rot_2_3

  aut Q4
    stt n[K4] to (++) rot_3_41 rot_3_42
              to (--) rot_4_out

  aut Services
    stt p1
      to (p1) rot_3_41(pi1)
      to (p2) rot_3_41(pi2)
    stt p2
      to (p1) rot_3_42(pi1)
      to (p2) rot_3_42(pi2)

results

used = (st Q1);
```

A.2 *First Available Server* - FAS

```
//=== FAS - First Available Server (N=09) ===

identifiers

  // Acquisition rate
  lambda = 2.000000;
  // Release rate
  mu = 1.000000;

events

  loc t1 (lambda);
  syn t2 (lambda);
  syn t3 (lambda);
  syn t4 (lambda);
```

```
  syn t5 (lambda);
  syn t6 (lambda);
  syn t7 (lambda);
  syn t8 (lambda);
  syn t9 (lambda);
  loc r1 (mu);
  loc r2 (mu);
  loc r3 (mu);
  loc r4 (mu);
  loc r5 (mu);
  loc r6 (mu);
  loc r7 (mu);
  loc r8 (mu);
  loc r9 (mu);

reachability = 1;

network FAS10c (continuous)

  aut Server1 stt idle to (busy) t1
              stt busy to (idle) r1
                       to (busy) t2 t3 t4 t5 t6 t7 t8 t9

  aut Server2 stt idle to (busy) t2
              stt busy to (idle) r2
                       to (busy) t3 t4 t5 t6 t7 t8 t9

  aut Server3 stt idle to (busy) t3
              stt busy to (idle) r3
                       to (busy) t4 t5 t6 t7 t8 t9

  aut Server4 stt idle to (busy) t4
              stt busy to (idle) r4
                       to (busy) t5 t6 t7 t8 t9

  aut Server5 stt idle to (busy) t5
              stt busy to (idle) r5
                       to (busy) t6 t7 t8 t9

  aut Server6 stt idle to (busy) t6
              stt busy to (idle) r6
                       to (busy) t7 t8 t9

  aut Server7 stt idle to (busy) t7
              stt busy to (idle) r7
                       to (busy) t8 t9

  aut Server8 stt idle to (busy) t8
              stt busy to (idle) r8
                       to (busy) t9

  aut Server9 stt idle to (busy) t9
              stt busy to (idle) r9

results

used1 = (st Server1 == busy);
```

```
used2 = (st Server2 == busy);
used3 = (st Server3 == busy);
used4 = (st Server4 == busy);
used5 = (st Server5 == busy);
used6 = (st Server6 == busy);
used7 = (st Server7 == busy);
used8 = (st Server8 == busy);
used9 = (st Server9 == busy);
```

A.3 *Resource Sharing* - RS

```
//=== Resource Sharing (N=10 P=5) ===

identifiers

  // Number of Resources
  R = 5;
 _R = [0..5];
  // Acquisition rate
  lambda = 1.000000;
  // Release rate
  mu = 2.000000;

events

  syn t0 (lambda);
  syn t1 (lambda);
  syn t2 (lambda);
  syn t3 (lambda);
  syn t4 (lambda);
  syn t5 (lambda);
  syn t6 (lambda);
  syn t7 (lambda);
  syn t8 (lambda);
  syn t9 (lambda);
  syn r0 (mu);
  syn r1 (mu);
  syn r2 (mu);
  syn r3 (mu);
  syn r4 (mu);
  syn r5 (mu);
  syn r6 (mu);
  syn r7 (mu);
  syn r8 (mu);
  syn r9 (mu);

reachability = ((nb [a9..a0] on <= R) && (nb [a9..a0] on == st Res));

network RS10_16 (continuous)

  aut Res
    stt using[_R]
      to (++)
        t0 t1 t2 t3 t4 t5 t6 t7 t8 t9
```

```
      to (--)
        r0 r1 r2 r3 r4 r5 r6 r7 r8 r9

  aut a9
    stt off to (on)  t9
    stt on  to (off) r9

  aut a8
    stt off to (on)  t8
    stt on  to (off) r8

  aut a7
    stt off to (on)  t7
    stt on  to (off) r7

  aut a6
    stt off to (on)  t6
    stt on  to (off) r6

  aut a5
    stt off to (on)  t5
    stt on  to (off) r5

  aut a4
    stt off to (on)  t4
    stt on  to (off) r4

  aut a3
    stt off to (on)  t3
    stt on  to (off) r3

  aut a2
    stt off to (on)  t2
    stt on  to (off) r2

  aut a1
    stt off to (on)  t1
    stt on  to (off) r1

  aut a0
    stt off to (on)  t0
    stt on  to (off) r0

results
  used = (st Res);
```