Cálculo Eficiente da Distribuição do Tamanho de Filas baseado em Modelos de Fluido *

Ana Paula Couto da Silva†,Rosa Maria Meri Leão,Edmundo de Souza e Silva
{anapaula,rosam,edmundo}@land.ufrj.br
Universidade Federal do Rio de Janeiro
COPPE - Programa de Engenharia de Sistemas e Computação
IM - Departamento de Ciência da Computação
Caixa Postal 68.511, Rio de Janeiro, RJ, 21945-970

Resumo

O estudo do comportamento de redes quando estas transportam tráfego gerado por aplicações multimídia tem sido objeto de diversos estudos na literatura. Modelos Markovianos com recompensa têm sido usados para o estudo de redes multimídia. Nestes modelos recompensas de taxas são associadas aos estados e/ou recompensas de impulso são associadas às transições. Desenvolvemos um algoritmo que calcula a distribuição do valor acumulado por um fluido em um reservatório limitado possibilitando a modelagem de um *buffer* finito alimentado por uma fonte de fluido. O algoritmo é baseado em modelos com recompensas de impulso. A matriz gerada a partir do modelo proposto possui uma estrutura especial que é a chave da nossa técnica. O algoritmo possui vantagens computacionais quando comparado com outros métodos encontrados na literatura.

Palavras chaves: avaliação de desempenho, modelos de fluido, análise estacionária, modelos com recompensa, distribuição do tamanho da fila.

Abstract

Multimedia networks have been extensively studied in the literature. Markovian reward models have been used to model and to obtain performance measures from multimedia networks. In these models, reward rates are associated with states and/or reward impulses are attributed to transitions between states. We have developed an algorithm to obtain the queue size distribution of a finite *buffer* when it is fed by a fluid. We base our analysis on impulse reward models. The probability transition matrix generated from the model we obtained has a special structure that is the key to our technique. The algorithm presents computational advantages when compared with others approaches in the literature.

*Este trabalho é parcialmente financiado pelo CNPq/ProTeM, PRONEX e FAPERJ.
† Ana Paula teve bolsa de mestrado do CNPq.

1 Introdução

A modelagem de sistemas de comunicação/computação é importante para o dimensionamento de recursos e previsão do comportamento de algoritmos usados no gerenciamento destes sistemas. Desta forma, após a determinação de algumas medidas de interesse, podemos observar como o sistema reage a certas variações de parâmetros bem como de características inerentes à sua natureza. Tomemos, por exemplo, a obtenção da função distribuição da ocupação de um *buffer* alimentado por uma fonte de tráfego. Esta medida de interesse é importante para o dimensionamento de redes de comunicação e para o estudo da qualidade de serviço das aplicações.

Com o advento das redes de alta velocidade, as aplicações multimídia se tornaram alvo de estudo. A modelagem do tráfego gerado por estas aplicações têm sido objeto de diversos trabalhos na literatura [1, 2]. Uma das abordagens amplamente encontrada na literatura é a de modelos baseados em fluidos (*fluid flow models*) [3, 4]. Podemos observar que o tipo de tráfego gerado pelas aplicações multimídia se caracteriza por uma grande quantidade de informação, uma grande variabilidade na taxa de transmissão dos dados e que o tempo entre chegadas é pequeno em relação à duração de uma rajada de informações. Na modelagem de fluido, o fluxo de chegada e o fluxo de saída de pacotes em um *buffer* são aproximados por um fluido contínuo de informação, representando de forma adequada as características do tráfego gerado por essas aplicações. Métodos que utilizam equações diferenciais e transformada de Laplace [5, 4, 6] têm sido utilizados para obtenção de medidas de interesse a partir de modelos de fluido. Outros métodos se utilizam de técnicas como a uniformização para a solução de modelos Markovianos com recompensa (que podem ser considerados modelos de fluido). Esses últimos, em geral, são matematicamente mais simples e se mostram extremamente eficientes.

Diversos algoritmos baseados na utilização de recompensas, têm sido propostos na literatura. Alguns desses algoritmos possibilitam a obtenção de medidas de interesse em estado transiente [7, 8, 9, 10]. No entanto, soluções em estado estacionário de modelos onde o fluido acumulado é limitado superiormente e inferiormente são raras e baseadas em equações diferenciais parciais.

O objetivo deste trabalho é apresentar um algoritmo que, através do paradigma de recompensa, visa obter a distribuição em estado estacionário do valor acumulado por um fluido em um reservatório limitado, possibilitando a modelagem de um *buffer* finito alimentado por uma fonte de fluido. A matriz gerada a partir do modelo proposto é a chave da nossa técnica. Não é de nosso conhecimento na literatura um outro algoritmo que permita a obtenção da medida descrita acima baseado em modelos com recompensa. É importante ressaltar que podemos encontrar na literatura diversos métodos para obtenção da distribuição da ocupação de um *buffer* em regime estacionário [11, 12, 13]. Contudo, estes algoritmos são usados na solução de modelos onde o *buffer* é explicitamente representado, o que por vezes torna o custo computacional bastante alto.

Um estudo do custo computacional do algoritmo proposto mostrou que o mesmo apresenta ganhos significativos quando comparado com outros algoritmos da literatura usados para a solução de modelos semelhantes. Uma outra consideração importante é que a aplicação deste algoritmo não se restringe somente ao modelo apresentado neste trabalho onde são representadas fontes de tráfego que alimentam um *buffer*. Ele pode ser utilizado para a obtenção da solução de outras classes de modelos que possuam a matriz de probabilidade de transição com uma determinada estrutura especial.

Este trabalho está organizado da forma descrita a seguir. Na seção 2 são introduzidos

alguns conceitos básicos relacionados a modelos com recompensa e a modelagem proposta é apresentada. O algoritmo para o cálculo da distribuição em estado estacionário do valor acumulado por um fluido em um reservatório limitado é descrito na seção 3. Na seção 4 são apresentados alguns exemplos que mostram a utilidade do trabalho na modelagem de redes multimídia e a seção 5 apresenta algumas conclusões deste trabalho.

2 Definição do Modelo

Nesta seção iremos primeiramente introduzir alguns conceitos básicos relacionados a modelos com recompensa. Em seguida iremos apresentar a modelagem proposta no nosso trabalho, onde recompensas de impulsos são utilizadas para o cálculo da distribuição em estado estacionário do valor acumulado por um fluido em um reservatório limitado.

2.1 Modelos com Recompensa

Modelos markovianos com recompensa são aqueles em que (a) o processo ganha uma recompensa quando transiciona de um estado para outro (recompensa de impulso) e/ou (b) ganha uma recompensa a cada unidade de tempo em que permanece em um determinado estado (recompensa de taxa).

Iremos descrever a seguir como o modelo com recompensas pode ser usado para representar uma fonte que gera um fluido que é armazenado em um reservatório quando a taxa de geração é maior que a taxa de consumo do fluido. Considere $X = \{X(t), t \geq 0\}$ o processo que modela uma fonte que alimenta um canal de comunicação, C a capacidade do canal e λ_i (*bits*, células, pacotes) a taxa de transmissão (constante) no estado s_i. Seja $r_i = \lambda_i - C$ a recompensa de taxa atribuída ao estado s_i. Neste modelo, se o processo permanece no estado s_i t unidades de tempo, o reservatório aumenta de $r_i t$ unidades de informação, caso r_i seja positiva. Caso r_i seja negativa, o reservatório decresce de $r_i t$ unidades de informação. A recompensa de taxa acumulada até o instante t, $CR(t)$, é igual ao tamanho do reservatório no instante t desde que $CR(t)$ esteja limitada entre zero e o tamanho máximo do reservatório [10]. Podemos ver que neste modelo estamos representando tanto o fluxo de chegada de informação quanto o fluxo de saída de informação do reservatório como um fluido contínuo, semelhante à aproximação que é utilizada nos modelos de fluido.

Definimos como modelos Markovianos com recompensas de impulso os modelos onde a uma dada transição particular ocorrida de um estado s_j para um estado s_i, uma recompensa $\rho(s_j, s_i)$ é atribuída ao processo. Como a recompensa de impulso está dissociada do tempo, tradicionalmente esta é utilizada para contabilizar eventos que ocorrem durante a evolução do processo, já que são os eventos que desencadeiam as transições.

Na seção seguinte apresentamos um modelo baseado em recompensas de impulsos para contabilizar o acréscimo ou decréscimo de um fluido em um reservatório. Neste modelo iremos limitar as recompensas de impulso inferiormente e superiormente.

2.2 Modelagem Proposta

Considere o processo $M = \{M(t), t \geq 0\}$ que representa uma fonte de tráfego conforme apresentado na Figura 1. Neste modelo λ_j representa a taxa de envio de informações associada ao estado s_j, C é a capacidade do canal (igual a 15 unidades de informação/s) e α_{ji} a taxa de

transição entre os estados. Um reservatório de capacidade limitada armazena as informações que ultrapassam a capacidade de transmissão em algum momento. Associamos uma recompensa de impulso $\rho(s_j, s_i)$ a uma transição do estado s_j para o estado s_i, sendo que esta deve ser igual a variação do reservatório em um intervalo de tempo t.

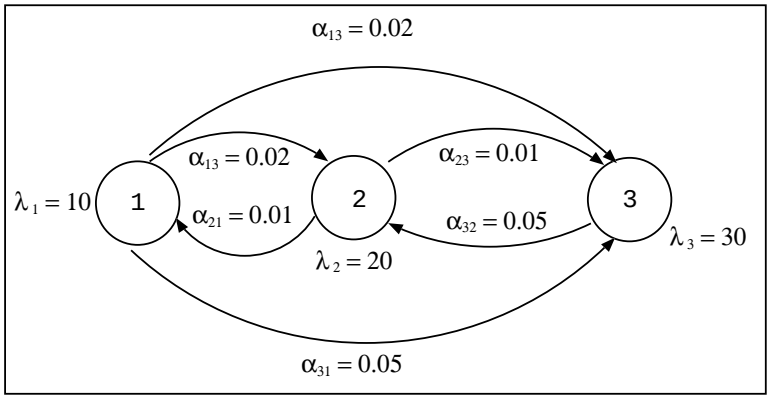


Figura 1: Modelo de uma fonte de tráfego.

Suponha que M tenha sido uniformizado com taxa Λ [14]. Logo, o tempo de permanência em cada estado do modelo é exponencialmente distribuído com média $1/\Lambda$. Dessa forma, podemos dizer que o tamanho do reservatório está aumentando em média de $(\lambda_j - C)/\Lambda$ (se $(\lambda_j - C) > 0$) ou diminuindo em média de $(\lambda_j - C)/\Lambda$ (se $(\lambda_j - C) < 0$), a cada visita ao estado s_j.

Nosso objetivo é definir um modelo onde, cada vez que um estado seja visitado, o reservatório acumule ou perca a mesma quantidade de informação. Desta forma estaremos limitando as transições entre os estados do modelo o que dará origem a uma matriz de probabilidade de transição (matriz **P**) com uma certa estrutura especial que será útil para obter uma solução eficiente. Para isso definimos uma taxa Λ_j para cada estado tal que:

$$\frac{|\ \lambda_j - C\ |}{\Lambda_j} = K$$

onde K é uma constante igual a quantidade de informação que será acrescida/retirada do reservatório cada vez que o processo fizer uma transição a partir do estado s_j. Associaremos à transição do estado s_j para o estado s_i uma recompensa de impulso $\rho(s_j, s_i) = (\lambda_j - C)/\Lambda_j$, caso o reservatório não tenha alcançado seu limite inferior nem o seu limite superior. Neste modelo a distribuição da recompensa de impulso acumulada em estado estacionário é igual a distribuição do valor acumulado por um fluido em um reservatório limitado que é a medida que pretendemos obter.

Conforme poderemos observar na seção 4, o parâmetro K pode assumir valores expressivos em relação ao tamanho do reservatório, o que resulta em economia computacional na solução do modelo, sem afetar, no entanto, a precisão do método.

Definiremos o estado do modelo como o par ordenado $S_i^j = (i, j)$, onde j é o estado da fonte e i o nível de recompensa acumulada no reservatório. Considerando para o modelo da Figura 1 um reservatório igual a 3 e $K = 1$ (significando que o tamanho do reservatório aumenta uma unidade quando $\rho(s_i^j, s_n^m) > 0$, ou que o tamanho do reservatório diminui uma unidade quando $\rho(s_i^j, s_n^m) < 0$), temos o seguinte espaço de estados $S = \{(0,1), (0,2), (0,3), (1,1),$ $(1,2), (1,3), (2,1), (2,2), (2,3), (3,1), (3,2), (3,3)\}$. A Figura 2 apresenta a matriz **P** do mode-

lo da Figura 1. É importante ressaltar que a unidade usada para o aumento/diminuição do fluido no reservatório não corresponde a um pacote de dados como é usual.

	S_0^1	S_0^2	S_0^3	S_1^1	S_1^2	S_1^3	S_2^1	S_2^2	S_2^3	S_3^1	S_3^2	S_3^3
S_0^1	0.992	0.004	0.004	0	0	0	0	0	0	0	0	0
S_0^2	0	0	0	0.002	0.996	0.002	0	0	0	0	0	0
S_0^3	0	0	0	0.003	0.003	0.994	0	0	0	0	0	0
S_1^1	0.992	0.004	0.004	0	0	0	0	0	0	0	0	0
S_1^2	0	0	0	0	0	0	0.002	0.996	0.002	0	0	0
S_1^3	0	0	0	0	0	0	0.003	0.003	0.994	0	0	0
S_2^1	0	0	0	0.992	0.004	0.004	0	0	0	0	0	0
S_2^2	0	0	0	0	0	0	0	0	0	0.002	0.996	0.002
S_2^3	0	0	0	0	0	0	0	0	0	0.003	0.003	0.994
S_3^1	0	0	0	0	0	0	0.992	0.004	0.004	0	0	0
S_3^2	0	0	0	0	0	0	0	0	0	0.002	0.996	0.002
S_3^3	0	0	0	0	0	0	0	0	0	0.003	0.003	0.996

Figura 2: Matriz de probabilidade de transições do modelo da Figura 1.

Generalizando , as probabilidades de transição entre os estados do modelo e os impulsos associados às transições são obtidos da forma descrita abaixo.

Seja $i = \{0, 1, 2, ..., M\}$, $n = \{0, 1, 2, ..., M\}$, $j = \{1, ..., N\}$ e $m = \{1, ..., N\}$, onde N é o número total de estados da fonte e M é o valor máximo da recompensa acumulada no reservatório.

1. Para $\lambda_j - C < 0$ e $i - n = K$ ou $\lambda_j - C > 0$ e $n - i = K$:

$$p_{s_i^j, s_n^m} = \frac{\alpha_{jm}}{\Lambda_j} \qquad \text{se j} \neq \text{m}$$

$$p_{s_i^j, s_n^m} = 1 - \sum_{\forall m m \neq j} p_{s_i^j, s_n^m} \quad \text{se j = m}$$

$$\rho(s_i^j, s_n^m) = \frac{\lambda_j - C}{\Lambda_j}$$

2. Para $\lambda_j - C < 0$ e $i = n = 0$ ou $\lambda_j - C > 0$ e $i = n = M$

$$p_{s_i^j, s_n^m} = \frac{\alpha_{jm}}{\Lambda_j} \qquad \text{se j} \neq \text{m}$$

$$p_{s_i^j, s_n^m} = 1 - \sum_{\forall m, m \neq j} p_{s_i^j, s_n^m} \quad \text{se j = m}$$

$$\rho(s_i^j, s_n^m) = 0$$

3. Para todos os casos em que as condições 1 ou 2 não são satisfeitas:

$$p_{s_i^j, s_n^m} = 0$$

Neste método sendo proposto cada estado é uniformizado com uma taxa adequada para o estado. Sendo assim o processo passa em cada estado do modelo representado acima um tempo médio distinto do outro. Desta forma, a matriz **P** pode ser representada como **P = I +** Λ_a**Q**, onde a matriz Λ_a é uma matriz diagonal com os tempos médios de permanência em cada estado do processo.

As vantagens da utilização do modelo com recompensas de impulso estão relacionadas a redução do espaço de estados do modelo em relação àquele comumente usado para representar armazenamento de pacotes e a estrutura particular da matriz **P** obtida. Esta estrutura permite a aplicação de um novo algoritmo para a obtenção da distribuição da recompensa de impulso acumulada que é descrito na seção seguinte.

3 Algoritmo Proposto

A medida de interesse a ser calculada é a distribuição em estado estacionário da recompensa de impulso acumulada para o modelo definido na seção 2.2 que possui estrutura especial da matriz **P**. Desta forma, estaremos obtendo a distribuição do valor acumulado por um fluido em um reservatório limitado.

O algoritmo proposto nesta seção resolve o sistema de equações $\pi P = \pi$. Este algoritmo baseia-se no trabalho apresentado em [11] e maiores detalhes podem ser encontrados em [15].

Iremos considerar os estados ordenados da seguinte forma: fixamos o nível de recompensa acumulada e variamos os estados da fonte, iniciando com os estados da fonte onde $\lambda_j - C < 0$ (conforme Figura 2). A matriz **P** apresentada na Figura 2 pode ser organizada em blocos conforme a Figura 3.

Analisando a divisão em blocos da matriz **P** apresentada, as seguintes observações devem ser feitas:

1. Os blocos $\mathbf{B}_0$ e $\mathbf{B}_1$ representam transições que partem de estados onde $\lambda_j - C < 0$ e o tamanho do reservatório é igual ao limite inferior. Logo não pode ocorrer alteração no tamanho do reservatório nas transições partindo desses estados (a recompensa associada às transições entre estados pertencentes a estes blocos é zero).

2. Os blocos $\mathbf{C}_0$ e $\mathbf{C}_1$ representam transições que partem de estados onde $\lambda_j - C > 0$ e o tamanho do reservatório é igual ao limite superior. Logo não pode ocorrer alteração no tamanho do reservatório nas transições partindo desses estados (a recompensa associada às transições entre estados pertencentes a estes blocos é zero).

3. Os blocos $\mathbf{A}_0$, $\mathbf{A}_1$, $\mathbf{A}_2$ e $\mathbf{A}_3$ representam transições entre níveis vizinhos de recompensa acumulada. Nos blocos $\mathbf{A}_0$ e $\mathbf{A}_1$ parte-se de estados onde $\lambda_j - C > 0$, logo a recompensa associada às transições entre estados pertencentes a estes blocos tem valor K. Já nos blocos $\mathbf{A}_2$ e $\mathbf{A}_3$ parte-se de estados onde $\lambda_j - C < 0$, logo a recompensa associada às transições entre estados pertencentes a estes blocos tem valor $-K$ (Lembramos que $K = \mid \lambda_j - C \mid / \Lambda_j$.)

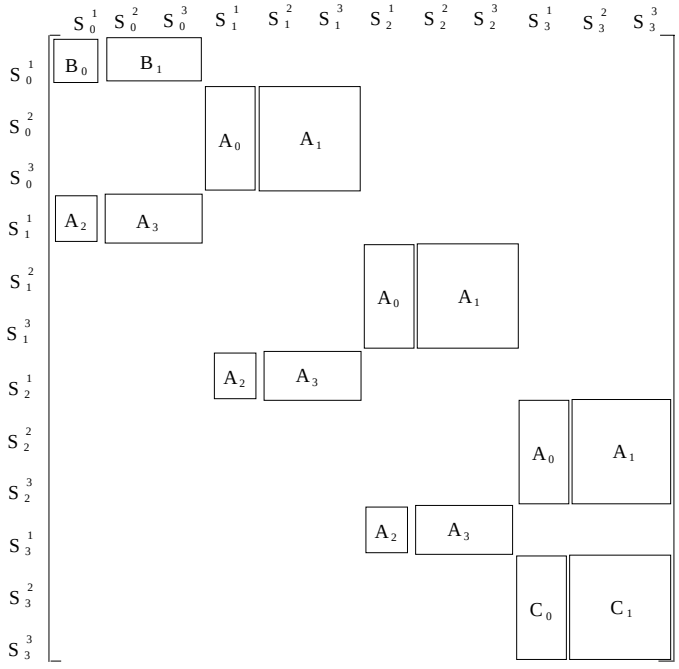


Figura 3: Matriz de probabilidades **P** particionada em blocos.

4. Os blocos diagonais são todos nulos, excetuando o primeiro e o último blocos. Este fato ocorre devido as características do modelo especificado, onde a toda transição do processo é associada uma recompensa. Dessa maneira, nunca é possível, exceto para os casos de contorno (limites inferior e superior), que se transicione para o mesmo estado da fonte sem a mudança do nível de recompensa acumulada.

Considere o valor máximo da recompensa acumulada igual a M como definido anteriormente. Temos então que o número total de níveis de recompensa acumulada é $L = M/K$. A partir da definição acima, temos que a matriz **P** particionada em blocos (como no exemplo da Figura 3) possui $2(L+1)$ blocos. Suponha esta mesma partição em blocos para o vetor π. Para cada bloco deste vetor, π_i^j, definiremos $j = 0$ para os estados onde $\lambda_j - C < 0$ e $j = 1$ para os estados onde $\lambda_j - C > 0$. O valor de i representa o nível de recompensa acumulada ($i = \{0, 1, 2, ..., L\}$).

Como característica deste novo método, a cada passo da redução da matriz **P** estaremos eliminando duas partições do vetor de probabilidades π, ou seja, dois subconjuntos de estados. Ao final da redução, todas as equações estarão em função de π_0^0.

A Figura 4 apresenta os blocos que devem ser eliminados a cada passo do algoritmo, bem como o vetor π devidamente particionado.

Consideremos as duas primeiras equações obtidas a partir das duas primeiras colunas da Figura 4:

$$\pi_0^0 \mathbf{B}_0 + \pi_1^0 \mathbf{A}_2 = \pi_0^0 \tag{1}$$

$$\pi_0^0 \mathbf{B}_1 + \pi_1^0 \mathbf{A}_3 = \pi_0^1 \tag{2}$$

Resolvendo a equação (1) para π_1^0 e a equação (2) para π_0^1, teremos:

$$\pi_1^0 = \pi_0^0 (\mathbf{I} - \mathbf{B}_0)(\mathbf{A}_2)^{-1} \tag{3}$$

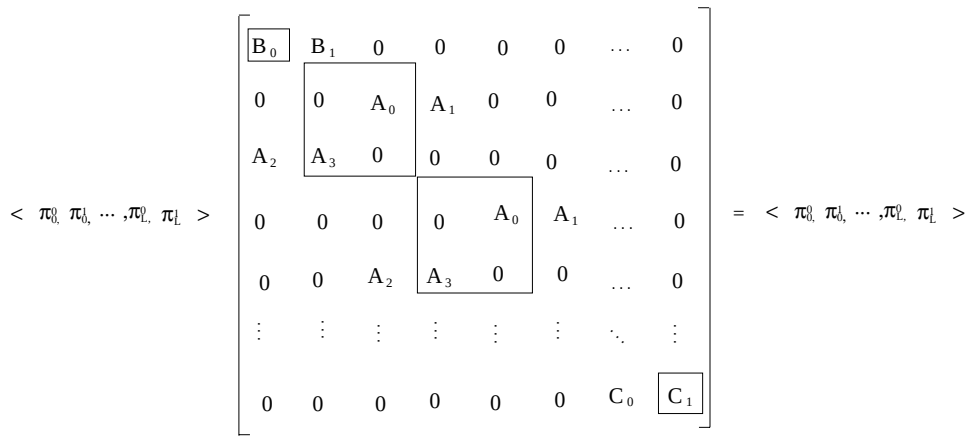


Figura 4: Blocos que devem ser eliminados a cada passo do algoritmo.

$$\pi_0^1 = \pi_0^0[\mathbf{B}_1 + (\mathbf{I} - \mathbf{B}_0)(\mathbf{A}_2)^{-1}\mathbf{A}_3] \tag{4}$$

O próximo passo é a eliminação de π_1^0 e π_0^1 através da substituição em todas as demais equações, obtendo o seguinte sistema reduzido:

$$\pi^{(1)}\mathbf{P}^{(1)} = \pi^{(1)}$$

onde $\pi^{(1)}$ não possui as componentes π_1^0 e π_0^1 e a matriz $\mathbf{P}^{(1)}$ possui a seguinte estrutura:

$$\mathbf{P}^{(1)} = \begin{bmatrix} \mathbf{B}_0^{(1)} & \mathbf{B}_1^{(1)} & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & \mathbf{A}_0 & \mathbf{A}_1 & 0 & 0 & \cdots & 0 \\ \mathbf{A}_2^{(1)} & \mathbf{A}_3 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & \mathbf{A}_0 & \mathbf{A}_1 & \cdots & 0 \\ 0 & 0 & \mathbf{A}_2 & \mathbf{A}_3 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \cdots & \mathbf{C}_0 & \mathbf{C}_1 \end{bmatrix}$$

sendo:

$$\mathbf{B}_0^{(1)} = [\mathbf{B}_1^{(0)} + (\mathbf{I} - \mathbf{B}_0^{(0)})(\mathbf{A}_2^{(0)})^{-1}\mathbf{A}_3]\mathbf{A}_0\mathbf{A}_2^{(0)} + \mathbf{B}_0^{(0)}$$

$$\mathbf{B}_1^{(1)} = [\mathbf{B}_1^{(0)} + (\mathbf{I} - \mathbf{B}_0^{(0)})(\mathbf{A}_2^{(0)})^{-1}\mathbf{A}_3]\mathbf{A}_1$$

$$\mathbf{A}_2^{(1)} = (\mathbf{A}_2^{(0)})^2$$

O processo de redução da matriz **P** passa a ser aplicado recursivamente, até que o seguinte conjunto de equações seja obtido:

$$\pi_0^0(\mathbf{B}_0^{(L)} + \mathbf{B}_1^{(L)}(\mathbf{I} - \mathbf{C}_1)^{-1}\mathbf{C}_0) = \pi_0^0 \tag{5}$$

A partir do valor de π_0^0 as demais partições do vetor π são calculadas.

O algoritmo pode ser descrito da seguinte forma:

1. Faça $k = 0$ e inicialize a matriz: $\mathbf{P}^{(k)} = \mathbf{P}$.

2. Compute a matriz $\mathbf{P}^{(k+1)}$ a partir de $\mathbf{P}^{(k)}$ da seguinte forma:

$$\mathbf{B}_0^{(k+1)} = [\mathbf{B}_1^{(k)} + (\mathbf{I} - \mathbf{B}_0^{(k)})(\mathbf{A}_2^{(k)})^{-1}\mathbf{A}_3]\mathbf{A}_0\mathbf{A}_2^{(k)} + \mathbf{B}_0^{(k)}$$
$$\mathbf{B}_1^{(k+1)} = [\mathbf{B}_1^{(k)} + (\mathbf{I} - \mathbf{B}_0^{(k)})((\mathbf{A}_2)^{(k)})^{-1}\mathbf{A}_3]\mathbf{A}_1$$
$$\mathbf{A}_2^{(k+1)} = [\mathbf{A}_2^{(0)}]^{k+1}$$

3. Incremente k e, se $k < L$, vá para 2.

4. Calcule π_0^0 a partir de:

$$\pi_0^0(\mathbf{B}_0^{(L)} + \mathbf{B}_1^{(L)}(\mathbf{I} - \mathbf{C}_1)^{-1}\mathbf{C}_0) = \pi_0^0$$

5. Calcule π_i^0, para i = $\{1, \cdots, L\}$

$$\pi_i^0 = \pi_0^0(\mathbf{I} - \mathbf{B}_0^{(i-1)})((\mathbf{A}_2)^{(i-1)})^{-1}$$

6. Calcule π_i^1, para i = $\{0, \cdots, L-1\}$

$$\pi_i^1 = \pi_0^0[\mathbf{B}_1^{(i)} + (\mathbf{I} - \mathbf{B}_0^{(i)})(\mathbf{A}_2^{(i)})^{-1}\mathbf{A}_3]$$

7. Calcule π_L^1, a partir da seguinte equação:

$$\pi_L^1 = \pi_0^0\mathbf{B}_1^{(L)}(\mathbf{I} - \mathbf{C}_1)^{-1}$$

8. Normalize o vetor π.

No algoritmo apresentado, devemos calcular a inversa da matriz $(\mathbf{I} - \mathbf{C}_1)$ e a inversa da matriz $\mathbf{A}_2^{(0)}$. A prova de que a inversa dessas matrizes sempre existe pode ser encontrada em [15].

Como descrito na seção 2, o processo passa um tempo médio $1/\Lambda_j$ quando este visita um determinado estado S_i^j, logo para obtermos os valores corretos para as probabilidades dos estados S_i^j, denominadas de $\pi_i^{j'}$, faremos [16]:

$$\pi_{\mathbf{i}}^{\mathbf{j}'} = \frac{\pi_i^j(1/\Lambda_j)}{\sum_k \pi_k(1/\Lambda_k)}$$

O custo do algoritmo apresentado está relacionado com o número total de passos para reduzir a matriz **P**, sendo que este número é igual ao total de níveis de recompensa acumulada, ou seja L. Além disso, cada bloco da matriz **P** é de ordem B x B, onde B é proporcional ao número de estados da fonte. O custo pode ser expresso através da seguinte expressão [15]:

$$T(L) = (7L + 7/3)B^3 + (2L)B^2 + C(\pi_0)$$

sendo $C(\pi_0)$ o custo para resolução do sistema de equações (5).

Dois fatores contribuem para a diminuição do custo computacional do algoritmo. O primeiro é o fato de termos um termo que se repete no cálculo de $\mathbf{B}_0^{(k+1)}$ e $\mathbf{B}_1^{(k+1)}$ (ver passo 2 do algoritmo). O segundo é que somente uma inversão de matriz de tamanho M x M é realizada. As demais inversões, relacionadas com o termo $(\mathbf{A}_2^{(k)})^{-1}$ não são realizadas, devido ao uso da seguinte propriedade matricial: se **A** e **B** são matrizes quadradas de mesma ordem, ambas inversíveis, então **AB** é inversível e $(\mathbf{AB})^{-1} = \mathbf{B}^{-1}\mathbf{A}^{-1}$ [17].

4 Aplicações

Nesta seção apresentaremos dois exemplos de modelos de fontes multimídia alimentando um *buffer* cujos objetivos são: (1) avaliar a precisão obtida com o algoritmo e (2) avaliar o custo computacional e os resultados obtidos em relação a outros algoritmos semelhantes da literatura.

No primeiro exemplo temos uma fonte *On-Off* alimentando um *buffer*. Esta fonte foi escolhida pois possui as características adequadas para que seja usado um modelo de fluido, ou seja, taxas de envio de dados e de serviço altas quando comparadas à duração de uma rajada. O principal objetivo deste primeiro exemplo é mostrar a precisão obtida e o custo computacional do novo método quando variamos o valor de K, lembrando que K é a quantidade de informação que é acrescida/decrescida da fila do *buffer* a cada transição da cadeia uniformizada.

O segundo exemplo é o modelo de uma fonte de tráfego que representa uma seqüência de vídeo alimentando um *buffer*. O principal objetivo deste segundo exemplo é realizar um estudo comparativo do custo computacional e dos resultados obtidos com relação a outros métodos da literatura.

Para avaliarmos a precisão obtida teríamos que implementar métodos exatos de solução de modelos de fluido em estado estacionário. Estes métodos, em geral, são baseados na solução de equações diferenciais e são bastante complexos. Considerando o que foi mencionado acima e que encontra-se implementado na ferramenta TANGRAM-II um algoritmo exato [10] que calcula a distribuição complementar do tamanho do *buffer* no instante t, utilizamos este último para o cálculo da medida exata para um tempo t de observação bastante grande.

Escolhemos dois métodos na literatura para compararmos o custo computacional do método proposto: GTH [18] e o método de [11]. O primeiro método é o proposto por Grassman, Taksar e Heyman, conhecido como GTH ou método de eliminação de estados. O segundo é o método proposto por Meo, de Souza e Silva e Ajmone Marsan apropriado para uma classe de modelos cuja matriz de transição de estados possui uma estrutura especial comum em modelos de sistemas de telecomunicações. A escolha destes dois métodos se deve a serem eficientes (baixo custo computacional) para a classe de modelos em estudo.

Os dois exemplos aqui apresentados foram elaborados com a ferramenta de modelagem TANGRAM-II [19, 20], desenvolvida pelo Laboratório de Análise e Desempenho de Sistemas de Computação e Comunicação da Universidade Federal do Rio de Janeiro (LAND-UFRJ), disponível para *download* em *www.land.ufrj.br*.

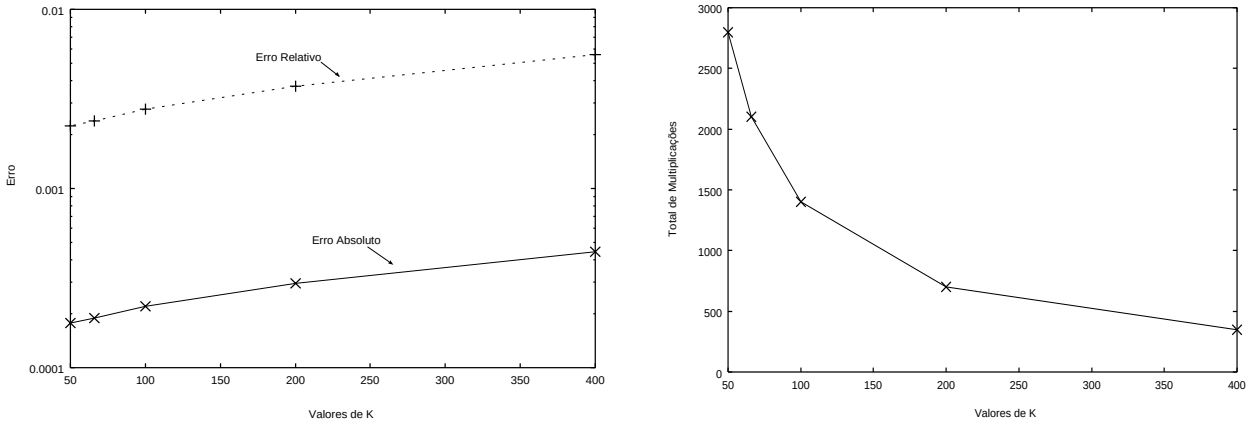
4.1 Fonte *On-Off*

Consideremos uma fonte *On-Off* que alimenta um *buffer* de capacidade igual a 20.000 unidades de informação. O tempo de permanência no estado *On* e no estado *Off* são exponencialmente distribuídos com média igual a $8ms$ e $16ms$, respectivamente. A taxa de transmissão da fonte no estado *On* é de $2.5Mbits/s$ e a capacidade do canal de comunicação é de $1.25Mbits/s$.

Consideramos K (quantidade de informação acrescida/decrescida da fila) variando de 50 até 400. A variação de K teve como objetivo estimar a sua influência no erro obtido com o método e estimar a diminuição do custo computacional com o aumento do valor de K. A medida utilizada para as estimativas acima foi a probabilidade do tamanho da fila ser igual a sua capacidade total.

O gráfico da figura 5.(a) apresenta os erros absoluto e relativo obtidos para diferentes valores de K. O erro absoluto é igual a $\mid medida\_exata - medida\_metodo \mid$ e o erro relativo é dado por $\mid medida\_exata - medida\_metodo \mid / medida\_exata$. Conforme podemos observar, o erro absoluto se encontra na quarta casa decimal para todos os valores de K. Já o erro relativo é da ordem de 10^{-3}, o que mostra que a precisão obtida com o método é boa.

O custo computacional para os diversos valores de K pode ser visualizado na figura 5.(b). Podemos ver que o custo diminui com o aumento do valor de K. Neste exemplo, podemos obter redução de até uma ordem de grandeza no custo computacional, mantendo o erro relativo na mesma ordem de grandeza.



a) Erros Absoluto e Relativo **b)Custo Computacional**

Figura 5: Erro e custo computacional - Probabilidade do tamanho da fila ser igual a sua capacidade total.

Com o objetivo de comparar os resultados e custo obtidos com o método proposto em relação a outros métodos semelhantes da literatura, definimos um modelo onde todos eventos tem distribuição exponencial e obtivemos os resultados usando o método GTH. Para obtermos um modelo exponencial semelhante ao modelo de fluido considerado (onde $K = 100$ unidades são retiradas/acrescidas da fila) dividimos todas as taxas e o *buffer* do modelo exponencial por 100. A figura 6 mostra a distribuição complementar do tamanho da fila. Conforme podemos observar, o método proposto é uma boa aproximação do valor da medida exata e, para este exemplo, podemos notar que o modelo exponencial fornece resultados semelhantes aos obtidos com o modelo de fluido.

O custo computacional do método GTH é igual a 183.466 multiplicações, o do método de [11] é igual a 102.400 multiplicações e para método proposto obtivemos um custo de 96.149. Podemos observar a redução de até uma ordem de grandeza quando utilizamos o método proposto.

4.2 Seqüência de Vídeo

Neste segundo exemplo, a partir da seqüência *Asterix* disponível em domínio público [21] definimos um modelo de fonte do tipo histograma [22] com 8 níveis e a parametrização do modelo foi feita conforme sugerida em [22]. Os parâmetros em *bits/s* estão apresentados na Tabela 1. Consideraremos, originalmente, um *buffer* igual a 10.000 unidades de informação.

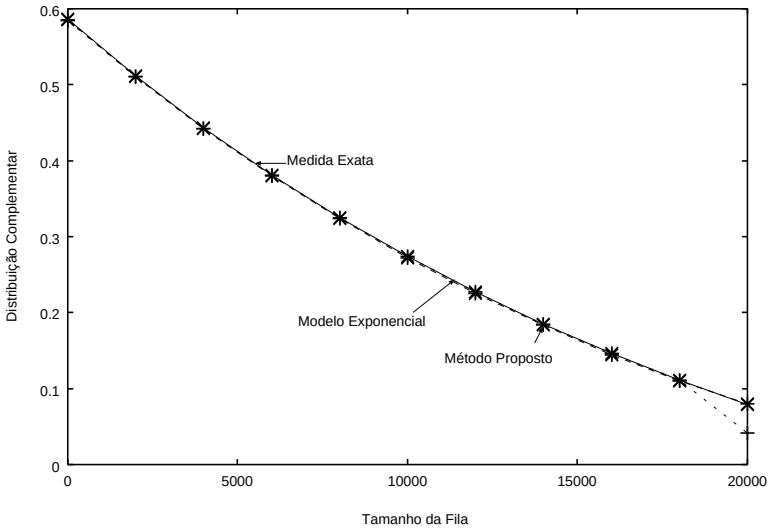


Figura 6: Distribuição complementar do tamanho da fila.

Estado Fonte	Taxa
1	9.496
2	27.880
3	46.264
4	64.648
5	83.032
6	101.416
7	119.800
8	138.184

Tabela 1: Parâmetros - Seqüência de Vídeo.

A Figura 7 mostra a distribuição complementar do tamanho da fila para $K = 200$. Fizemos a mesma aproximação que foi usada para o exemplo da fonte *On-Off*: dividimos todas as taxas e o *buffer* do modelo exponencial por 200. O modelo exponencial foi resolvido com o método GTH. Conforme podemos observar, o método proposto é uma boa aproximação do valor da medida exata. Já o modelo exponencial, no caso deste exemplo, apresenta resultados bastante diferentes dos obtidos com o modelo de fluido.

Para cálculo do custo computacional, consideramos K variando de 1 até 10. Usamos a versão em bloco para o algoritmo GTH, sendo o tamanho do bloco igual a 8. Identicamente os blocos de [11] tem tamanho 8. Para o método proposto o tamanho do bloco é igual a 4. A figura 8 mostra o custo computacional de cada um dos métodos para os diversos valores de K. Podemos notar que o custo do método proposto sempre se encontra abaixo do custo dos outros métodos apresentados. O fator principal, entre outros, que conduz a esse resultado está relacionado com a multiplicação de matrizes de ordem menor do que as matrizes do método GTH e do método [11].

De maneira geral, avaliando a expressão do custo computacional (para resolver a classe de modelos que estamos estudando) dos algoritmos GTH, do proposto em [11] e do proposto neste trabalho podemos estabelecer as condições em que o algoritmo proposto é mais eficiente.

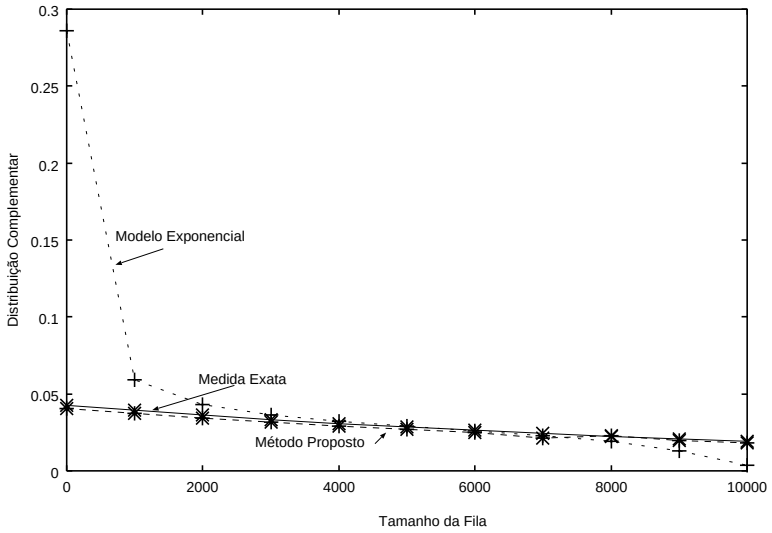


Figura 7: Distribuição complementar do tamanho da fila.

No caso do GTH, o algoritmo proposto é mais eficiente (menor custo computacional), sempre que o tamanho do bloco B for menor ou igual a 40. Já no caso do algoritmo de [11], o custo é menor para $B \leq 8$. Lembrando que B é proporcional ao número de estados da fonte. Maiores detalhes podem ser encontrados em [15].

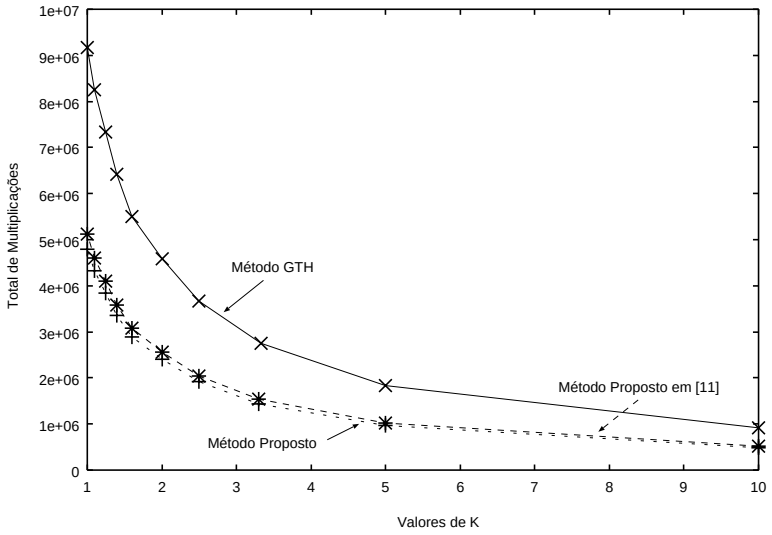


Figura 8: Custo computacional.

5 Conclusão

Neste trabalho apresentamos um algoritmo para o cálculo da distribuição da ocupação de um reservatório limitado quando este é alimentado por um fluido, possibilitando a modelagem de um *buffer* finito alimentado por uma fonte de fluido. Propusemos uma nova forma de modelar uma fonte alimentando um reservatório de fluido baseada em modelos com recompensa de impulso. O algoritmo proposto é baseado na eliminação de blocos e foi elaborado visando

fornecer uma solução eficiente para a classe de modelos que possui a matriz de probabilidade de transição com a estrutura do modelo proposto.

Dois exemplos foram apresentados visando mostrar a aplicabilidade do algoritmo proposto na obtenção de medidas de desempenho de redes multimídia. O primeiro exemplo foi de uma fonte *On-Off* e o segundo uma fonte histograma que modela um filme codificado em MPEG alimentando um *buffer*. Calculamos como medida de desempenho a distribuição complementar do tamanho da fila.

Através dos resultados dos exemplos podemos concluir que o algoritmo apresenta uma boa precisão mesmo quando a quantidade de fluido acrescida/retirada da fila é significativa em relação ao tamanho do *buffer*. Comparamos o custo computacional do algoritmo com o custo de outros métodos semelhantes na literatura conhecidos por serem eficientes (baixo custo computacional) para a classe de modelos em estudo. O algoritmo proposto apresentou menor custo computacional em todos os exemplos estudados. Mostramos também em que condições o algoritmo proposto é mais eficiente que o método GTH e o método de [11].

Referências

[1] A. Adas, "Traffic Models in Broadband Networks", *IEEE Communications Magazine*, no. 7, no. 7, pp. 82–89, 1997.

[2] K. Park e W. Willinger, eds., *Self-Similar Network Traffic and Performance Evaluation*. Jonh Wiley and Sons, 2000.

[3] A. Elwalid e D. Mitra, "Fluid Models for the Analysis and Design of Statistical Multiplexing with Loss Priorities on Multiple Classes of Bursty Traffic", in *Infocom-92*, pp. 415–425, 1992.

[4] D. Mitra, "Stochastic Theory of a Fluid Model of Producers and Consumers Coupled by a Buffer", *Adv. Appl. Prob*, vol. 20, pp. 646–676, 1988.

[5] D. Anick, D. Mitra e M. M. Sondhi, "Stochastic Theory of a Data-Handling System with Multiple Sources", *The Bell System Technical Journal*, vol. 61, no. 8, no. 8, pp. 1871–1894, 1982.

[6] Q. Ren e H. Kobayashi, "Transient Solutions for the buffer behavior in Statical Multiplexing", *Performance Evaluation*, no. 23, no. 23, pp. 65–87, 1995.

[7] M. C. Diniz, *Técnicas de Solução para Modelos Provenientes de Redes Multimidias*. PhD thesis, Programa de Pós-Graduação de Engenharia de Sistemas e Computação da COPPE/UFRJ, 2000.

[8] E. de Souza e Silva e H. Gail, "An Algorithm to Calculate Transient Distributions of Cumulative Rate and Impulse based Reward", *Communications in Statistics - Stochastic Models*, vol. 14, pp. 509–536, 1998.

[9] H. Nabli e B. Sericola, "Performability Analysis: a New Algorithm", *IEEE Trans. Computers*, vol. 45, pp. 491–494, 1996.

[10] C. E. F. de Brito, E. de Souza e Silva, M. C. Diniz e R. M. M. Leão, "Análise Transiente de Modelos de Fonte Multimídia", in $18^{\underline{th}}$ *Simpósio Brasileiro de Redes de Computadores*, pp. 519–534, Maio 2000.

[11] M.Meo, E. S. e Silva e M. A. Marsan, “Efficient Solution for a Class of Markov Chain Models of Telecommunication Systems”, *Performance Evaluation*, vol. 27&28, pp. 603–625, Outubro 1996.

[12] M. F. Neuts, *Matrix-Geometric Solutions in Stochastic Models*. Johns Hopkins University Press, Baltimore, 1981.

[13] W. K. Grassmann, ed., *Computational Probability*, ch. 6. Kluwer Academic Publishers, 2000.

[14] E. de Souza e Silva e H. R. Gail, *Performability Modeling*, ch. 3. Jonh Wiley & Sons, 2001.

[15] A. P. C. da Silva, “Métodos de Solução para Modelos Markovianos com Recompensa”, Tese de Mestrado, Programa de Pós-Graduação de Engenharia de Sistemas e Computação da COPPE/UFRJ, 2001.

[16] S. M. Ross, *Introduction to Probability Models*. Academic Press, 1989.

[17] G. Golub e C. van Loan, *Matrix Computation, 2nd edition*. The John Hopkins University Press, 1989.

[18] W.K.Grassmann, M.I.Taksar e D.P.Heyman, “Regenerative Analysis and Steady State Distributions for Markov Chains”, *Operation Research*, vol. 33, pp. 1107–1116, 1985.

[19] E. de Souza e Silva e R. M. M. Leão, “The TANGRAM-II Environment”, in *Computer Peformance Evaluation - Modelling Techniques and Tools* - $11^{\underline{th}}$ *International Conference (TOOLS2000)*, vol. 1786, pp. 366–369, Springer, Março 2000.

[20] R. Carmo, L. de Carvalho, E. de Souza e Silva, M. Diniz e R. Muntz, “Performance/Availability Modeling with the TANGRAM-II Modeling Environment”, *Performance Evaluation*, vol. 33, pp. 45–65, 1998.

[21] O. Rose, “URL http://nero.informatik.uni-wuerzburg.de/ rose/”. MPEG traces.

[22] P. Skelly, M. Schwartz e S. Dixit, “A Histogram-Based Model for Video Traffic Behavior in an ATM Multiplexer”, *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, no. 4, pp. 445–459, 1993.