Simulação de modelos Markovianos utilizando a técnica *Bootstrap*[*]

Dione Taschetto[1], Paulo Fernandes[1], Thais Webber[1], Afonso Sales[1]

[1]Faculdade de Informática – Pontifícia Universidade Católica do Rio Grande do Sul
Av. Ipiranga, 6681, Prédio 32 – 90619-900 – Porto Alegre – RS – Brasil

{dione.taschetto, paulo.fernandes, thais.webber, afonso.sales}@pucrs.br

Abstract. *Simulation is an interesting alternative to solve Markovian models. However, when compared to analytical and numerical solutions it suffers from a lack of confidence in the accuracy of results due to the very nature of simulation, which is the choice of samples through pseudorandom generation. This paper proposes a different way to simulate Markovian models by using a Bootstrap-based statistical technique to minimize the effect of sample choices. The effectiveness of the proposed technique, called Bootstrap Simulation, is compared to the exact results of numerical solution for a set of examples described using Stochastic Automata Networks modeling formalism.*

Resumo. *Simulação é uma alternativa interessante para a solução de modelos Markovianos. No entanto, comparando-a com métodos analíticos e numéricos percebe-se uma falta de confiança na precisão dos resultados pela própria natureza da simulação que escolhe amostras de funcionamento através da geração de números pseudo-aleatórios. Este artigo propõe uma nova forma de simular modelos Markovianos pelo uso de uma técnica estatística baseada em Bootstrap para reduzir os efeitos de escolha de amostras. A eficácia da técnica proposta, chamada Simulação Bootstrap, é comparada aos resultados exatos obtidos pela solução numérica de um conjunto de exemplos descritos pelo formalismo de Redes de Autômatos Estocásticos.*

1. Introdução

É possível representar o comportamento de um sistema real através da descrição dos diferentes estados que o mesmo pode ocupar, indicando como este muda de um estado para outro no tempo. Um formalismo amplamente utilizado para a representação de sistemas é *Cadeias de Markov* [Stewart 2009]. Por sua facilidade de representação, Cadeias de Markov são frequentemente empregadas na modelagem de sistemas computacionais, entretanto, a utilização deste formalismo não se restringe somente ao ramo da computação, mas também é empregado em diferentes áreas da ciência, tais como: biologia, química, física, ciências sociais, bem como engenharia e administração [Stewart 2009].

Porém, uma das limitações das Cadeias de Markov é a ocorrência da *explosão do seu espaço de estados*, a qual ocorre quando as configurações possíveis do sistema aumentam significativamente, tornando sua solução impraticável dado os recursos computacionais disponíveis [Stewart 2009]. Dentre as alternativas de modelagem conhecidas, as

[*]Este trabalho é financiado pela Petrobras (0050.0048664.09.9). Paulo Fernandes é também financiado pelo CNPq-Brasil (307272/2007-9) e Afonso Sales recebe financiamento da CAPES-Brasil (02388/09-0).

quais tentam minimizar este problema, sobressaem-se os formalismos estruturados, tais como, *Stochastic Automata Networks* (SAN) [Fernandes et al. 1998], *Generalized Stochastic Petri Nets* (GSPN) [Donatelli 1993] e *Performance Evaluation Process Algebras* (PEPA) [Hillston 1996]. A utilização do formalismo SAN se sobressai quando comparado a outros formalismos por apresentar uma forma de representação compacta em memória, permitindo assim modelar sistemas com maior número de estados, e também por permitir uma representação compacta de interações complexas entre os componentes do sistema.

Dentre as soluções utilizadas para a resolução de sistemas através de formalismos, destacam-se principalmente os métodos numéricos e a simulação. Os métodos numéricos consistem, normalmente, no emprego de métodos matemáticos iterativos, tais como o *Método da Potência* [Stewart 2009], o *Método de Arnoldi* [Arnoldi 1951] e o *Método GMRES* (*Generalized Minimal RESidual method*) [Saad and Schultz 1986]. A simulação baseia-se geralmente na geração de números pseudo-aleatórios que atuam nas decisões relacionadas ao disparo de transições no modelo e na interação entre os componentes do sistema analisado. A simulação também consiste em uma aproximação da solução do modelo, sendo necessário executá-la por um tempo de simulação suficientemente grande para obter-se um conjunto de amostras significativas.

Embora as soluções numéricas destes modelos apresentem confiabilidade nos resultados, as mesmas deparam-se com limitações que estão ligadas, principalmente, no que diz respeito a explosão do espaço de estados do modelo. Em relação ao formalismo SAN, algumas técnicas para otimizar a resolução de modelos têm sido aprimoradas com a finalidade de acelerar a convergência na resolução dos métodos iterativos [Fernandes et al. 1998]. Quando se deseja resolver modelos com um número de estados que ultrapassam a capacidade atual de resolução dos métodos numéricos, 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 outros métodos, tornando possível a resolução de modelos com um número de estados ainda maior.

Ao resolver modelos através de simulação, um fator de grande relevância nesse processo é a precisão dos resultados. Como a simulação corresponde a aproximações da solução analítica, existem erros associados aos resultados obtidos com simulação. Nesse sentido existem diferentes técnicas de simulação conhecidas que podem ser aplicadas aos modelos Markovianos na tentativa de obter resultados mais precisos, tais como a simulação *tradicional* [Ross 2002], simulação *Monte Carlo* [Häggström 2002] e simulação por *amostragem perfeita* [Propp and Wilson 1996]. Dentre elas, destaca-se a simulação tradicional, a qual consiste no disparo de uma trajetória na cadeia de Markov e contabiliza a quantidade de visitas em cada um dos estados da cadeia, sendo cada estado visitado uma amostra da simulação. Nesta técnica define-se um estado inicial de forma arbitrária e também o tamanho da trajetória. O tamanho ideal da mesma pode ser inferido através de cálculos de intervalos de confiança. Por fim, a divisão da quantidade de amostras coletadas de cada estado do modelo pela quantidade total de amostras define o vetor de probabilidades resultante do modelo, *i.e.*, a probabilidade do sistema se encontrar em cada estado.

Utilizada para resolver uma grande variedade de problemas de estimativa em diversas áreas (*e.g.*, algoritmos de votação [Bauer and Kohavi 1999]), a técnica *Bootstrap*,

proposta por Efron [Efron 1979], apresenta resultados bastante satisfatórios na redução de "ruídos" observados em algumas aplicações. No contexto de simulação, esse ruído corresponde ao erro observado em relação à solução analítica comparada à solução aproximada por simulação. A técnica consiste na reamostragem de uma determinada amostra, cuja qual, neste contexto, corresponde a um conjunto de valores e não mais a um único valor conforme a simulação tradicional. Sendo assim, os resultados obtidos através das médias dessas reamostragens tendem a ser melhores que a média de uma única amostra.

Em busca de resultados mais precisos, o objetivo principal deste trabalho é propor uma nova técnica de simulação para solução de modelos Markovianos aplicando a técnica *Bootstrap* e evidenciar quantitativamente a precisão dos resultados observados para alguns modelos ao compará-la com a simulação tradicional. O foco das comparações e análises que serão realizadas neste trabalho será em relação ao vetor de probabilidades resultante da técnica de simulação proposta e o resultado da solução aproximada pelo método iterativo, analisando o erro relativo entre os mesmos.

O restante do trabalho segue organizado da seguinte maneira. Na Seção 2, apresenta-se brevemente os conceitos do formalismo de Cadeias de Markov, bem como do formalismo de Redes de Autômatos Estocásticos (SAN), ilustrando suas respectivas representações e os modelos utilizados neste trabalho. Na Seção 3, apresentam-se as principais características empregadas na simulação de modelos Markovianos. Na Seção 4, explicita-se a técnica *Bootstrap* e como esta pode ser empregada no contexto de simulação de modelos Markovianos. Na Seção 5 são apresentados alguns resultados obtidos com a simulação *Bootstrap* comparada com a simulação tradicional. Por fim, na Seção 6 é discutida a conclusão e os trabalhos futuros referentes a este artigo.

2. Formalismos de Modelagem

O emprego de formalismos Markovianos 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 [Brenner et al. 2009, Dotti et al. 2005, Bertolini et al. 2004, Sayre 1999]. 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 [Stewart 2009]. 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 [Stewart 2009] de tempo contínuo ou discreto, as quais são definidas por distribuições de probabilidade exponenciais ou geométricas respectivamente. Este trabalho irá tratar apenas sistemas de tempo contínuo e espaço de estados discreto.

Na Figura 1, apresenta-se um exemplo de uma cadeia de Markov de um sistema a fim de facilitar a compreensão da representação deste formalismo. Esta cadeia apresenta os estados $S = \{00, 01, 02, 10, 11, 12\}$ possíveis do sistema representados por círculos 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 determinam a frequência do disparo das transições.

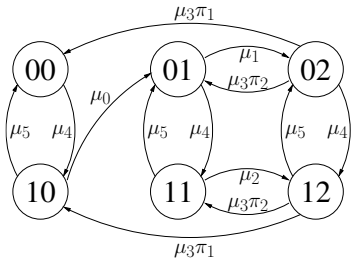


Figura 1. Exemplo de uma cadeia de Markov

Esta mesma cadeia (Figura 1) pode também ser representada através de uma *matriz de transição*, conforme pode ser observado na Tabela 1. Cada linha i e coluna j da matriz indicam a taxa de transição do estado i para o estado j na cadeia de Markov, sendo a dimensão da matriz igual ao espaço de estados do modelo, *i.e.*, a cardinalidade do conjunto S dado por $|S|$. Na resolução do sistema de equações gerado pelo modelo, é necessário verificar que a matriz de transição seja um *gerador infinitesimal*[1] Q [Stewart 2009]. Na Tabela 2, apresenta-se o gerador infinitesimal da cadeia de Markov da Figura 1, onde adiciona-se o complemento da soma de todos os elementos não diagonais de cada linha à sua diagonal.

Tabela 1. Matriz de Transição correspondente à cadeia de Markov da Figura 1

		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

Tabela 2. Gerador Q correspondente à cadeia de Markov da Figura 1

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

A partir do gerador infinitesimal Q, extrai-se um sistema de equações na forma $\pi Q = 0$, onde deseja-se encontrar o vetor de probabilidade π (não confundir o vetor π com as probabilidades π_1 e π_2 multiplicadas pela taxa μ_3 da matriz), cuja a soma de seus elementos é igual a 1. 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 π é o tradicional método da Potência [Stewart 2009]. Este vetor 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 uma diferença aceitável entre vetores de duas iterações. Para que este método possa ser aplicado, é necessário primeiramente transformar o gerador infinitesimal Q em uma matriz estocástica[2] P. Este

[1]Gerador infinitesimal é uma matriz cuja a soma dos elementos de cada linha deve ser igual a $zero$.

[2]Uma matriz estocástica é uma matriz de elementos positivos cuja a soma de cada linha é igual a 1.

processo corresponde à discretização da matriz Q [Stewart 2009] que pode ser realizada dividindo toda a matriz pelo seu maior elemento Λ_{max} em módulo, e somando a esta uma matriz identidade I de mesma dimensão da seguinte forma:

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

Um dos principais problemas das cadeias de Markov é a 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 Q [Stewart 2009] pois aumenta também a dimensão da matriz P envolvida no processo iterativo. Para minimizar este problema, formalismos estruturados como SAN podem ser utilizados.

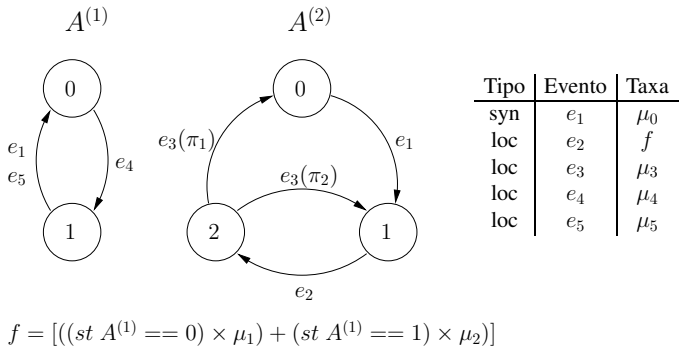
2.2. Redes de Autômatos Estocásticos

Redes de Autômatos Estocásticos (SAN) [Plateau 1985] é um formalismo baseado em Cadeias de Markov, o qual descreve de maneira modular sistemas complexos com grandes espaços de estados. Este formalismo consegue aumentar o poder de solução obtido com a utilização de Cadeias de Markov, propondo para tal um novo formato de armazenamento e solução baseado em álgebra tensorial que minimiza o problema da explosão do espaço de estados [Fernandes et al. 1998].

Um modelo descrito por 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. O disparo de uma transição em um autômato é dependente da ocorrência de um evento. Quando um evento ocorre somente em um único autômato, este é denominado *evento local* (por exemplo, o evento e_5 que ocorre somente no autômato $A^{(1)}$ realiza a transição do estado 1 para o estado 0 neste autômato). Entretanto, quando um evento aparece em dois ou mais autômatos, este é denominado como *evento sincronizante*, pois todos os autômatos envolvidos mudarão seus estados. Por exemplo, a ocorrência do evento e_1 faz com que o autômato $A^{(1)}$ mude do estado 1 para 0 ao mesmo tempo que $A^{(2)}$ muda do estado 0 para 1. Todo evento possui uma taxa de ocorrência associada que pode ser um valor *constante* ou *funcional* (*i.e.*, depende da avaliação de uma função). Por exemplo, o evento local e_2 esta associado a uma taxa funcional f cuja a função avalia o estado do autômato $A^{(1)}$ para realizar a transição no autômato $A^{(2)}$. Neste caso, se $A^{(1)}$ se encontra em 0, a taxa de ocorrência aplicada é μ_1, caso esteja em 1, a taxa é μ_2.

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 rotação* do evento, *i.e.*, quando da ocorrência do evento e_3, $A^{(2)}$ estando em 2 irá para 0 com probabilidade π_1 ou para 1 com probabilidade π_2 (onde $\pi_1 + \pi_2 = 1$).

O estado em que um autômato se encontra em um dado momento é chamado de *estado local*, enquanto que o conjunto de estados em que todos os autômatos se encontram é denominado *estado global*, o qual corresponde ao estado na Cadeia de Markov subjacente ao modelo. A cadeia de Markov subjacente ao modelo da Figura 2 é a da Figura 1, apresentada anteriormente. Esse mapeamento é feito pelas combinações de estados possíveis entre $A^{(1)}$ e $A^{(2)}$ da SAN. Para entender este mapeamento, considere o



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. Exemplo de uma rede de autômatos estocásticos

estado local de $A^{(1)}$ igual a 0 e de $A^{(2)}$ igual a 1, neste caso, o estado global na cadeia de Markov da Figura 1 é o estado 01. Desta forma, o espaço de estados é calculado pelo produto cartesiano do espaço de estados de cada autômato da SAN (chamado de *espaço de estados produto* - *Product State Space* - PSS). No caso do modelo da Figura 2, o PSS é $2 \times 3 = 6$. Entretanto, existem modelos em que o PSS comporta estados que não são válidos no comportamento do sistema. Denomina-se este conjunto de estados válidos (ou atingíveis) do modelo como *espaço de estados atingíveis* (*Reachable State Space* - RSS).

2.3. Exemplos de modelos

Exemplos do uso de SAN podem ser encontrados em diversos trabalhos [Sales and Plateau 2009, Brenner et al. 2009, Dotti et al. 2005, Bertolini et al. 2004]. Para este artigo, foram selecionados três modelos: Padrão de Serviço Alternado (*Alternate Service Pattern* - ASP); Primeiro Servidor Disponível (*First Available Server* - FAS); e Compartilhamento de Recursos (*Resource Sharing* - RS). Mais detalhes sobre estes modelos podem também ser encontrados em [Taschetto 2010].

O modelo ASP descreve uma Rede de Filas de Espera [Stewart 2009] aberta em que alguns servidores apresentam mais de um padrão de atendimento. Neste modelo, tem-se quatro filas, onde cada uma é representada por um autômato, cujo o espaço de estados é dado por $K + 1$, sendo K a capacidade da fila. Além disso, os padrões de serviço são representados por um autômato de P estados, sendo P o número de padrões. O PSS deste modelo é dado por $(K + 1)^4 \times P$. Neste modelo, o PSS é igual ao RSS.

No modelo FAS, descreve-se a disponibilidade de N servidores, onde cada servidor é representado por um autômato composto por dois estados (*disponível* ou *ocupado*). Neste exemplo, os pacotes devem ser tratados pelo primeiro servidor, caso este não esteja ocupado. Se o mesmo estiver ocupado, o pacote deve ser tratado pelo segundo servidor e assim por diante, de maneira que o pacote encontre o primeiro servidor disponível para tratá-lo. O PSS deste modelo é de 2^N estados, sendo o PSS também igual ao RSS.

O modelo RS é um exemplo clássico do compartilhamento de R recursos entre P processos. Cada processo é representado por um autômato composto por dois estados (*em repouso* ou *em uso*). Os recursos são representados por um autômato que indica o número de recursos R em uso. O PSS deste modelo é formado por $2^P \times (R + 1)$ estados, porém, ao contrário dos demais modelos, nem todos os seus estados são atingíveis (ou válidos).

3. Simulação de modelos Markovianos

Utilizando métodos numéricos iterativos é possível resolver um grande número de modelos. No entanto, conforme aumenta-se o espaço de estados dos mesmos, esgota-se a capacidade de resolução destes métodos, devido principalmente aos limites dos recurso computacionais disponíveis. Devido a este problema, destaca-se então a *simulação* [Ross 2002, Shannon 1998] como uma alternativa para a resolução de modelos com grandes espaços de estados.

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 numérica, o número de amostras coletadas deve ser grande suficiente para que os resultados possam se aproximar da solução estacionária. Note que um dos grandes problemas da simulação é a precisão dos resultados, uma vez que esta consiste na tiragem de números pseudo-aleatórios para determinar transições no modelo regidas por distribuições de probabilidades.

Destaca-se que o vetor de probabilidades será obtido simulando o RSS de um modelo SAN. Assume-se, então, um conjunto de estados $S = \{s_0, s_1, s_2, ..., s_r\}$, o qual corresponde ao RSS do modelo, a matriz estocástica P (onde P_{ij} corresponde ao elemento da linha i e coluna j da matriz) e um gerador de números pseudo-aleatórios $U(0..1)$ [Häggström 2002]. A matriz estocástica P é a discretização do gerador infinitesimal Q do modelo em questão. A transição entre os estados do modelo 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 resultado desta função indicará o próximo estado da cadeia de Markov durante a simulação, sendo s_i o estado corrente. A função de transição é definida genericamente pela expressão a seguir:

$$\phi(s_i, U) = \begin{cases} s_0 & \textit{para } U \in [0, P_{i0}) \\ s_1 & \textit{para } U \in [P_{i0}, P_{i0} + P_{i1}) \\ \vdots & \vdots \\ s_j & \textit{para } U \in \left[\sum_{l=0}^{j-1} P_{il}, \sum_{l=0}^{j} P_{il}\right) \\ \vdots & \vdots \\ s_r & \textit{para } U \in \left[\sum_{l=0}^{r-1} P_{ir}, 1\right] \end{cases} \tag{2}$$

Discutido o processo básico de simulação, que consiste no disparo de uma trajetória dada pela visita de estados no modelo, pode-se então apresentar a adaptação da técnica *Bootstrap* no contexto de simulação de modelos Markovianos.

4. Simulação *Bootstrap*

A técnica *Bootstrap* é aplicada no campo de estatística para derivar estimativas de erro padrão e intervalo de confiança para estimadores complexos 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 [Manly 1997]. 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 3. 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 3, 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*.

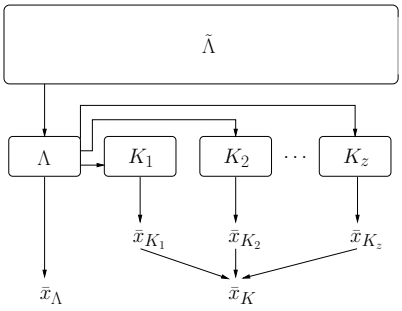


Figura 3. Técnica *Bootstrap*

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.

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 4 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 4, 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 4, $|\underline{\bar{b}}|$ 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*.

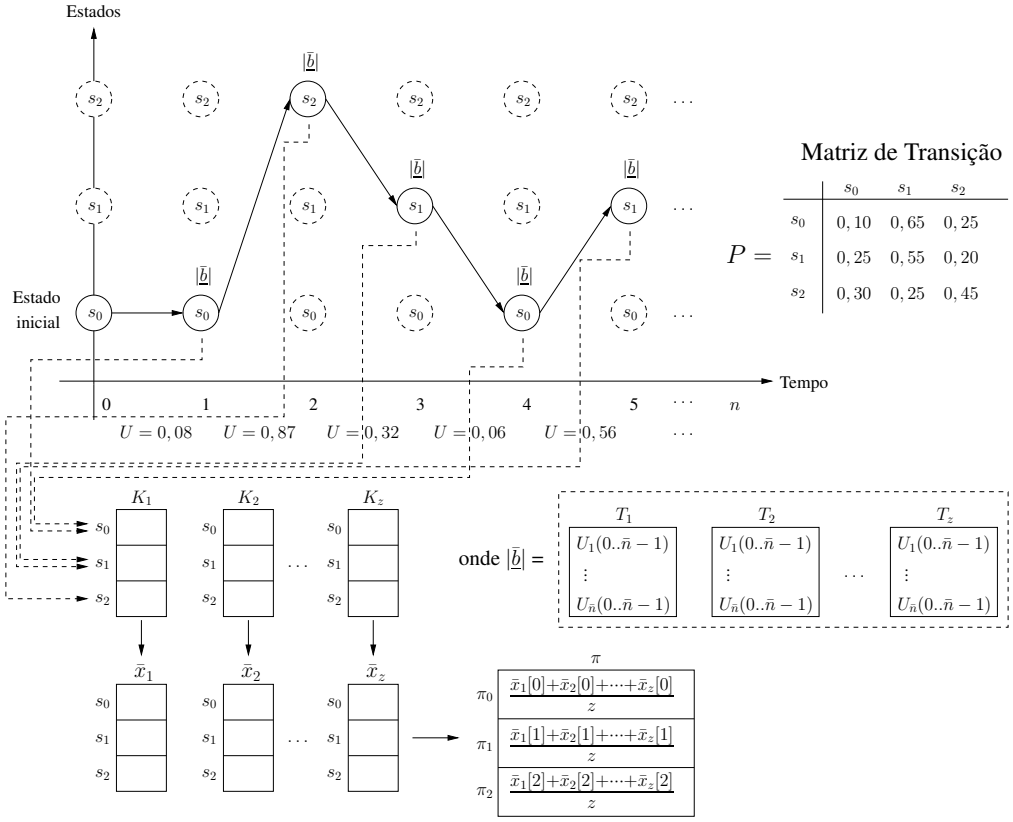


Figura 4. Simulação *Bootstrap*

O Algoritmo 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 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* [Aczel 1998] - equação (3) - o qual consiste em descobrir as chances de em um conjunto de pessoas, escolhidas aleatoriamente, existirem pares que façam aniversário na mesma data.

$$\text{Probabilidade} = 1 - \left[\frac{n!}{(n-\bar{n}+1)!} \times \left(\frac{1}{n}\right)^{\bar{n}+1} \right] \tag{3}$$

A fim de determinar um valor satisfatório para $\bar{n}$, definiu-se $\bar{n} = 20$ para o cálculo das probabilidades dada pela equação (3). 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%. 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 $\frac{1}{\bar{n}}$.

Algoritmo 1 Simulação *Bootstrap*

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 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*.

Na seção seguinte, apresentam-se os resultados numéricos da simulação *Bootstrap* para os modelos SAN mencionados na Seção 2.3.

5. Experimentos numéricos

Nesta seção serão feitas comparações dos resultados da simulação tradicional com a simulação *Bootstrap*, considerando o erro relativo médio em relação à solução aproximada obtida por método iterativo implementado na ferramenta PEPS [Brenner et al. 2007].

Para a realização dos testes, os modelos da Seção 2.3 foram parametrizados da seguinte maneira: *modelo ASP* - filas com capacidade para dois clientes ($K = 2$) e dois

padrões de serviço ($P = 2$); *modelo FAS* - número de servidores igual a 9 ($N = 9$); e *modelo RS* - 10 processos ($P = 10$) e 5 recursos ($R = 5$).

Uma questão ainda em aberto para a utilização desta técnica é a definição de um número eficiente de *bootstraps*. Para os gráficos apresentados nesta seção, este número foi fixado em 10 de forma arbitrária. Um estudo sobre diferentes quantidades de *bootstraps*, bem como o impacto no tempo de execução da simulação pode ser encontrados em [Taschetto 2010].

Os resultados das simulações são apresentados na Figura 5, considerando intervalo de confiança de 95% para 50 execuções, sendo que o eixo x de cada gráfico corresponde ao tamanho da trajetória 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.

Observando os gráficos referentes ao modelo ASP - (a) e (b) - nota-se que os resultados obtidos para a simulação *Bootstrap* demostraram maior precisão com a redução do erro relativo médio para a simulação de trajetórias de todos os tamanhos. Destacando a simulação da última trajetória, o erro relativo médio que é de $0,0061$ na simulação tradicional cai significativamente para $0,0009$ na simulação *Bootstrap*.

Comparando os resultados obtidos para o modelo FAS - gráficos (c) e (d) - nota-se também uma redução nos erros relativos médios na simulação *Bootstrap* para trajetórias maiores que 10^6.

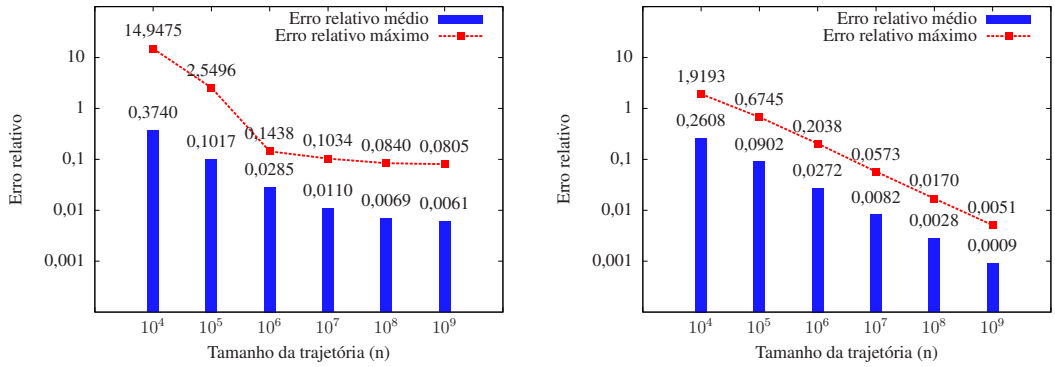
Em relação ao modelo RS, a simulação *Bootstrap* apresenta resultados similares aos resultados da simulação tradicional (*e.g.*, na trajetória de tamanho 10^9, o erro relativo médio na simulação tradicional foi de $0,0010$; enquanto que na simulação *Bootstrap* o erro foi de $0,0011$). No entanto, mesmo não obtendo ganhos de precisão, a mesma continua garantido um baixo erro relativo médio.

É fato que aumentar o tamanho da trajetória de simulação garante maior precisão, *i.e.*, a queda do erro relativo médio, para qualquer técnica de simulação. Como pode-se notar nos gráficos da Figura 5, a simulação *Bootstrap* apresentou de maneira geral ganhos de precisão para um mesmo tamanho de trajetória a partir de $n = 10^7$ quando comparada à simulação tradicional. Ressalta-se que no pior caso (modelo RS) a mesma obteve praticamente a mesma precisão da simulação tradicional.

6. Conclusão

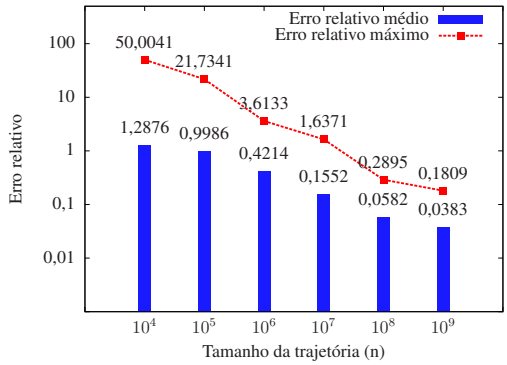
A principal contribuição deste artigo é propor a adaptação da técnica *Bootstrap* no contexto de simulação de modelos Markovianos no intuito de aumentar a precisão das probabilidades resultantes. Foram obtidos resultados satisfatórios para esta técnica, reduzindo, na maioria dos casos observados, o erro relativo médio e máximo em relação aos resultados da simulação tradicional.

Em relação à precisão dos resultados obtidos com a aplicação destas técnicas de simulação, conclui-se que os mesmos apresentaram melhor precisão com a técnica *Bootstrap*, levando em consideração principalmente os índices para trajetórias maiores do que $n = 10^6$ para os modelos analisados. Quanto à simulação tradicional, pode-se dizer que embora tenha apresentado erros maiores para trajetórias de menor tamanho, esta contém o melhor custo benefício quando a precisão requerida não for muito grande, uma

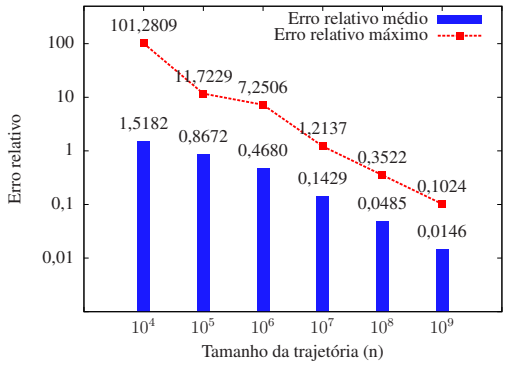


(a) Modelo ASP - Tradicional

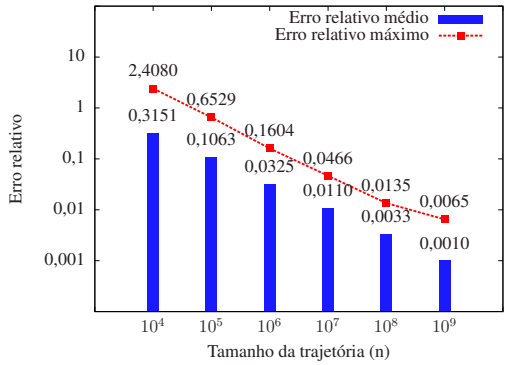
(b) Modelo ASP - *Bootstrap*



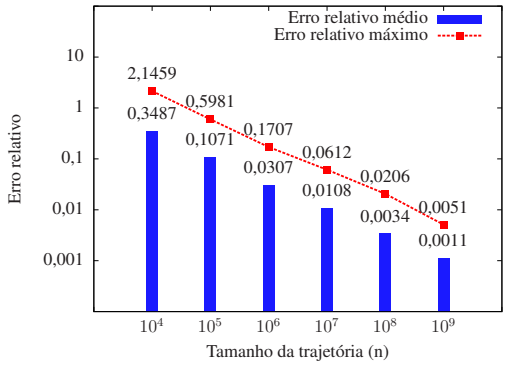
(c) Modelo FAS - Tradicional



(d) Modelo FAS - *Bootstrap*



(e) Modelo RS - Tradicional



(f) Modelo RS - *Bootstrap*

Figura 5. Resultados das simulações *tradicional* e *Bootstrap*

vez que esta técnica requer menor custo computacional em relação à simulação *Bootstrap*. Apesar disso, pelas vantagens oferecidas em relação à precisão, a técnica *Bootstrap* pode ser facilmente paralelizada a fim de reduzir o tempo de simulação, e consequentemente diminuir o seu custo computacional. Um estudo mais detalhado em relação ao tempo de processamento das técnicas de simulações empregadas neste artigo podem ser encontrados em [Taschetto 2010].

Uma das motivações para este trabalho é que métodos numéricos iterativos (como por exemplo, o método da Potência) possui uma solução aproximada, porém próxima da exata. Entretanto, os limites computacionais não permitem a solução para modelos de

larga escala. Então, uma solução numérica alternativa é a simulação destes modelos. Visto que a simulação utiliza um gerador pseudo-aleatório para determinar as transições em uma trajetória de maneira probabilística, técnicas estatísticas (como por exemplo, *Bootstrap*) podem ser empregadas de maneira eficiente, atenuando os "ruídos" eventuais do gerador.

A simulação de modelos Markovianos é um eixo muito importante no contexto de avaliação de desempenho de sistemas, pois possibilita a obtenção de índices de desempenho ou previsões de comportamentos em modelos cada vez mais complexos (*i.e.*, com grandes espaços de estados). Modelos descritos através do formalismo SAN destacam-se no contexto de modelagem de sistemas, pois este formalismo permite um grande poder de abstração de forma compacta e modular, inclusive para interações complexas entre os componentes do sistema, *i.e.*, interações representadas na forma de transições funcionais, onde o disparo destas transições depende da avaliação de uma função.

O estudo apresentado neste artigo permite a proposta de trabalhos futuros voltados para a solução de modelos Markovianos através de simulação, tais como: (i) *impacto do número e do tamanho dos bootstraps* - ainda é necessário um estudo mais aprofundado para relacionar o impacto do número de bootstraps (z), bem como o valor de reamostragens ($\bar{n}$) a serem efetuadas em cada *bootstrap*, no custo computacional e na precisão dos resultados; (ii) *categorização de modelos* - modelos com taxas que determinam transições muito raras podem apresentar comportamentos diferentes e influenciar na precisão dos resultados. Como pode ser observado, o modelo RS, ao contrário dos demais utilizados neste trabalho, não demonstrou diferenças significativas na precisão utilizando as duas técnicas. Dito isso, identificar possíveis classes de modelos para aplicar diferentes técnicas de simulação, ou até mesmo propor uma nova técnica que apresente melhor eficiência em termos de precisão, é um trabalho futuro a ser realizado; (iii) *paralelização da simulação Bootstrap* - visto que a tiragem dos valores pseudo-aleatórios para o *bootstrap* é totalmente independente da tiragem para outro *bootstrap*, esta tarefa poderia perfeitamente ser paralelizada a fim de reduzir o custo computacional da técnica.

Referências

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

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

Bauer, E. and Kohavi, R. (1999). An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants. *Machine Learning*, 36(1-2):105–139.

Bertolini, C., Brenner, L., Fernandes, P., Sales, A., and Zorzo, A. F. (2004). Structured Stochastic Modeling of Fault-Tolerant Systems. In 12^{th} *IEEE/ACM Internacional Symposium on Modelling, Analysis and Simulation on Computer and Telecommunication Systems (MASCOTS'04)*, pages 139–146, Volendam, The Netherlands. IEEE Press.

Brenner, L., Fernandes, P., Fourneau, J. M., and Plateau, B. (2009). Modelling Grid5000 point availability with SAN. *Elect. Notes in Theoret. Computer Science (ENTCS)*, 232:165–178.

Brenner, L., Fernandes, P., Plateau, B., and Sbeity, I. (2007). PEPS2007 - Stochastic Automata Networks Software Tool. In *Fourth Internat. Conf. on the Quantitative Evaluation of Systems (QEST'07)*, pages 163–164, Edinburgh, UK. IEEE Press.

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

Dotti, F. L., Fernandes, P., Sales, A., and Santos, O. M. (2005). 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 (WiOpt'05)*, pages 164–173, Trentino, Italy. IEEE Computer Society.

Efron, B. (1979). Bootstrap Methods: Another Look at the Jackknife. *The Annals of Statistics*, 7(1):1–26.

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

Häggström, O. (2002). *Finite Markov Chains and Algorithmic Applications*. Cambridge University Press. Based on lecture notes.

Hillston, J. (1996). *A compositional approach to performance modelling*. Cambridge University Press, New York, USA.

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

Plateau, B. (1985). On the stochastic structure of parallelism and synchronization models for distributed algorithms. In *Proc. of the 1985 ACM SIGMETRICS Conf. on Measurements and Modeling of Comp. Systems*, pages 147–154, Austin, Texas. ACM Press.

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

Ross, S. M. (2002). *Simulation*. Academic Press, Inc., Orlando, FL, USA.

Saad, Y. and Schultz, M. H. (1986). GMRES: A Generalized Minimal RESidual Algorithm for Solving Nonsymmetric Linear Systems. *SIAM Journal on Scientific and Statistical Computing*, 7(3):856–869.

Sales, A. and Plateau, B. (2009). Reachable state space generation for structured models which use functional transitions. In *Proc. of the* 6^{th} *Internat. Conf. on the Quantitative Evaluation of Systems (QEST'09)*, pages 269–278, Budapest, Hungary. IEEE Press.

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

Shannon, R. E. (1998). Introduction to the art and science of simulation. In *Proceedings of the* 30^{th} *Conference on Winter Simulation (WSC'98)*, pages 7–14, Los Alamitos, CA, USA. IEEE Computer Society Press.

Stewart, W. J. (2009). *Probability, Markov Chains, Queues, and Simulation*. Princeton University Press, Princeton, NJ, USA.

Taschetto, D. (2010). Precisão de Simulações para Solução de Modelos Estocásticos. Master's thesis, PUCRS-FACIN-PPGCC, Porto Alegre.