MÉTODOS DE SOLUÇÃO PARA MODELOS MARKOVIANOS COM RECOMPENSA

por

Ana Paula Couto Silva

UFRJ

Tese submetida para a obtenção do título de

Mestre em Ciências em Engenharia de Sistemas e Computação

ao Programa de Pós-Graduação de Engenharia de Sistemas e Computação

da COPPE/UFRJ

por

Ana Paula Couto Silva

Outubro 2001

MÉTODOS DE SOLUÇÃO PARA MODELOS MARKOVIANOS COM RECOMPENSA

Ana Paula Couto Silva

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por:

Profa. Rosa Maria Meri Leão, Dr.

Prof. Edmundo de Souza e Silva, Ph.D.

Prof. Valmir Carneiro Barbosa, Ph.D.

Prof. José Roberto Boisson de Marca, Ph.D.

RIO DE JANEIRO, RJ - BRASIL

OUTUBRO DE 2001

SILVA, ANA PAULA COUTO

Métodos de Solução para Modelos Markovianos com Recompensa [Rio de Janeiro] 2001

XVIII, 117 p. 29,7 cm(COPPE/UFRJ, M.Sc., Engenharia de Sistemas e Computação, 2001)

Tese - Universidade Federal do Rio de Janeiro, COPPE

1. Análise Estacionária
2. Análise Transiente
3. Modelos com Recompensa

I. COPPE/UFRJ II. Título (Série)

Aos meus familiares e amigos com muito carinho.

Agradecimentos

A Deus pela sabedoria em todos os momentos e pela oportunidade de alcançar mais um objetivo.

Aos meus pais, Ana Maria e Paulo Araújo, por me incentivar em cada momento, me dando ânimo quando achava que os problemas não tinham saída. Pelo carinho e apoio em todos os momentos. Amo vocês.

A minha irmã, Ana Cristina, que sempre foi minha companheira em todas as horas. Amo você. E ao Marcelo que, mesmo não estando comigo nesta reta final, me apoiou e compreendeu os momentos de ausência durante a elaboração deste trabalho.

A toda a minha família que sempre orou e torceu por mim.

A minha orientadora, Profa. Rosa, que mais que uma orientadora acadêmica foi uma amiga, sempre pronta a dar conselhos, ensinar e ouvir minhas idéias. Muito obrigada. E ao Prof. Edmundo que sempre esteve pronto a me ajudar e a me ensinar.

Aos meus amigos, Drika, Marluce, Kelvin e Flávio, que foram companheiros nos momentos de estudos e aprendizado, não medindo esforços para me ajudar. Aos amigos do LAND, que em todos os momentos estavam presentes e sempre prontos a me ouvir e contribuir com idéias e soluções. Ao Bruno, que me ajudou a superar alguns problemas inesperados, se tornando, dia após dia, uma pessoa cada vez mais especial. A minha amiga Regiane e ao meu amigo Guilherme que mesmo de longe sempre estiveram comigo.

Ao governo brasileiro que através do CNPq concedeu uma bolsa de estudos para o desenvolvimento deste trabalho. Obrigada pela oportunidade e confiança.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

MÉTODOS DE SOLUÇÃO PARA MODELOS MARKOVIANOS COM RECOMPENSA

Ana Paula Couto Silva

Outubro/2001

Orientadora: Rosa Maria Meri Leão
Programa: Engenharia de Sistemas e Computação

Modelos Markovianos com recompensa tem sido amplamente abordados na literatura. Nestes modelos, recompensas de taxa são associadas aos estados e/ou recompensas de impulso são atribuídas às transições entre estados. Modelos com recompensa têm sido usados para o estudo de sistemas mulimídea. Nestes sistemas, o tráfego de entrada e as taxas de transmissão possuem valores bem maiores quando comparados com os valores dos demais eventos do sistema. Devido a esta característica, modelos com recompensa podem ser computacionalmente mais atrativos que os métodos tradicionais.

A ênfase do nosso trabalho é desenvolver algoritmos que permitam obter medidas de performance a partir de modelos Markovianos com recompensa. O primeiro método estudado calcula o valor esperado da recompensa acumulada em um intervalo finito. A técnica utilizada aproxima o valor esperado da recompensa acumulada de taxa no tempo t a partir do tempo aleatório com distribuição erlangiana. Um exemplo de medida de performance que pode ser obtida é o tamanho médio do *buffer* em um intervalo de tempo.

Mostramos, também, que recompensas de impulso podem ser associadas a modelos Markovianos para a obtenção da distribuição do tamanho do *buffer* em estado transiente. O método utilizado para obter esta distribuição é baseado em técnicas encontradas na literatura.

Desenvolvemos um algoritmo que calcula medidas de interesse para estado estacionário, baseado em recompensas de impulso. A matriz gerada a partir deste modelo possui uma estrutura especial que é a chave da nossa técnica. O algoritmo desenvolvido se baseia no algoritmo apresentado em [18] para resolução de modelos que possuam esta estrutura.

O algoritmo possui vantagens computacionais quando comparado com outros métodos encontrados na literatura.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

SOLUTIONS METHODS FOR MARKOVIAN REWARDS MODELS

Ana Paula Couto Silva

October/2001

Advisor: Rosa Maria Meri Leão
Department: Computer and System Engineering

Markovian reward models have been extensively studied in the literature. In these models, reward rates are associated with states and/or reward impulses are attributed to transitions between states. Reward models have also been used to study multimedia systems. In these systems the input traffic and the transmission rates have very high values when compared with those of other events in the system. Due to this characteristic, reward models can be computationally more attractive than other traditional approaches.

Our emphasis in this work is on the development of algorithms to obtain performance measures from markovian reward models. The first method we studied aims at the calculation of the expected cumulative reward over a finite time interval. The technique we used approximates the cumulative reward gained at time t by that calculated at a random time with Erlangian distribution. An example of a performance measure that can be obtained is the expected buffer size averaged over a time interval.

We also show that impulse rewards can be associated to a markovian model to obtain the transient buffer size distribution. The method we used to obtain this distribution is based on known techniques founded in the literature to calculate the distribution of the cumulative impulse reward gained in a finite time interval.

We have developed algorithms to calculate both transient as well as steady state measures. For steady state we base our analysis on impulse rewards The probability transition matrix generated from the model we obtained has a special structure that is the key to our technique. The algorithm we develop extends the work of [18] to solve the class of models we are concerned with. The algorithm presents computational advantages when compared with others approaches in the literature.

Sumário

Lista de Figuras

Lista de Tabelas

Palavras-chave

1. Métodos de Solução
2. Modelos com Recompensa
3. Modelos de Fluido
4. Recompensa com Limites

Capítulo 1

Introdução

A modelagem de sistemas de comunicação/computação é importante para uma análise preliminar do comportamento destes, sendo possível o estudo detalhado e minucioso antes mesmo da implementação real. Desta forma, após a determinação de algumas medidas de interesse, podemos obter resultados precisos de como o sistema reage a certas condições, bem como de características inerentes da sua natureza.

Para representação destes sistemas, modelos matemáticos podem ser utilizados. Dentre estes modelos, os processos estocásticos se destacam. Entende-se como processo estocástico um conjunto de variáveis aleatórias indexadas por um parâmetro de tempo. Assim, em um processo estocástico $\{X(t) : t \geq 0\}$, o índice t é interpretado como tempo e $X(t)$ corresponde ao estado do processo no instante t.

Entre os diversos processos estocásticos encontrados na literatura [25, 22] destaca-se o processo Markoviano. No caso dos processos Markovianos, o comportamento do sistema depende apenas do estado atual do sistema, ou seja, o estado do modelo fornece todas as informações necessárias sobre o sistema.

Juntamente com os processos Markovianos, pode-se utilizar o paradigma de recompensas, onde o processo em estudo recebe uma recompensa quando transiciona entre estados ou quando passa uma unidade de tempo em um certo estado do modelo. A abordagem de modelos Markovianos com recompensa tem sido amplamente usada devido à eficiência na determinação de certas medidas de interesse para o estudo do sistema modelado. Como aplicação de modelos Markovianos com recompensa, consideremos, por exemplo, modelos de confiabilidade. Se associarmos a recompensa $r_i = 1$ para todos os estados s_i que representam que o sistema está operacional e $r_j = 0$ para todos os outros estados, a recompensa acumulada pelo processo é o tempo em que este se encontra operacional.

Modelos com recompensa podem ser resolvidos usando métodos de solução analíticos ou simulação. Considerando os métodos de solução matemáticos, dois grandes grupos podem ser especificados: os métodos que se destinam ao cálculo de medidas em estado transiente, onde o tempo de observação t é considerado curto em relação aos demais tempos de ocorrência de eventos, e os métodos que se destinam ao cálculo de medidas em estado estacionário, onde o tempo de observação t é considerado extremamente longo em relação aos demais tempos de ocorrência de eventos, mais precisamente, quando o tempo $t \to \infty$.

A especificação de um novo método de solução deve ter como principais objetivos a diminuição da complexidade algorítmica, a exploração de estruturas especiais das matrizes de probabilidade e, se possível, a definição de métodos que possibilitem a resolução de modelos que possuem uma grande quantidade de estados, por exemplo, na ordem de 10^6. Nestes casos, recursões e mecanismos especiais devem ser definidos.

Finalmente, com o advento das redes de alta velocidade, as aplicações multimídia se tornaram alvo de estudo. Especificações de modelos que representem o tráfego gerado por estas aplicações têm sido encontradas na literatura. Uma das abordagens utilizadas para modelagem desse tráfego é a utilização de modelos de fluido. Nestes, as altas taxas de transmissão de informações, bem como as altas taxas de geração de informações possibilita utilizarmos a abstração de fluidos para o estudo do comportamento dos *buffers* alimentados por este tráfego. Métodos que utilizam equações diferenciais e transformada de Laplace têm sido utilizados para obtenção das medidas de interesse a partir de modelos de fluido. No entanto, modelos Markovianos com recompensa podem ser utilizados para a obtenção destas medidas. Em geral, os métodos para resolução de modelos com recompensa são matematicamente mais simples e se mostram extremamente eficientes.

1.1 Objetivos da Tese

O objetivo desta tese é apresentar métodos de solução matemáticos utilizados para a resolução de modelos Markovianos com recompensa tanto em estado estacionário quanto em estado transiente. No caso do estudo em estado transiente, será apresentado um método para o cálculo do valor esperado da recompensa de taxa acumulada e mostramos como pode ser obtida a distribuição de probabilidade da ocupação do *buffer* baseada em métodos de solução para modelos com impulso existentes na literatura. Para o estudo em estado estacionário, um novo método de solução para o cálculo da distribuição de probabilidades da ocupação do *buffer* baseado em modelos com impulso é proposto. Todos os métodos apresentados são considerados métodos de aproximação, onde o resultado calculado difere de um pequeno erro em relação ao resultado calculado pelos métodos exatos. Soluções calculadas com baixo custo computacional, mesmo que aproximadas, são extremamente atraentes, especialmente nos casos em que os métodos exatos requerem alto custo computacional, invibializando a utilização destes. A seguir apresentamos a organização da tese, seguida de um breve comentário de cada um dos capítulos.

Nos capítulos 2 e 3 apresentamos a implementação e o estudo de um método de solução para obtenção do valor esperado da recompensa acumulada de taxa em um instante de tempo t de observação [13]. Neste método o valor esperado da recompensa acumulada em t é aproximado por uma variável aleatória com distribuição erlangiana de **R** estágios. Este método de aproximação é indicado para a resolução de modelos onde as taxas de ocorrência dos eventos diferem em muitas ordens de grandeza. Nestes casos, o custo computacional do método de aproximação é significativamente menor que o custo computacional da técnica de uniformização.

No capítulo 4 mostramos como podemos calcular a distribuição transiente da ocupação do *buffer* baseada em um método da literatura para o cálculo da distribuição da recompensa acumulada de impulso. A modelagem proposta possibilita a determinação da distribuição da ocupação do *buffer* alimentado por uma fonte de tráfego, sem que o *buffer* seja modelado. Neste caso, é atribuída uma recompensa de impulso com limites pré-estabelecidos ao processo modelado. No caso particular do *buffer*, estes limites passam a ser iguais a zero e ao seu tamanho total. Este também é um método de aproximação e é indicado para modelos onde o tempo t que representa o total da fase transiente do modelo é expressivo.

No capítulo 5 apresentamos a proposta de um novo método de solução para estudo do comportamento estacionário de um *buffer* alimentado por um fluido. A abordagem é semelhante à apresentada no capítulo 4. São atribuídas recompensas de impulso ao modelo de forma que a matriz de transição de probabilidades (matriz **P**), gerada a partir do modelo possua estrutura especial. O algoritmo proposto é uma extensão do método apresentado em [18] e possui vantagens computacionais quando comparado com outros métodos da literatura.

No capítulo 6 são feitas considerações finais sobre esta tese e propostas para trabalhos futuros.

Capítulo 2

Método de Aproximação para o Cálculo do Valor Esperado da Recompensa de Taxa Acumulada em um Tempo Finito

2.1 Introdução

O estudo do comportamento de um sistema de computação e comunicação em sua fase transiente tem importância quando o foco de interesse está direcionado à análise de medidas que sofrem variações significativas durante um amplo intervalo de observação. Nestes casos, a utilização de resultados em estado estacionário impossibilita o estudo e a análise minuciosa do comportamento do sistema.

Por este motivo, diversos métodos para obtenção de resultados em estado transiente têm sido propostos na literatura [9]. Alguns desses métodos utilizam equações diferenciais para o cálculo da probabilidade do sistema se encontrar em um determinado estado em um instante específico t no tempo. No entanto, estes métodos possuem altos custos computacionais, já que requerem pequenos intervalos de observação para se obter resultados precisos (no caso das equações diferenciais) ou possuem problemas numéricos (no caso de exponenciais de matrizes) [9].

Em alternativa aos métodos tradicionais de análise transiente, novas abordagens têm sido propostas. Uma dessas abordagens utiliza a especificação de recompensas para obtenção de medidas de interesse que refletem o comportamento do sistema [9].

Para a resolução de modelos com recompensas, a técnica de uniformização é amplamente utilizada e aceita. No entanto, quando os modelos possuem característica *stiff*, onde a diferença entre as taxas de transição possui um valor com elevada ordem de grandeza, esta se mostra ineficiente já que o custo computacional é diretamente proporcional à taxa de uniformização do modelo [13].

Em contrapartida, o custo computacional do método proposto em [13], baseado na aproximação de Ross [21] e que também utiliza o paradigma de recompensas, está dissasociado da taxa de uniformização do modelo, estando relacionado a outros fatores específicos. Entre estes fatores específicos destacam-se o custo de se reduzir matrizes ou o custo de se obter o resultado de um sistema de equações utilizando métodos iterativos.

O principal objetivo deste capítulo é implementar e apresentar o método de aproximação proposto em [13] utilizado no cálculo do valor esperado da recompensa de taxa acumulada, descrevendo suas principais características e vantagens. A equação utilizada pela técnica de uniformização para obter o valor esperado da recompensa de taxa acumulada e o seu custo computacional também serão abordados.

A organização do capítulo é descrita a seguir. Na seção 2.2 apresentamos os principais conceitos que envolvem a utilização de recompensas em modelos Markovianos. Na seção 2.3 mostramos a técnica de uniformização sendo utilizada para o cálculo do valor esperado da recompensa acumulada. Na seção 2.4 discutimos a aproximação proposta por Ross para a obtenção de medidas transientes. Na seção 2.5 apresentamos o método proposto, baseado na aproximação de Ross, e as diferentes técnicas utilizadas para o cálculo do valor esperado da recompensa de taxa acumulada.

2.2 Modelos Markovianos com Recompensa

A utilização do paradigma de recompensas para o cálculo de medidas de performance, confiabilidade e *performability* em conjunto com a representação destes sistemas através da utilização de cadeias de Markov, começa a ser amplamente adotado. Este fato se deve à afirmação de que a definição de recompensas é global o suficiente para se adequar ao cálculo da medida em estudo. Entre estas medidas, podemos citar *throughput*, disponibilidade do sistema, tamanho de filas, etc.

A recompensa atribuída a um processo Markoviano pode se classificar em duas categorias distintas:

1. **Recompensas de Taxa** - A cada estado da cadeia é atribuída uma recompensa, sendo que o acúmulo dessa recompensa está relacionado com o tempo de permanência no estado em questão;
2. **Recompensas de Impulso** - A recompensa está associada a transições entre os estados da cadeia.

Para melhor definirmos modelos Markovianos com recompensas de taxa, consideremos uma cadeia de Markov homogênea de tempo contínuo $X = \{X(t), t \geq 0\}$, com espaço de estados finito $S = \{s_i; i = 1, ..., M\}$. A cada estado $s \in S$ está associado uma recompensa de taxa de um conjunto $R = \{r_i; i = 1, ..., M\}$, onde r_i representa a recompensa ganha por unidade de tempo em que o sistema se encontra no estado i. Consideremos, também, a variável aleatória $IR(t) = r_i$ que representa a recompensa instântanea no tempo t, se $X(t) = s_i$. As seguintes medidas de interesse podem ser definidas:

1. **Recompensa de Taxa Acumulada** - Seja $[0, t]$ o período de observação de um sistema em particular. Podemos definir a recompensa acumulada, neste intervalo, como sendo:

   $$CR(t) = \int_0^t IR(\tau) d\tau$$

   Se dividirmos a recompensa acumulada pelo período de observação, obtemos a média da recompensa acumulada:

   $$ACR(t) = \frac{CR(t)}{t}$$

Consideremos, por exemplo, modelos de confiabilidade onde podemos associar $r_i = 1$ para todos os estados s_i onde o sistema está operacional e $r_j = 0$ para os estados onde o sistema não está operacional. Neste caso, a recompensa acumulada $CR(t)$, entre $[0, t]$, é o tempo total em que o sistema está operacional neste intervalo de observação.

A diferença entre os conceitos de recompensa acumulada e de recompensa instântanea pode ser observada na figura 2.1.

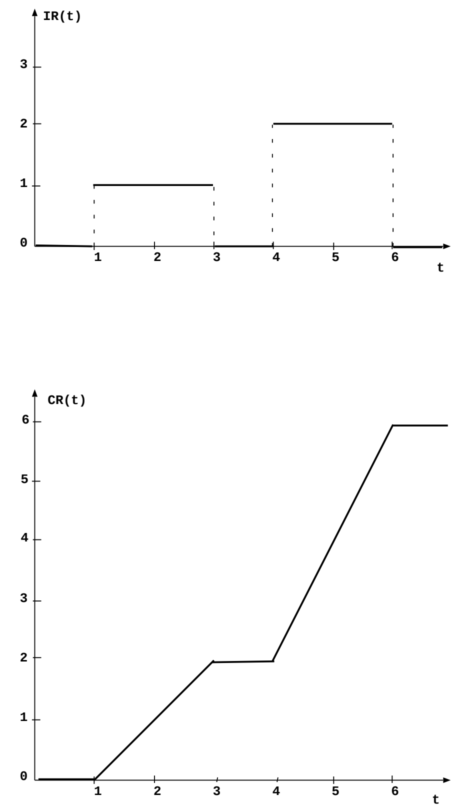


Figura 2.1: Comparação entre IR(t) e CR(t).

2. **Valor Esperado da Recompensa Acumulada**

Além da recompensa acumulada, podemos também estimar o valor esperado da recompensa acumulada que está associada a uma medida de interesse. Para calcular tal medida, utiliza-se a determinação de todos os caminhos amostrais pelos quais o modelo evolui com o tempo, baseados em conceitos de probabilidade.

O valor esperado da recompensa acumulada pode ser definida como:

$$E[CR(t)] = \sum_{i=1}^{M} r_i \int_0^t P[X(\tau) = s_i] d\tau$$

onde M representa o número de estados do modelo.

2.3 A Técnica de Uniformização e a Utilização de Recompensas

A técnica de uniformização ou técnica de randomização é amplamente utilizada na análise de modelos de performance, confiabilidade e *performability*. Esta técnica foi proposta por Jensen em 1953, e tem-se tornado muito popular nos últimos anos para obtenção de medidas transientes em modelos Markovianos [10].

O principal objetivo desta técnica é transformar uma cadeia de Markov de tempo contínuo em uma cadeia de Markov de tempo discreto, o que facilita o cálculo de diversas medidas transientes. Além disso, a utilização da matriz **P** definida como a matriz de probabilidades de transição em um passo no lugar do gerador infinitesimal **Q** evita problemas com erros de arredondamento, provenientes da manipulação dos elementos negativos da diagonal principal de **Q**. Diversos métodos para o cálculo de medidas transientes utilizando a matriz **Q** podem ser encontrados em [10].

Para a análise da técnica de uniformização, consideremos uma cadeia de Markov *X* com espaço de estados finito *S*, constituída por *M* estados. O tempo de permanência em cada estado da cadeia é exponencialmente distribuído e a probabilidade de transição entre estados pode ser definida como sendo igual a q_{ij}/q_i, onde q_{ij} é a taxa de transição entre os estados (s_i, s_j) e q_i é a taxa de saída do estado s_i. Seja Λ um parâmetro cujo o valor, em módulo, é maior ou igual à maior taxa de saída encontrada na matriz **Q**. O valor Λ sempre pode ser calculado, já que o espaço de estados é finito. Assim, podemos definir um processo X' com as seguintes características:

1. *X'* possui o mesmo espaço de estados de *X*;

2. O tempo de residência em qualquer estado da cadeia antes de uma transição é exponencialmente distribuído com taxa Λ. Portanto, o número de transições em $(0, t)$ é dado por uma distribuição de Poisson com a mesma taxa Λ;

3. A probabilidade de *X'* transicionar de um estado s_i para um estado s_j ($s_i \neq s_j$) é igual a q_{ij}/Λ . Todavia, pode existir a transição para o mesmo estado, com probabilidade $(1 - \sum_{j \neq i} q_{ij}/\Lambda)$.

Observando o processo *X'*, podemos notar que tanto a probabilidade de transição entre os estados, como o tempo de permanência em cada estado da cadeia e o espaço de estados são semelhantes aos do processo *X* [14], o que nos possibilita afirmar que os processos *X* e *X'* são semelhantes.

Seja $\mathbf{P} = \mathbf{I} + \mathbf{Q}/\Lambda$, a matriz que fornece as transições de probabilidade entre os estados do processo *X'*. Assim, o processo *X* é equivalente à cadeia de Markov de tempo discreto subordinado a um processo de Poisson. Consideremos uma cadeia de Markov de tempo discreto $Z = \{Z_n : n = 0, 1, ..\}$ com espaço de estados finito *S* e matriz de transições de probabilidade **P**, e seja $N = \{N(t) : t \geq 0\}$ o processo de Poisson com taxa Λ independente de *X*. A transição de tempos de *X'* ocorre de acordo com o processo de Poisson, e as transições entre estados ocorrem de acordo com a cadeia discreta Z e são dadas pela matriz **P**. Assim, $X(t) = X'(t) = Z_N(t)$.

A partir destas considerações, podemos calcular a fração de visitas a cada estado após n transições utilizando a matriz **P**, onde $\mathbf{P} = \mathbf{I} + \mathbf{Q}/\Lambda$, e a seguinte equação, em notação vetorial (onde **v**(0) é o vetor distribuição inicial de probabilidades):

$$\mathbf{v}(n) = \mathbf{v}(0)\mathbf{P}^n \tag{2.1}$$

ou, alternativamente, utilizando a seguinte equação recursiva:

$$\mathbf{v}(n) = \mathbf{v}(n-1)\mathbf{P} \tag{2.2}$$

Descondicionando-se no número total de transições entre $(0, t)$, temos que a distribuição de probabilidade em um instante t é definida pela seguinte equação:

$$\pi(t) = \sum_{n=0}^{\infty} e^{-\Lambda t} \frac{(\Lambda t)^n}{n!} \mathbf{v}(n) \tag{2.3}$$

que é a equação básica da técnica de uniformização.

Consideremos o cálculo do valor esperado de *CR*(t). Sabemos que a variável aleatória $IR(t) = r_i$ representa a recompensa instântanea no tempo t, se $X(t) = s_i$. Assim, o valor esperado para a recompensa instântanea pode ser designado por:

$$E[IR(t)] = \mathbf{r}.\pi(t) \tag{2.4}$$

onde o vetor **r** é o vetor com as recompensas de taxa em cada estado do modelo e o vetor π(t) é o vetor distribuição de probabilidade no instante t, sendo o produto interno entre estes vetores definido como $\mathbf{x}.\mathbf{y} = \sum_{i=1}^{M} x_i y_i$, onde M é o número total de estados do processo.

Substituindo a equação 2.3 na equação 2.4, temos:

$$E[IR(t)] = \sum_{n=0}^{\infty} e^{-\Lambda t} \frac{(\Lambda t)^n}{n!} \mathbf{r}.\mathbf{v}(n) \tag{2.5}$$

Para obtermos o valor esperado da recompensa acumulada entre $(0, t)$, basta integramos o valor esperado da recompensa instântanea neste intervalo de observação:

$$E[CR(t)] = \int_0^t \sum_{n=0}^{\infty} e^{-\Lambda x} \frac{(\Lambda x)^n}{n!} \mathbf{r}.v(n) dx \tag{2.6}$$

Manipulando a equação 2.6:

$$E[CR(t)] = \frac{1}{\Lambda} \sum_{n=0}^{\infty} \mathbf{r}.v(n) \int_0^t e^{-\Lambda x} \frac{\Lambda(\Lambda x)^n}{n!} dx \tag{2.7}$$

Observando que o termo de integração é a função de densidade da distribuição erlangiana com $n + 1$ estágios [10], chegamos à seguinte equação:

$$E[CR(t)] = t \sum_{n=0}^{\infty} e^{-\Lambda t} \frac{(\Lambda t)^n}{n!} \left[\frac{\sum_{j=0}^{n} \mathbf{r}.v(j)}{n+1} \right] \tag{2.8}$$

A equação 2.8 possui a seguinte interpretação probabilística: dado que ocorreram n transições do processo uniformizado entre $(0, t)$, o período de observação é dividido em $n + 1$ intervalos, que são determinados pelo processo de Poisson $N(t)$. O valor esperado do tamanho de cada intervalo é $t/(n + 1)$. Considerando que a distribuição de probabilidade no j-ésimo intervalo é **v**(j), o produto de t pelo termo entre colchetes em 2.8 é o valor esperado da recompensa acumulada dado n transições [10].

Tanto para o cálculo de probabilidade quanto para o cálculo do valor esperado, as equações 2.3 e 2.8 contêm uma série infinita que deve ser truncada. Este limite N é estimado através do erro tolerado para a medida, e é obtido utilizando a seguinte equação:

$$\epsilon(N) = 1 - \sum_{n=0}^{N} e^{-\Lambda t} \frac{(\Lambda t)^n}{n!}$$

A partir da definição de N, pode-se estimar o custo computacional do cálculo do valor esperado da recompensa acumulada utilizando a técnica de uniformização. O custo computacional depende tanto do valor de N, ponto de truncagem da série de *Poisson*, quanto da estrutura esparsa da matriz **P**. Se o número médio de transições para fora do estado s_i é d (número total de entradas diferentes de zero em um linha i de **P**) e o total de estados é igual a M, então o número de multiplicações necessárias para obtenção das medidas tem a ordem de grandeza limitada superiormente por $O(NMd)$ [14].

Se considerarmos valores elevados de Λt , a distribuição de Poisson pode ser aproximada por uma curva normal de parâmetros (Λt , Λt). Logo, pode-se mostrar que *N*, e conseqüentemente o custo do método são proporcionais a Λt [10]. Em razão disso, para modelos onde as taxas q_{ij} têm valores que diferem por diversas ordens de grandezas, o custo da uniformização é bem alto.

2.4 A Aproximação de Ross

Ross propõe uma nova equação para o cálculo da probabilidade de estado em um instante t [21] e, com pequenas modificações, esta equação pode ser aplicada ao cálculo do valor esperado da recompensa de taxa acumulada.

Nesta nova abordagem a probabilidade dos estados do modelo passa a ser calculada para um intervalo de tempo aleatório com distribuição erlangiana. Ou seja, o tempo t de interesse deixa de ser um tempo determinístico e passa a ser um tempo aproximado através de uma soma de diversas variáveis com distribuição exponencial. Mostra-se que o cálculo da probabilidade para este último caso é bem mais fácil do que para o caso em que o tempo t é determinístico.

Para tal, consideremos um sistema modelado por uma cadeia de Markov de tempo contínuo X. Suponhamos que este sistema esteja sendo observado a intervalos que são representados pelas variáveis aleatórias $Y_i = \{Y_1, Y_2, Y_3, \ldots\}$. Estas possuem distribuição exponencial com taxa λ e são independentes de X. Seja o evento de observação a X denominado ε. Se considerarmos que X está no estado i, podemos dizer que o evento transição em X ocorre anteriormente ao evento de observação ε com a probabilidade $q_i/(q_i + \lambda)$. Da mesma maneira, o evento observação ε ocorrerá anteriormente a uma transição em X com a probabilidade $\lambda/(q_i + \lambda)$. Estas probabilidades são, respectivamente, a probabilidade de saída do estado i e a probabilidade de permanência neste estado antes de uma nova observação. Caso ocorra a saída do estado i, a probabilidade de transicionar para o estado j é q_{ij}/q_i. A figura 2.2 ilustra os processos X e Y e a relação entre as transições e os eventos de observação.

Condicionando à ocorrência de um dos eventos, isto é, ao evento ε ou ao evento transição em X, e usando a propriedade Markoviana podemos definir $\pi_{ij}(Y_1) = P[Y_1 = j|Y_0 = i]$ como sendo:

$$\pi_{ij}(Y_1) = \frac{q_i}{q_i + \lambda} \sum_k \pi_{kj}(Y1) \frac{q_{ik}}{q_i} + \frac{\lambda}{(q_i + \lambda)} \delta_{ij}, \tag{2.9}$$

onde $\delta_{ij} = 1$ se $i = j$ e $\delta_{ij} = 0$, caso contrário.

Seja $Y(r) = \{Y_1 + Y_2 + .. + Y_r\}$ a variável aleatória erlangiana independente de X. A partir da equação 2.9 a seguinte equação recursiva pode ser obtida, lembrando que X permanece no mesmo estado se ε ocorre antes de uma transição em X:

$$\pi_{ij}(Y(r)) = \frac{1}{q_i + \lambda} \sum_k \pi_{kj}(Y(r)) q_{ik} + \frac{\lambda}{(q_i + \lambda)} \pi_{ij}(Y(r-1)) \tag{2.10}$$

Figura 2.2: Aproximação de Ross.

onde π_{ij}(Y(0))= δ_{ij}.

Reescrevendo a equação anterior na forma matricial, temos:

$$-\lambda\mathbf{\Pi}(r-1) = (\mathbf{Q} - \lambda I)\mathbf{\Pi}(r) \tag{2.11}$$

onde $\mathbf{\Pi}$(r) = [π_{ij}(Y(r))]. Finalmente:

$$\mathbf{\Pi}(r) = (\mathbf{I} - \mathbf{Q}/\lambda)^{-1}\mathbf{\Pi}(r-1) \tag{2.12}$$

O cálculo da probabilidade pode ser efetuado através da aplicação recursiva da equação acima, no total de R vezes. Como $Y(R)$ é uma variável erlangiana de parâmetros (R, λ)

o seu valor médio, é dado por $E[Y(R)] = R/\lambda$. Definindo $\lambda = R/t$, fazemos com que o tempo de observação t seja igual a $E[Y(R)]$. Assim, quando fazemos $R \to \infty$, a variância tende a 0 e $E[Y(R)]$ tende a $Y(R)$ com probabilidade 1, o que, conseqüentemente, faz com que $Y(R) \to t$. Portanto, podemos escolher R grande o suficiente para que a precisão desejada no cálculo de $\mathbf{\Pi}(t)$ seja alcançada.

2.5 O Método de Aproximação e o Cálculo do Valor Esperado da Recompensa Acumulada

A equação para o cálculo do valor esperado da recompensa de taxa acumulada pode ser obtida a partir da equação 2.10. No entanto, a equação proposta por Ross possui um inconveniente: deve-se inverter a matriz obtida através da subtração de **Q** da matriz identidade de mesma ordem, resultando em um alto custo computacional. Em razão disto, o método se torna eficiente somente para modelos onde o espaço de estados é reduzido.

Uma nova equação baseada no método de aproximação de Ross é apresentada em [13]. Nesta, a recompensa acumulada é calculada levando em conta as características da matriz **Q**: sua estrutura especial (matrizes banda) ou sua estrutura esparsa, reduzindo o custo de processamento e de armazenamento.

Inicialmente, o processo Markoviano *X* que representa o modelo é discretizado e a matriz de transições de estados **P**, obtida através de **Q**, passa a ser utilizada. Uma equação recursiva do tipo **Ax = b** é obtida. Esta equação pode ser resolvida por métodos iterativos tradicionais ou por um método direto também proposto em [13].

Com esta nova equação, o custo computacional para o cálculo do valor esperado da recompensa de taxa acumulada, diferentemente do custo computacional da uniformização, está mais relacionado com a estrutura da matriz de transição de probabilidades do modelo em estudo. Assim, o custo passa a estar relacionado com o custo de um método iterativo para resolução de sistemas de equação ou com a redução de uma matriz, no caso do método direto. Estaremos mostrando que a equação, proposta e implementada, é extremamente atraente para o cálculo da recompensa em modelos com características *stiff*.

2.5.1 A Equação Proposta em [13]

Consideremos o conjunto de recompensas $R = \{r_1, r_2, r_3, ...\}$. A cada estado s_i é associada uma recompensa específica, que é acumulada a cada instante de tempo em que a cadeia permanece neste estado.

Seja *X'* o processo discretizado que representa o processo original *X* e **P** a matriz de transição de estados, onde cada elemento p_{ij} representa a probabilidade de transição do estado i para o estado j. O tempo que o processo *X'* passa em cada estado antes de uma transição é Λ e o tempo entre ocorrências de um evento de observação ε é λ. Assim, o valor esperado do tempo entre ocorrências de eventos no modelo, sendo este uma transição na cadeia ou um evento de observação ε, é $1/(\Lambda + \lambda)$. O valor esperado para a recompensa de taxa acumulada durante um período de observação pode ser obtido pela seguinte recursão [13]:

$$c_{ij}(Y(r)) = \frac{r_i}{(\Lambda + \lambda)}\delta_{ij} + \frac{\Lambda}{(\Lambda + \lambda)}\sum_k c_{kj}(Y(r))p_{ik} + \frac{\lambda}{(\Lambda + \lambda)}c_{ij}(Y(r-1)) \quad (2.13)$$

onde $\delta_{ij} = 1$ para $i = j$, $\delta_{ij} = 0$ caso contrário e $c_{ij}(Y(0)) = 0$.

Podemos notar que o primeiro termo do lado direito da equação 2.13 representa a recompensa r_i que está sendo acumulada a cada unidade de tempo em que a cadeia se encontra no mesmo estado. Esta situação acontece quando a transição da cadeia ocorre para o mesmo estado. O segundo termo representa a ocorrência de uma transição na cadeia. Como a equação 2.13 é uma equação do tipo *backward*, temos um acúmulo de recompensas do estado j para um estado intermediário k, no estágio r e uma transição de i para k. Para este termo, devemos considerar todos os estados da cadeia como possíveis estados intermediários. O terceiro termo representa a ocorrência do evento de observação ε. Neste caso, não existem transições na cadeia e a recompensa de taxa acumulada no estágio anterior é somada ao ganho obtido no estágio atual.

A equação 2.13 pode ser escrita na seguinte forma matricial:

$$\left(\mathbf{I} - \frac{\Lambda}{(\Lambda+\lambda)}\mathbf{P}\right)\mathbf{C}(r) = \frac{1}{(\Lambda+\lambda)}\mathbf{R} + \frac{\lambda}{(\Lambda+\lambda)}\mathbf{C}(r-1) \tag{2.14}$$

onde a matriz **R** é uma matriz diagonal, cujo o i-ésimo elemento da diagonal representa a taxa r_i e **C** é a matriz onde os elementos representam a recompensa acumulada em cada estado, considerando todos os possíveis estados iniciais.

Consideremos o vetor coluna $\mathbf{1}_j$ como sendo o vetor onde o j-ésimo elemento é igual a 1 e os demais elementos são iguais a zero. Sendo $\mathbf{c}(j,r) = \mathbf{C}(\mathbf{r})\mathbf{1}_j$ e substituindo este termo na equação 2.14, temos:

$$\left(\mathbf{I} - \frac{\Lambda}{(\Lambda+\lambda)}\mathbf{P}\right)\mathbf{c}(j,r) = \frac{1}{(\Lambda+\lambda)}\mathbf{R}\mathbf{1}_j + \frac{\lambda}{(\Lambda+\lambda)}\mathbf{c}(j,r-1) \tag{2.15}$$

Se considerarmos que o tempo de observação t está sendo aproximado por uma distribuição erlangiana com **R** estágios, o vetor $\mathbf{c}(j,R)$ se torna uma boa aproximação para a j-ésima coluna da matriz de recompensa acumulada $\mathbf{C}(t)$ e o i-ésimo elemento do vetor $\mathbf{c}(j,R)$ se torna uma boa aproximação para $c_{ij}(t)$.

A partir da equação 2.14 é possível a obtenção do valor esperado da recompensa de taxa acumulada para um subconjunto de estados Ω dentro do espaço de estados da cadeia de Markov. Para tal, basta preenchermos as entradas do vetor $\mathbf{1}_\Omega$ correspondentes aos estados pertencentes ao subconjunto com o valor 1. As demais entradas permanecerão com o valor zero. Assim, fazendo $\mathbf{c}(\Omega,r) = \mathbf{C}(r)\mathbf{1}_\Omega$ e substituindo na equação 2.14, obtemos o valor esperado da recompensa de taxa acumulada considerando um subconjunto. Isto é verdadeiro dado que o vetor $\mathbf{c}(\Omega,r)$ é a soma das colunas da matriz $\mathbf{C}(r)$ que representam os estados pertencentes ao subconjunto Ω.

2.5.2 A Técnica Iterativa

Ao analisarmos a equação 2.15 podemos notar que esta possui a forma **Ax = b**, onde a matriz **A** é a matriz resultante do cálculo de $\mathbf{I} - \Lambda/(\Lambda+\lambda)\mathbf{P}$, o vetor **x** é representado por $\mathbf{c}(j,r)$ e **b** é representado por $1/(\Lambda+\lambda)\mathbf{R}\mathbf{1}_j + \mathbf{c}(j,r-1)$. Dessa maneira, podemos utilizar alguns métodos iterativos propostos na literatura [14] para a resolução do sistema de equações apresentado.

Os métodos iterativos clássicos são adequados para modelos onde a matriz de taxas **Q** é esparsa e possui uma ordem elevada. Assim, a matriz **P** será esparsa e, conseqüentemente, a matriz **A** também será esparsa. As iterações realizadas não promovem o enchimento da matriz, evitando problemas de armazenamento.

Alguns dos métodos iterativos utilizados são os métodos de Jacobi, Gauss-Seidel, *Sucessive Over-Relaxation* - SOR e o *Power*. Estes métodos possuem pequenas variações, sendo que o principal objetivo é obter de maneira mais eficiente as soluções do sistema [14]. Para cada iteração, em todos os métodos, o custo computacional é aproximadamente igual ao custo da multiplicação de um vetor por uma matriz esparsa. Assim, o número total de iterações é determinante para o custo computacional de cada método iterativo.

Seja a i-ésima equação pertencente ao sistemas de equação:

$$\sum_{j=1}^{N} a_{ij}x_j = b_i$$

Resolvendo para a variável x_i obtemos:

$$x_i = \frac{-\sum_{j=i,j\neq i}^{N} a_{ij}x_j + b_i}{a_{ii}} \tag{2.16}$$

A cada iteração um novo valor para x_i é calculado. O número total de iterações está relacionado com o critério de parada escolhido: número máximo de iterações, o erro entre soluções de iterações consecutivas, número de iterações baseados no autovalor subdominante da matriz, etc [14].

Método de Jacobi

Neste método, os valores para x_j, $j \neq i$, na equação 2.16 são valores que foram previamente calculados no passo anterior. Assim:

$$x_i^{(k)} = \frac{-\sum_{j=i,j\neq i}^{N} a_{ij}x_j^{(k-1)} + b_i}{a_{ii}} \tag{2.17}$$

Método de Gauss-Seidel

Na equação 2.17 somente os valores de x_j calculados durante a iteração anterior são utilizados para o cálculo de x_i na iteração atual. No entanto, valores para $j < i$, estão disponíveis na iteração atual e espera-se que estejam bem mais próximos da resposta esperada.

Em razão disso, o método de Gauss-Siedel utiliza para o cálculo de x_i todas as soluções já calculadas na iteração atual. Assim:

$$x_i^{(k)} = \frac{-\sum_{j=1}^{i-1} a_{ij}{x_j}^{(k)} - \sum_{j=i+1}^{N} a_{ij}x_j^{k-1} + b_i}{a_{ii}} \tag{2.18}$$

Uma consideração a ser feita é que no método de Gauss somente as mais recentes estimativas para cada x_i devem ser armazenadas. Desde que a matriz original permaneça inalterada, os requerimentos de armazenamento do método de Gauss-Seidel são mínimos, o que é importante para resolver sistemas com muitas variáveis [14].

A utilização de valores mais recentes para o cálculo de x_i pode fazer com que o método de Gauss-Seidel convirja mais rapidamente para solução esperada do que o método de Jacobi. No entanto, este fato nem sempre é verdadeiro, já que a taxa de convergência depende dos autovalores e autovetores da matriz **P** [14].

Método *Sucessive Over-Relaxation* **- SOR**

Através de uma pequena modificação no método de Gauss-Seidel, podemos definir o método SOR. Esta modificação visa promover uma convergência mais rápida para a solução esperada.

Neste método iterativo, tanto a solução obtida no passo k quanto todos os valores obtidos no passo $k+1$ para $j < i$ são utilizados para a obtenção do valor de x_i no passo atual $(k+1)$. A importância de cada componente da solução para o resultado no passo atual é dada através de um coeficiente denominado parâmetro de relaxação w.

$$x_i^{(k)} = (1-w)x_i^{(k)} + w\left(\frac{-\sum_{j=1}^{i-1} a_{ij}{x_j}^{(k)} - \sum_{j=i+1}^{N} a_{ij}x_j^{k-1} + b_i}{a_{ii}}\right) \tag{2.19}$$

Observando 2.19, podemos notar que o método de Gauss é um caso particular do método SOR, para o parâmetro de relaxação igual a 1.

O maior desafio do método SOR é a obtenção de um valor ótimo para o parâmetro w. Este valor deve proporcionar a maior diferença possível entre o autovalor dominante e o autovalor sub-dominante, resultando assim, em uma taxa de convergência mais rápida para a solução [14]. Estudos mostram que o método SOR somente converge se $0 < w < 2$.

Método *Power*

Neste método iterativo a solução é obtida através de sucessivas multiplicações de um vetor **x** por uma matriz **A**. Este método sempre converge para solução desejada. No entanto, na grande maioria das vezes, a taxa de convergência para solução final é extremamente lenta.

2.6 A Técnica Direta de [13]

Uma outra maneira de se resolver a equação 2.15 é através da utilização do método direto que é baseado na eliminação de blocos. A eliminação de blocos é indicada para modelos onde a matriz $A = (\mathbf{I} - \Lambda/(\Lambda + \lambda)\mathbf{P})$ possui uma estrutura regular ou simétrica, como por exemplo matrizes do tipo banda. Matrizes do tipo banda são aquelas em que os elementos não-nulos se organizam em torno da diagonal principal, ficando os elementos nulos na parte mais periférica da matriz. Assim, a estrutura em banda da matriz permite a manipulação de seus blocos sem que haja o preenchimento do blocos originais com valores não-nulos, resultando em importante economia de espaço em memória.

O método direto proposto para a resolução do sistema de equações 2.14 primeiramente particiona a matriz $A = (\mathbf{I} - \Lambda/(\Lambda + \lambda)\mathbf{P})$ em diversos blocos. Baseado nesta partição, a cada passo é eliminado um bloco, resultando em operações de multiplicações entre os blocos e de inversões dos blocos diagonais. Ao final da eliminação um pequeno sistemas de equações é resolvido e juntamente com os blocos modificados os valores de $\mathbf{c}(\mathbf{r})$ são determinados. Em razão da matriz **A** ser previamente dividida em blocos e da intensa manipulação destes é que este método possui um performance melhor quando o modelo possui uma matriz **P** com estrutura do tipo banda.

A descrição da técnica direta é feita a seguir. Consideremos que a matriz **A** esteja inicialmente particionada em blocos da seguinte maneira:

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} & \cdots & \mathbf{A}_{1K} \\ \mathbf{A}_{21} & \mathbf{A}_{22} & \cdots & \mathbf{A}_{2K} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}_{K1} & \mathbf{A}_{K2} & \cdots & \mathbf{A}_{KK} \end{bmatrix} \tag{2.20}$$

Consideremos, também, que o vetor $\mathbf{c}(\mathbf{r})$ esteja particionado em K blocos $\{\mathbf{c_1}(r)$, $\mathbf{c_2}(r)$, $\mathbf{c_3}(r)$, $\cdots$, $\mathbf{c_k}(r)\}$ com as mesmas dimensões das partições em **A**. A partir dessas considerações, podemos reescrever a equação 2.15 da seguinte maneira:

$$\sum_{j=1}^{K} \mathbf{A}_{ij}^{(0)} \mathbf{c}_j(r) = \mathbf{t}_i^{(0)}(r) \qquad i = 1, 2, \cdots, K \tag{2.21}$$

onde

$$\mathbf{A}_{ij}^{(0)} = \mathbf{A}_{ij} \quad \text{e} \quad \mathbf{t}_i^{(0)}(r) = \tfrac{1}{(\Lambda+\lambda)} \mathbf{R} \mathbf{1}_j + \tfrac{\lambda}{(\Lambda+\lambda)} \mathbf{c}_i(r-1) \tag{2.22}$$

Sendo os blocos da diagonal inversíveis [13], podemos obter o valor de $\mathbf{c}_1$, com $i = 1$, utilizando a equação 2.21. Assim:

$$\mathbf{c}_1(r) = -(\mathbf{A}_{11}^{(0)})^{-1} \sum_{j=2}^{K} \mathbf{A}_{1j}^{(0)} \mathbf{c}_j(r) + (\mathbf{A}_{11}^{(0)})^{-1} \mathbf{t}_1^{(0)}(r) \tag{2.23}$$

substituindo $\mathbf{c}_1$ em 2.21, temos:

$$\sum_{j=2}^{K} \underbrace{[\mathbf{A}_{ij}^{(0)} - \mathbf{A}_{i1}^{(0)} (\mathbf{A}_{11}^{(0)})^{-1} \mathbf{A}_{1j}^{(0)}]}_{\mathbf{A}_{ij}^{(1)}} \mathbf{c}_j(r) = \underbrace{[\mathbf{t}_i^{(0)}(r) - \mathbf{A}_{i1}^{(0)} (\mathbf{A}_{11}^{(0)})^{-1} \mathbf{t}_i^{(0)}(r)]}_{\mathbf{t}_i^{(1)}} \quad i = 2, 3, \cdots, K \tag{2.24}$$

A equação 2.24 é a equação base do método de redução. Esta pode ser reescrita da seguinte forma $\mathbf{A}^{(1)}\mathbf{c}^{(1)}(r) = \mathbf{t}^{(1)}(r)$, onde $\mathbf{c}^{(1)}(r)$ é idêntico a $\mathbf{c}^{(0)}(r)$ sem o primeiro bloco $\mathbf{c}_1(r)$, $\mathbf{A}^{(1)}$ possui dimensão $(K-1)$ x $(K-1)$ e $\mathbf{t}^{(1)}(r)$ possui dimensão $(K-1)$.

O procedimento de redução da matriz passa a ser aplicado recursivamente para $k = \{2, 3, 4, \cdots, K-1\}$. Após $K-1$ passos, a seguinte matriz é obtida:

$$\mathbf{A}_{KK}^{(K-1)}\mathbf{c}_K(r) = \mathbf{t}_K^{(K-1)}(r) \tag{2.25}$$

A figura 2.3 ilustra a matriz **A** e os vetores **c**(r) e **t**(r), após a redução.

$$\begin{bmatrix} \mathbf{A}_{11}^{(0)} & \mathbf{A}_{12}^{(0)} & \mathbf{A}_{13}^{(0)} & & \cdots & & \mathbf{A}_{1K}^{(0)} \\ \mathbf{A}_{21}^{(0)} & \mathbf{A}_{22}^{(1)} & \mathbf{A}_{23}^{(1)} & \mathbf{A}_{24}^{(1)} & \cdots & & \mathbf{A}_{2K}^{(1)} \\ \mathbf{A}_{31}^{(0)} & \mathbf{A}_{32}^{(1)} & \mathbf{A}_{33}^{(2)} & \mathbf{A}_{34}^{(2)} & & \cdots & \mathbf{A}_{3K}^{(2)} \\ \vdots & \mathbf{A}_{42}^{(1)} & \mathbf{A}_{43}^{(2)} & \mathbf{A}_{44}^{(3)} & \mathbf{A}_{45}^{(3)} & \cdots & \mathbf{A}_{4K}^{(3)} \\ . & . & . & . & . & \cdots & . \\ . & . & . & . & . & \cdots & . \\ . & . & . & . & . & \cdots & . \\ \mathbf{A}_{K1}^{(0)} & \mathbf{A}_{K2}^{(1)} & \mathbf{A}_{K3}^{(2)} & \mathbf{A}_{K4}^{(3)} & & \cdots & \mathbf{A}_{KK}^{(K-1)} \end{bmatrix} \begin{bmatrix} \mathbf{c}_1 \\ \mathbf{c}_2 \\ \mathbf{c}_3 \\ \mathbf{c}_4 \\ . \\ . \\ . \\ \mathbf{c}_K \end{bmatrix} = \begin{bmatrix} \mathbf{t}_1^{(0)} \\ \mathbf{t}_2^{(1)} \\ \mathbf{t}_3^{(2)} \\ \mathbf{t}_4^{(3)} \\ . \\ . \\ . \\ \mathbf{t}_K^{(K-1)} \end{bmatrix}$$

Figura 2.3: Matriz $\mathbf{A}$ e vetores $\mathbf{c}$ e $\mathbf{t}$ após a redução

A equação 2.25 pode ser resolvida por um método direto, como por exemplo, a eliminação gaussiana, fornecendo $\mathbf{c}_K$. Os outros blocos do vetor **c** podem ser obtidos através da fórmula:

$$\mathbf{c}_i(r) = (\mathbf{A}_{ii}^{(i-1)})^{-1}\left[\mathbf{t}_i^{(i-1)}(r) - \sum_{j=0}^{(K-i-1)} \mathbf{A}_{i,K-j}^{(i-1)}\mathbf{c}_{K-j}(r)\right] \tag{2.26}$$

Podemos reescrever o algoritmo de redução da matriz **A** através dos passos descritos a seguir. Devemos ressaltar que o processo da redução da matriz é realizado somente uma única vez.

Algoritmo:

1. Inicialização

$$\mathbf{A}^{(0)} = \left(\mathbf{I} - \frac{\Lambda}{(\Lambda + \lambda)}\mathbf{P}\right) \tag{2.27}$$

$$\mathbf{t}^{(0)}(0) = \mathbf{c}(0) * \frac{\lambda}{(\Lambda + \lambda)} + \frac{1}{(\Lambda + \lambda)}\mathbf{R1_j} \tag{2.28}$$

onde **c**(0) possui todos os valores iguais a zero, já que no início do período de observação não existe o acúmulo de recompensa.

2. Procedimento de redução da matriz **A**

Para $1 \leq k \leq K - 1$ e $k + 1 \leq i, j \leq K$:

$$\mathbf{A}_{ij}^{(k)} = \mathbf{A}_{ij}^{(k-1)} - \mathbf{A}_{ik}^{(k-1)}(\mathbf{A}_{kk}^{(k-1)})^{-1}\mathbf{A}_{kj}^{(k-1)} \tag{2.29}$$

3. Cálculo do vetor de recompensa acumulada **c**(r), para os **R** estágios da distribuição erlangiana

Atualização de **t**(r):

$$\mathbf{t}^{(0)}(r) = \mathbf{c}(r-1) * \frac{\lambda}{(\Lambda + \lambda)} + \frac{1}{(\Lambda + \lambda)}\mathbf{R1_j} \tag{2.30}$$

Para $1 \leq r \leq R$:

Cálculo de **t**(r):

para $1 \leq k \leq K-1$ e $k+1 \leq i \leq K$:

$$\mathbf{t}_i^{(k)}(r) = \mathbf{t}_i^{(k-1)}(r) - \mathbf{A}_{ik}^{(k-1)}(\mathbf{A}_{kk}^{(k-1)})^{-1}\mathbf{t}_k^{(k-1)}(r) \tag{2.31}$$

Cálculo de **c**(r):

para $K \leq i \geq 1$:

$$\mathbf{c}_i(r) = (\mathbf{A}_{ii}^{(i-1)})^{-1}\left[\mathbf{t}_i^{(i-1)}(r) - \sum_{j=0}^{(K-i-1)} \mathbf{A}_{i,K-j}^{(i-1)}\mathbf{c}_{K-j}(r)\right] \tag{2.32}$$

A figura 2.4 mostra os diversos blocos da matriz **A** envolvidos no cálculo de $\mathbf{A}_{ij}^{(k)}$ e de $\mathbf{t}_i^{(k)}$.

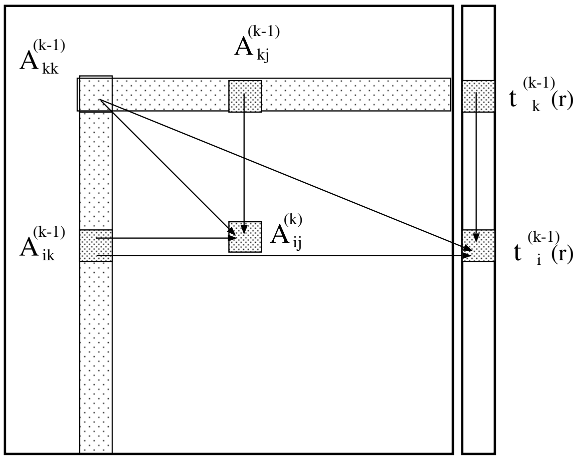


Figura 2.4: Manipulação dos blocos pelo algoritmo.

Para o cálculo do custo computacional da técnica direta, consideremos, sem perda de generalidade, que a matriz **A** possua K blocos de dimensão B x B e que a matriz original tenha um número elevado de elementos não-nulos. Além disso, consideremos também que a matriz original não tenha uma estrutura especial, caracterizando, com isso, a análise do custo no pior caso.

Tendo estabelecido as considerações anteriores, podemos dividir o custo computacional da técnica direta em três pontos principais: o cálculo da matriz $\mathbf{A}_{ij}^{(k)}$, realizado no processo de redução; o cálculo do vetor **c**(r), que representa a recompensa acumulada até o estágio **r** da distribuição erlangiana e o cálculo do vetor **t**(r).

Para o cálculo da redução da matriz são necessárias $O((BK)^3)$ multiplicações, já que devemos realizar multiplicações entre matrizes para um *loop* alinhado com três laços (o primeiro variando entre 1 e o número total de blocos e os demais variando para as linhas e colunas relacionadas).

Para a obtenção dos valores dos blocos do vetor **t**(r), são necessárias $O(B^2(K^2 + K)/2)$ multiplicações e para computar o valor do vetor **c**(r) são necessárias $O(B^2K^2/2)$ multiplicações.

É válido ressaltar que o processo de redução da matriz **A** é efetuado uma única vez. O cálculo de **c**(r) e **t**(r) é realizado **R** vezes. Assim, o peso do custo computacional do cálculo de **c**(r) e **t**(r) será diretamente proporcional ao número total de estágios da distribuição erlangiana utilizado para aproximar o tempo determinístico t.

2.7 Estudo da Precisão do Método de Aproximação

Para o estudo da precisão do método de aproximação apresentado, suponha que a medida de interesse seja o vetor $\mathbf{r}(\tau)$ onde o i-ésimo elemento representa a recompensa acumulada pelo processo em um subconjunto de estados Ω no tempo τ começando do estado i, isto é, $\mathbf{r}(\tau) = \mathbf{C}(\tau)\mathbf{1}_\Omega$. O erro do método de aproximação depende da variabilidade da medida $\mathbf{r}(t)$ em torno do tempo τ aproximado e da variabilidade da variável aleatória erlangiana em torno da sua média. Assim, considerando que o desvio padrão da distribuição erlangiana com taxa $\lambda = R/\tau$ é $\tau/\sqrt{R}$, podemos definir [13]:

$$\epsilon_\Omega = \left[\frac{\delta\mathbf{r}(t)}{\delta t}\right]_{t=\tau} \frac{\tau}{\sqrt{R}} = \mathbf{Q}\mathbf{r}(\tau) \tag{2.33}$$

Fazendo $\mathbf{Q} = [(\mathbf{P} - \mathbf{I})\Lambda]$ e substituindo na equação 2.33, temos:

$$\epsilon_\Omega = [\mathbf{P} - \mathbf{I}]\mathbf{r}(\tau)\frac{\Lambda\tau}{\sqrt{R}} \tag{2.34}$$

O valor de ϵ_Ω é uma estimativa de erro no tempo τ, onde a segunda igualdade é obtida através da *Kolmogorov's backward equation*. A equação 2.33 depende

Capítulo 3

Aplicações do Método de Aproximação - Recompensa de Taxa Acumulada

3.1 Introdução

A modelagem e análise de sistemas de computação têm recebido crescente atenção de pesquisadores que desejam entender e estudar o comportamento destes sistemas. Assim, a modelagem tem como principal objetivo responder perguntas a respeito da performance e confiabilidade de sistemas e protocolos que ainda não foram efetivamente desenvolvidos. A partir do estudo dos modelos e dos resultados obtidos, e se estes se mostram satisfatórios, podemos implementar o novo protocolo ou tornar realidade o projeto de um novo sistema.

O processo de modelagem se subdivide em várias etapas. Primeiramente, devemos promover um estudo minucioso do sistema ou protocolo a ser modelado. A partir deste estudo, podemos passar à especificação do modelo. Com o modelo especificado, passamos para a fase de solução e obtenção de medidas de interesse. Nesta etapa podemos optar pela solução matemática, onde o modelo é mapeado para um processo estocástico, como, por exemplo, o processo Markoviano, ou podemos optar pela utilização da simulação. Caso o caminho escolhido seja a solução matemática, métodos analíticos devem ser utilizados. Ao final, a solução matemática e a simulação convergem para a obtenção das medidas de interesse, conforme podemos observar na figura 3.1.

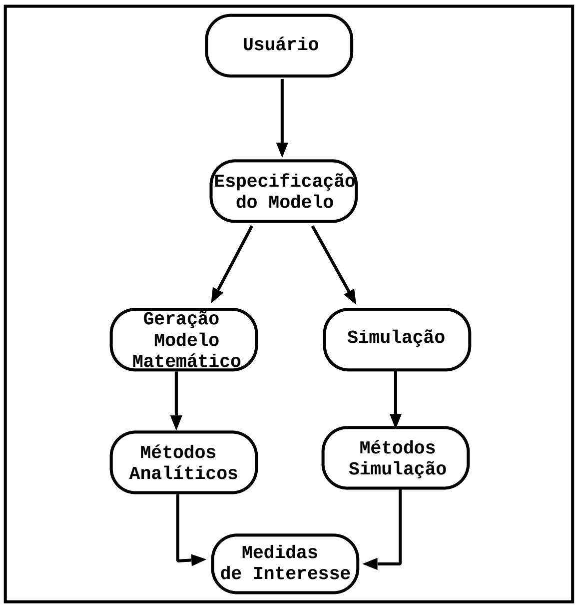


Figura 3.1: Os principais passos da modelagem.

Devido a complexidade do processo de modelagem, ferramentas são implementadas para possibilitar a geração de um ambiente integrado onde todos os passos da modelagem possam ser executados. Além da integração e da utilização de uma interface amigável para o usuário, a ferramenta deve prover uma maneira natural de especificação do modelo, ou seja, o modelo deve ser especificado da maneira mais próxima possível da forma como o usuário pensa. Uma outra questão importante é a integração de métodos de solução que se adequam a análise do modelo em estado transiente e/ou estacionário.

TANGRAM-II [2, 11] é uma importante ferramenta de modelagem e análise de sistemas de computação e comunicação. Desenvolvida pelo Laboratório de Análise e Desempenho de Sistemas de Computação e Comunicação da Universidade Federal do Rio de Janeiro [17], visa propósitos educacionais e de pesquisa e tem como principais objetivos fornecer um ambiente integrado e amigável para a modelagem de sistemas, bem como oferecer um conjunto de ferramentas multimídia para serem utilizadas no processo de modelagem e no trabalho em grupo.

Escrita nas linguagens C, C++ e com interface em Java, a ferramenta TANGRAM-II permite a especificação dos modelos através da utilização do paradigma de orientação a objetos. Além disso, permite ao usuário obter medidas de interesse em estado estacionário e/ou transiente através de um amplo conjunto de métodos de solução.

Neste capítulo usaremos a ferramenta TANGRAM-II para modelagem de três sistemas de computação que serão utilizados para o estudo comparativo entre o método de aproximação para obtenção do valor esperado da recompensa de taxa acumulada no intervalo $(0, t)$ e a técnica tradicional de uniformização. Estes dois métodos se encontram integrados a ferramenta.

A organização deste capítulo é descrita a seguir. Na seção 3.2 estaremos mostrando algumas considerações a respeito da implementação do método de aproximação, usando as técnicas direta e iterativa. A seção 3.3 apresenta três sistemas de computação juntamente com resultados numéricos da aplicação do método de aproximação e da técnica de uniformização. A seção 3.4 apresenta um exemplo do estudo da estimativa do erro originado pela utilização do método de aproximação apresentado neste trabalho.

3.2 Implementação do Método de Aproximação para o cálculo do Valor Esperado da Recompensa de Taxa Acumulada - Técnica Direta e Técnica Iterativa

O método de aproximação para o cálculo do valor esperado da recompensa de taxa acumulada - técnicas direta e iterativa - se encontra integrado a ferramenta TANGRAM-II. Como os demais métodos de solução analítica, o método proposto também está implementado utilizando a linguagem C, tendo a interface de entrada dos dados escrita na linguagem Java.

Para a implementação do método, rotinas anteriormente desenvolvidas foram utilizadas (rotinas para redução de matrizes, alocação de matrizes esparsas, manipulação de elementos na matriz ...). No entanto, uma série de modificações foram realizadas para a obtenção da equação proposta:

1. Criação de rotinas para o cálculo dos vetores da recompensa acumulada **c**(r) e do elemento independente **t**(r);

2. Estruturação e manipulação dos arquivos de entrada para o método de solução, entre eles: arquivo com probabilidade inicial, arquivo com o intervalo de observação, arquivo com os estados destinos (utilizado para o cálculo da recompensa acumulada em um subconjunto de estados);

3. Adaptação dos métodos iterativos. Novo critério de parada: número máximo de iterações. Nesta nova opção, o usuário pode determinar um número máximo de iterações que devem ser realizadas pelo o método iterativo escolhido (SOR, Gauss-Siedel, *Power* ou Jacobi). Este critério de parada é útil para os modelos onde a matriz **A** possui um autovalor sub-dominante próximo a 1, pois esta característica faz com que a convergência do método seja lenta [14]. Caso o método convirja antes do número total de iterações estabelecido, a execução do método é dada como encerrada;

4. Na técnica iterativa o usuário pode especificar diferentes valores de **R** para diferentes intervalos de observação. Esta flexibilidade possibilita aproximar os primeiros intervalos, onde existem uma maior variabilidade das medidas, com um número maior de estágios exponenciais, ficando os últimos intervalos de tempo com um número de estágios relativamente menor. Esta especificação também resulta em uma diminuição do custo computacional para o cálculo da medida de interesse.

   Na técnica direta esta opção não está disponibilizada, pois para cada novo valor de **R** uma nova redução matricial deve ser feita, o que elevaria o custo computacional a valores exorbitantes, visto que, conforme veremos nos exemplos a seguir, a redução matricial é o passo com maior custo computacional;

5. Especificação da interface em Java.

3.2.1 Técnica Direta

A interface da técnica direta para o cálculo da recompensa acumulada no intervalo $(0, t)$ pode ser visualizada na figura 3.2.

A utilização do método, bem como a especificação de cada parâmetro de entrada podem ser encontrados em [5].

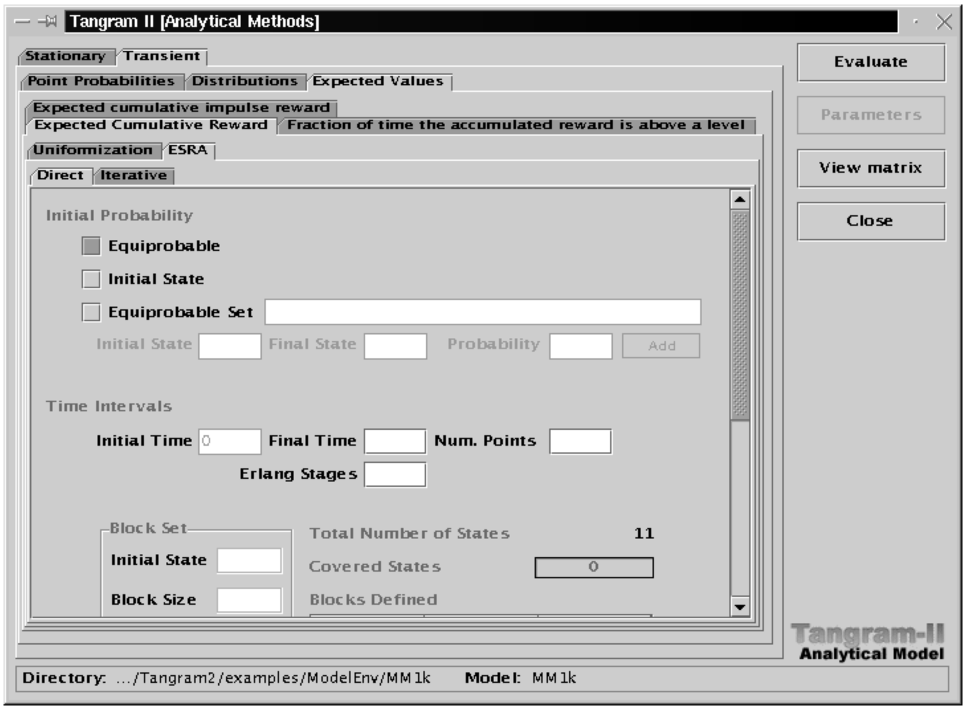


Figura 3.2: A técnica direta para obtenção do valor esperado da recompensa de taxa acumulada.

3.2.2 Técnica Iterativa

A interface da técnica iterativa para o cálculo da recompensa acumulada no intervalo $(0, t)$ pode ser visualizada na figura 3.3.

A utilização do método, bem como a especificação de cada parâmetro de entrada podem ser encontrados em [5].

3.3 Exemplos

Nesta seção são mostrados resultados numéricos da comparação de custos entre o método de aproximação e a técnica de uniformização para o cálculo do valor esperado da recompensa de taxa acumulada, bem como a eficácia do resultado produzido pelo método proposto. Para tal, três modelos com características *stiff* foram utilizados.

O primeiro modelo representa um canal multiplexador que alterna o envio de pacotes de duas filas com tráfegos distintos. Este tipo de modelo pode representar um canal de

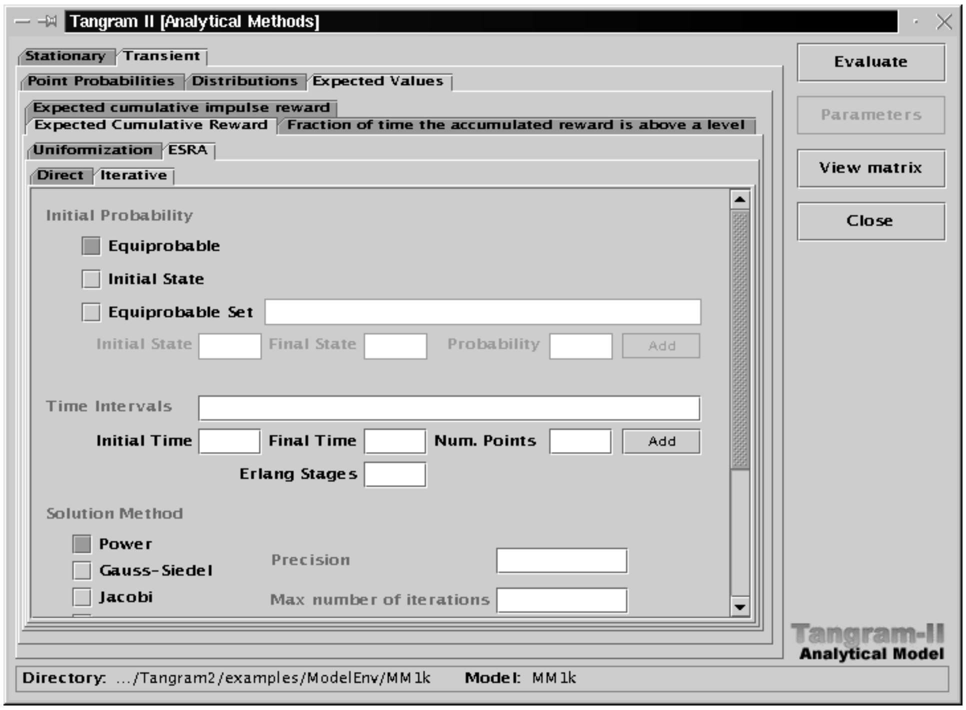


Figura 3.3: A técnica iterativa para obtenção do valor esperado da recompensa de taxa acumulada.

comunicação que serve duas filas tratando de forma diferenciada os tráfegos, fazendo, por exemplo, com que um deles tenha uma maior prioridade no envio de pacotes. Este tipo de modelo também pode representar uma fila que implementa a disciplina de serviço WFQ (*Wheighted Fair Queuing*).

O segundo modelo estuda o comportamento de um sistema composto por duas filas, cujos pacotes de dados armazenados são servidos por um conjunto de servidores. Estes servidores possuem uma característica especial: em intervalos de tempo exponencialmente distribuídos, interrompem o serviço de pacotes. Estes servidores são denominados *vacation servers*. Após um intervalo de tempo, também exponencialmente distribuído, os servidores retornam a servir pacotes.

O terceiro modelo estuda o comportamento de um sistema que está sendo sobrecarregado por pacotes de dados. Para tal, modelamos uma fonte de pacotes cuja a taxa de geração de pacotes é maior que a taxa de serviço. Assim, a fila que armazena os pacotes de dados fica, na maior parte do tempo, com uma grande quantidade de pacotes. Este cenário é ideal para estudo e dimensionamento de *buffers* em redes de dados.

Variável	Descrição
Canal.Token	Indica comutação ou filas vazias
Canal.Último_Servido	Indica a última fila servida
Canal.Filas_Vazias	Número de filas sem pacotes
Fila[1,2].Token	Indica que a fila está sendo servida
Fila[1,2].Buffer	Indica quantidade total de pacotes na fila

Tabela 3.1: Variáveis de estado do modelo canal multiplexador.

3.3.1 Modelo 1: Canal Multiplexador

O sistema modelado neste primeiro exemplo consiste em um canal multiplexador que alterna o envio de pacotes pertencentes a duas filas, denominadas q_1 e q_2, que recebem tráfego de origens diferentes.

O comportamento do canal multiplexador, que opera pelo sistema de *polling*, é descrito a seguir. A fila q_1 é servida enquanto houver pacotes ou até expirar o seu tempo máximo de serviço (*timeout*). Logo após o serviço à fila q_1, a fila q_2 passa a enviar os seus pacotes, reiniciando todo o processo descrito. Quando as filas se encontram vazias, o servidor entra em *estado de espera* e serve imediatamente a fila que primeiramente receber um novo pacote.

Seguindo o paradigma utilizado pela ferramenta TANGRAM-II [2, 11], cada estado do modelo possui as variáveis de estado que podem ser visualizadas na tabela 3.1. Na tabela 3.2 listamos os parâmetros utilizados no modelo. Podemos notar que a alta taxa de comutação (tempo decorrido entre o fim do serviço à fila i e o começo do serviço à fila seguinte, $i + 1$) e as baixas taxas de *timeout*, fazem com que o modelo tenha características *stiff*.

Como o principal objetivo deste modelo é estabelecer um estudo do comportamento do método de aproximação e da técnica de uniformização para o cálculo do valor esperado da recompensa de taxa acumulada, as seguintes medidas são definidas:

1. **Erro absoluto** - módulo da diferença entre a solução real (técnica de uniformização)

Parâmetros	Valores
Tempo médio de comutação entre as filas	0,0001s
Intervalo médio entre pacotes (Fila 1)	1,33s
Tempo médio de serviço (Fila 1)	0,26s
Timeout (Fila 1)	2,00s
Número máximo de pacotes (Fila 1)	12 pacotes
Intervalo médio entre pacotes (Fila 2)	0,5s
Tempo médio de serviço (Fila 2)	0,36s
Timeout (Fila 2)	0,008s
Número máximo de pacotes (Fila 2)	8 pacotes

Tabela 3.2: Parâmetros do modelo canal multiplexador.

e a solução aproximada (método proposto);

2. **Erro relativo** - erro absoluto dividido pela solução real.

A seguir apresentamos o desempenho do método proposto, considerando a técnica iterativa e a técnica direta. Um estudo comparativo entre o custo computacional alcançado pelo método de aproximação para o cálculo do valor esperado da recompensa de taxa acumulada e a técnica de uniformização também é realizado. Para este estudo, a medida de interesse utilizada foi a proporção de tempo em que q_1 permanece com o *token*, representando, assim, a proporção de tempo em que o serviço foi oferecido à fila em questão.

Desempenho da Técnica Iterativa

Nesta seção estaremos estudando o comportamento do erro absoluto, do erro relativo e do custo computacional do método de aproximação - técnica iterativa, utilizando o modelo do canal multiplexador e o método iterativo de Gauss-Siedel. Para tal, duas situações foram consideradas, onde:

1. variamos o número total de estágios R e fixamos o tempo final de observação e o número total de instantes de interesse;

2. variamos o número total de instantes de interesse e fixamos o tempo final de observação e o número total de estágios R.

Considerando a primeira situação, estabelecemos como tempo final de observação o instante $t = 5s$ e 1 como o número total de instantes de interesse. A variação do valor do total de estágios R está estabelecida dentro do intervalo $[5, 50]$. A figura 3.4 mostra os resultados.

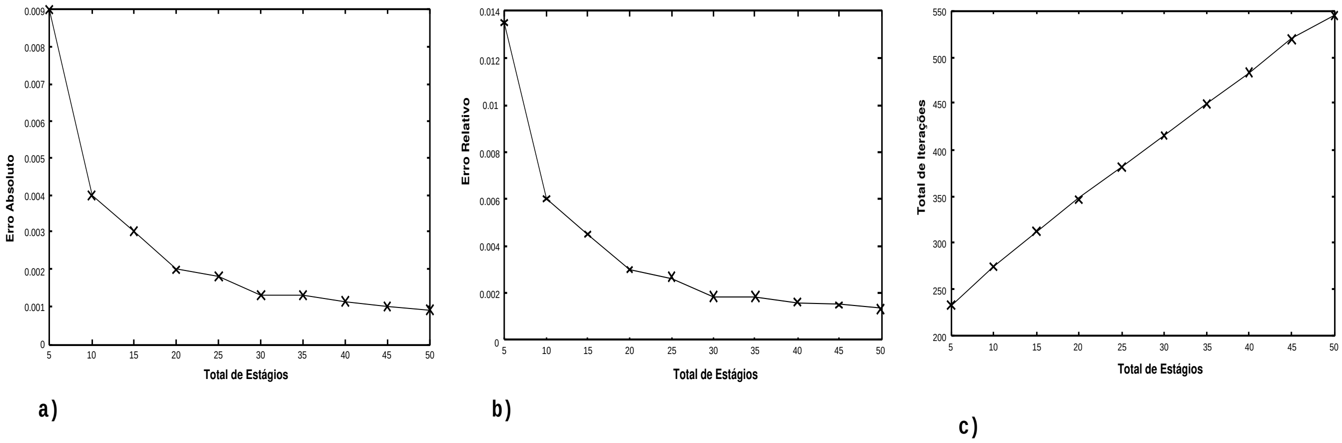


Figura 3.4: Variação de R: a) Erro absoluto; b) Erro Relativo; c) Número total de iterações.

Analisando a figura 3.4 podemos notar que o erro absoluto juntamente com o erro relativo decaem quase que exponencialmente a medida em que aumentamos o valor de R. Este comportamento reflete exatamente a idéia de que se fizermos $R \to \infty$, a variância tende a 0 e a variável erlangiana tende ao valor do tempo determinístico t. Em relação ao número total de iterações, podemos observar que este tem um aumento linear, fazendo com que o número total de multiplicações necessárias varie dentro do intervalo $[441.960, 1.038.225]$. Para a técnica de uniformização o total de multiplicações necessárias foi de $97.307.400$, o que demonstra que para se obter um erro na ordem 10^{-3} o número total de multiplicações, utilizando o método de aproximação, é duas ordens de grandeza menor que o número total alcançado pela técnica de uniformização. Para se ter uma maior precisão (erro na ordem de 10^{-4}) o custo total alcançado pelo método de aproximação é 93 vezes menor. Este fato mostra que o método proposto é extremamente atraente para os modelos com características *stiff*.

Para a segunda situação, calculamos a medida de interesse para o mesmo instante final de observação $t = 5s$. Variamos o total de instantes de interesse dentro do intervalo $[1, 10]$. Para cada instante de interesse foram utilizados 10 estágios exponenciais. A figura 3.5 mostra os resultados.

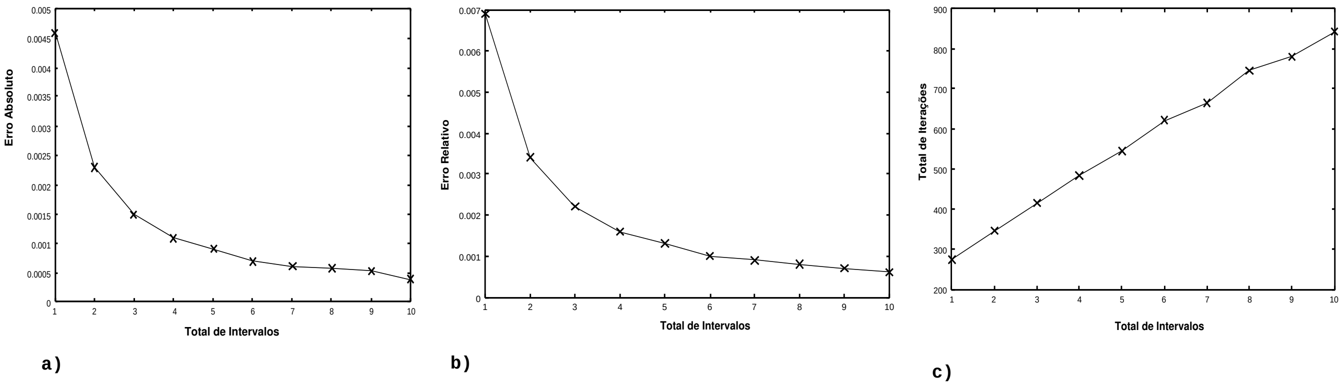


Figura 3.5: Variação do total de instantes: a) Erro absoluto; b) Erro Relativo; c) Número total de iterações.

Segundo a implementação do método de aproximação, para cada um dos instantes de interesse R estágios exponenciais estão sendo utilizados. A recompensa obtida no instante $n - 1$ é utilizada como recompensa inicial para o instante de interesse posterior. Dessa maneira, se para um instante final de interesse t temos **n** instantes de interesse intermediários aproximados por R estágios exponenciais, podemos afirmar que t está sendo aproximado por **nR** estágios exponenciais. Assim, o tempo $t = 5s$ está sendo aproximado por um número total de estágios que varia no intervalo $[10s, 100s]$. Desta maneira, o erro absoluto, o erro relativo e o total de iterações necessárias possuem comportamento semelhante ao comportamento da variação no número total de estágios R.

Através destes dois cenários, podemos observar que o método de aproximação possibilita calcularmos medidas de interesse com uma alta precisão, mesmo para a técnica iterativa, onde a solução difere de um erro em relação a solução real do problema. Além disso, o custo computacional do método de aproximação se mostra significativamente menor que o custo computacional da técnica de uniformização.

Partição	Total de Multiplicações
1	1.465.145
2	789.676
3	2.436.727
4	10.948.399
5	14.468.350

Tabela 3.3: Diferentes particionamentos e os respectivos custos computacionais.

Desempenho da Técnica Direta

Para estudar o desempenho da técnica direta, estaremos computando o custo computacional para diferentes configurações de divisão de blocos. Os erros absoluto e relativo não serão abordados nesta seção, já que possuem o mesmo comportamento descrito no estudo da técnica iterativa.

As diferentes divisões de blocos utilizadas, escolhidas aleatoriamente, foram as seguintes:

1. Partição 1 - 10 blocos de dimensão 44 e 1 bloco de dimensão 5;
2. Partição 2 - 20 blocos de dimensão 22 e 1 bloco de dimensão 5;
3. Partição 3 - 1 bloco de dimensão 445;
4. Partição 4 - 2 blocos de dimensão 200 e 1 bloco de dimensão 45;
5. Partição 5 - 1 bloco de dimensão 300 e 1 bloco de dimensão 145.

O número total de multiplicações necessárias para o procedimento de redução da matriz **A** estão listadas na tabela 3.3.

Conforme podemos observar na tabela 3.3, o particionamento da matriz **A** influencia diretamente no número total de multiplicações necessárias para obtenção da solução utilizando a técnica direta. A partição 2 resulta em um menor custo computacional, enquanto a partição 5 resulta em um custo 18 vezes maior do que o custo alcançado pela partição que gera o menor custo computacional.

No entanto, é extremamente difícil estabelecer um particionamento favorável a obtenção de um custo computacional reduzido, a não ser que tenhamos uma ferramenta que possibilite a visualização da estrutura da matriz utilizada, bem como vibialize uma possível divisão em blocos que seja favorável a utilização de técnicas diretas. Na próxima seção apresentamos uma ferramenta integrada ao TANGRAM-II, que permite a visualização da estrutura da matriz **P** e o seu particionamento em blocos.

3.3.2 Modelo 2: *Vacation Servers*

A descrição do segundo modelo é feita a seguir. Consideremos o sistema composto por duas fontes de dados: uma do tipo *Poisson* que continuamente envia pacotes com intervalos de geração exponencialmente distribuídos e outra fonte do tipo *ONOFF*, onde existem dois períodos distintos: um período ativo de envio de pacotes e um período de silêncio, onde não existe envio de pacotes. O tempo de permanência em cada período da fonte *ONOFF* e o intervalo de chegada entre pacotes são exponencialmente distribuídos.

Cada uma das fontes envia os seus pacotes para *buffers* distintos. Cada *buffer* possui no máximo cinco servidores. Assim, a formação de fila no *buffer* somente ocorre se o número de pacotes for maior que o número máximo de servidores. O *buffer* ligado a fonte *ONOFF* repassa os pacotes de dados ao outro *buffer* quando atinge a sua ocupação total.

Além dessas características, os servidores possuem períodos exponencialmente distribuídos onde os pacotes deixam de ser servidos, e por esta razão o *buffer* se enche. Após este período de "férias", os servidores retornam a servir os pacotes, diminuindo, assim, o número de pacotes nos *buffers*. É devido a este período de recesso no serviço de pacotes que estes servidores são denominados *Vacation Servers*.

Os parâmetros utilizados se encontram listados na tabela 3.4. Podemos notar que é devido as altas taxas de envio e serviço de pacotes e as baixas taxas do período de "férias" que temos um modelo *stiff*.

Para o estudo do comportamento deste sistema, duas medidas de interesse foram utilizadas: o tamanho médio do *buffer* ligado a fonte *Poisson* e o tamanho do *buffer* ligado a fonte *ONOFF*. Para tal, devemos definir uma recompensa que acumula o tamanho do

Parâmetros	Valores
Tempo médio de serviço (para cada servidor)	0,0000136s
Intervalo médio entre pacotes (fonte *ONOFF*)	0,00000136s
Tempo médio entre périodos de "férias"	1.000s
Tempo médio em que o servidor permanece em "férias"	400s
Intervalo médio entre pacotes (fonte de *Poisson*)	0,0000068s
Número máximo de servidores (em cada *buffer*)	5 servidores
Tamanho do *buffer*	20 pacotes

Tabela 3.4: Parâmetros do modelo *vacation servers*.

buffer a cada instante de tempo que permanecemos em um determinado estado. Ao final, se dividirmos o valor esperado da recompensa de taxa acumulada pelo tempo total de observação, teremos o tamanho médio do *buffer*.

As recompensas são assim definidas, utilizando a linguagem de especificação de modelos da ferramenta TANGRAM-II [5]:

```
rate_reward   = média_buffer1
   condition  = (TRUE)
   value      = buffer;

rate_reward   = média_buffer2
   condition  = (TRUE)
   value      = buffer;
```

Observamos o comportamento transiente da medida no intervalo de tempo $[1s, 200s]$. Subdividimos o intervalo de tempo entre $t = 1s$ e $t = 10s$ em 10 subintervalos aproximados individualmente por 10 estágios exponenciais. Assim, estamos aproximando o tempo determinístico $t = 10s$ através da utilização de 100 estágios exponenciais. Para o tempo entre $t = 10s$ e $t = 200s$ foram utilizados 20 subintervalos que também foram aproximados individualmente por 10 estágios exponenciais. Esta estratégia de obtenção de resultados foi utilizada para que pudéssemos acompanhar os primeiros intervalos de tempo com uma maior precisão, visto que são nestes intervalos que a medida de interesse sofre uma variação mais brusca. Para a obtenção dos resultados, foi utilizada a técnica direta.

O comportamento do tamanho médio de cada *buffer* pode ser visualizado através da figura 3.6.

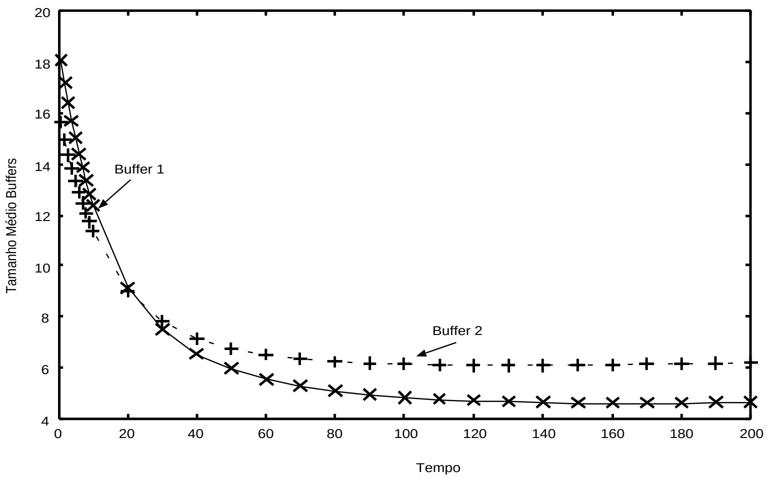


Figura 3.6: Tamanho médio do *buffer*1 e *buffer*2.

Podemos observar que, a princípio, o *buffer* ligado a fonte *ONOFF* atinge no instante $t = 1s$ 90% da sua ocupação total. Este fato decorre da fonte estar inicialmente no período ativo e enviar uma rajada de pacotes. Além disso, a capacidade total de serviço dos servidores ligados ao *buffer*1 é de, no máximo, 365.550 pacotes/s, o que equivale a 50% da taxa de pacotes enviados pela fonte quando em período ativo, o que também influencia no enchimento acelerado do *buffer*. A partir do tempo $t = 30s$, o *buffer* já se encontra com um número médio de pacotes abaixo de 50% de sua capacidade. Esta diminuição do número de pacotes reflete a transição da fonte para o período de silêncio.

O *buffer* ligado a fonte *Poisson* atinge no instante $t = 1s$ 80% da sua ocupação total, apesar da taxa de envio de pacotes pela fonte *Poisson* não chegar a 50% da capacidade máxima dos seus servidores. Os seguintes fatos influenciam diretamente no comportamento do *buffer*2:

1. o enchimento do *buffer*1 ocasiona a transferência dos pacotes para o *buffer*2;

2. a fonte *Poisson* envia continuamente pacotes ao *buffer*2;

3. o período de recesso no serviço de pacotes.

Estes fatos fazem com que a média do *buffer*2 seja maior que a média do *buffer* 1 no instante $t \approx 25s$.

Para completar a análise do sistema, calculamos, em estado estacionário, o tamanho médio do *buffer*1. Obtivemos um valor de, aproximadamente, 6,45 pacotes armazenados no *buffer*. Ora, podemos observar pela figura 3.6 que a medida da média de recompensa acumulada tende ao resultado em estado estacionário. Este fato era de se esperar, pois quando analisamos o processo em estado estacionário temos a proporção de tempo em que passamos em cada estado do processo. Assim, os valores de média e de média da recompensa acumulada convergem. Uma outra observação a ser feita é a importância da análise em estado transiente. O valor em estado estacionário, neste caso, não reflete satisfatoriamente o comportamento dos *buffers* ao longo do tempo. Assim, o resultado em estado estacionário não seria útil, por exemplo, para estudo do comportamento de *buffers* em casos de congestionamento em redes. Esta afirmação é aplicada a sistemas onde o tempo para se atingir o estado estacionário é considerado longo e as medidas possuem uma alta variação. Caso contrário, a análise em estado estacionário se mostra satisfatória, acarretando, para alguns casos, um custo computacional menor.

O Custo Computacional e a Comparação com a Técnica de Uniformização.

Passemos ao estudo do custo computacional alcançado pelo método de aproximação. Para a obtenção de cada medida, considerando o primeiro subintervalo de observação $t = [1s, 10s]$ e dividindo a matriz **A** em 11 blocos, sendo 10 blocos com 350 estados cada e 1 bloco com 28 estados, foram necessárias 995.008.596 multiplicações para a redução da matriz **A**, 471.174.900 multiplicações para o cálculo do vetor de recompensa acumulada **c**(100) (já que o tempo $t = 10s$ foi aproximado utilizando **R** = 100) e 593.958.400 multiplicações para o cálculo do vetor independente **t**(100). Assim, o custo total do método ficou na ordem de 10^9 multiplicações.

No entanto, este custo computacional pode ter uma redução significativa. Sabemos que a técnica direta para resolução de sistemas é indicada para sistemas onde a matriz possui uma estrutura especial, cujos elementos não-nulos estejam posicionados bem próximos a diagonal principal. Assim, cada iteração do método não provoca o enchimento da matriz original que ocasiona o aumento de espaço de memória para armazenamento dos elementos, bem como não provoca o aumento do número total de multiplicações.

Iremos agora ilustrar como o fato de reorganizar a matriz pode reduzir o custo computacional do método. A figura 3.7 mostra a matriz **P** originalmente gerada pelo TANGRAM-II. Para determinar novas estruturas para esta matriz, a fim de serem utilizadas pelos métodos de solução com intuito de diminuição do número total de multiplicações, os estados do modelo devem ser reordenados. Esta reordenação pode ser alcançada através do modulo de visualização e permutação de matrizes, integrado a ferramenta TANGRAM-II.

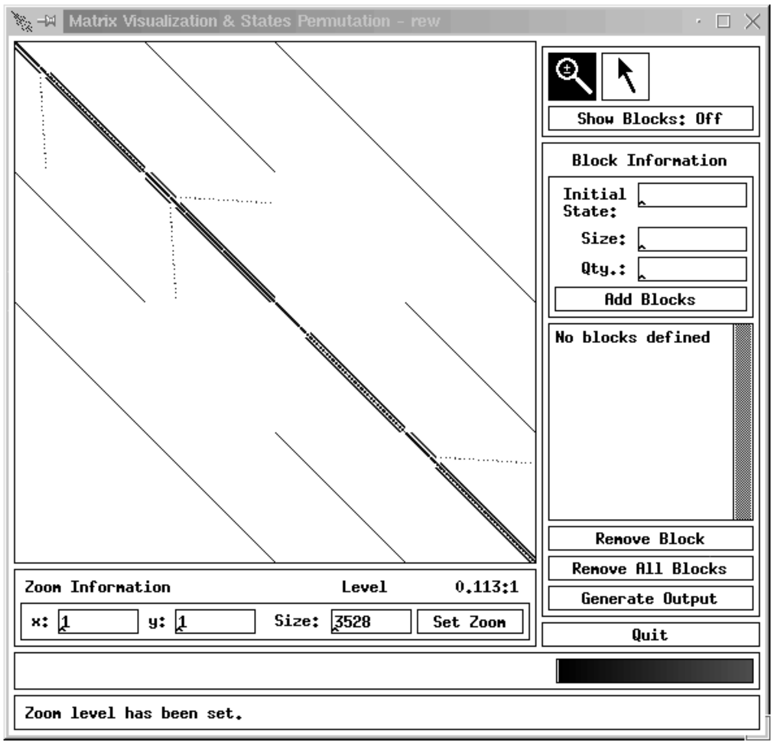


Figura 3.7: Estrutura original da matriz **P**.

A nova estrutura para a matriz **P**, após a permutação das variáveis do modelo, pode ser visualizada na figura 3.8. Podemos observar que a nova matriz **P** possui uma estrutura onde os elementos não-nulos se localizam próximos a diagonal. Utilizando esta nova estrutura e para uma divisão em blocos idêntica a da matriz original, foram necessárias 453.604.226 multiplicações para o procedimento de redução da matriz **A**, 111.181.350

multiplicações para o cálculo do vetor de recompensa acumulada e 233.808.400 multiplicações para o cálculo do vetor independente. O custo total do método passou a ser de 10^8 multiplicações, com uma queda de uma ordem de grandeza em relação ao proporcionado pela matriz original do modelo. Com esta nova estrutura, o custo do processo de redução diminui pela metade, o custo do cálculo do vetor de recompensa acumulada diminui em $4,2$ vezes e o custo do cálculo do vetor independente diminui em $2,5$ vezes.

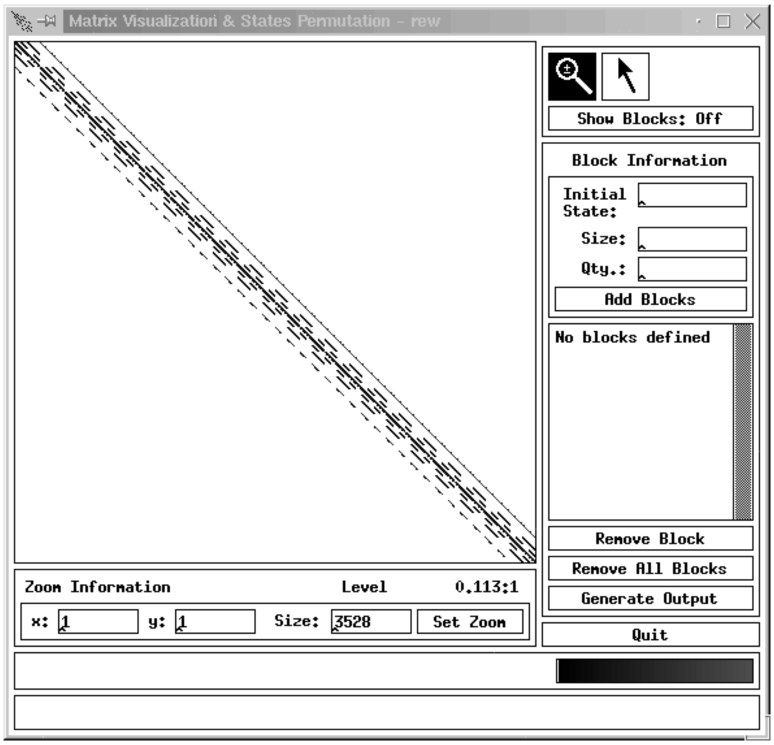


Figura 3.8: Nova estrutura da matriz **P**.

Esta flexibilidade na obtenção de novas estruturas, que acarreta um aumento da eficiência no cálculo da solução desejada, é uma das vantagens do método de aproximação. No caso da utilização da técnica de uniformização, esta flexibilidade não pode ser alcançada, já que o número total de multiplicações está relacionada com o número total de elementos não-nulos da matriz **P**.

Passemos ao estudo do custo da técnica de uniformização utilizando as mesmas medidas já definidas. Devido ao alto custo computacional da técnica de uniformização, a resolução da medida de interesse para este modelo foi limitada ao intervalo de observação $t = 1s$. O custo computacional é descrito a seguir. Sabemos que o número total de multiplicações necessárias pela técnica de uniformização é da ordem $O(NdM)$, onde **N**

Medidas	Método de Aproximação	Uniformização
Buffer 1	18,00384	18,08730
Buffer 2	15,62067	15,55836

Tabela 3.5: Medidas obtidas: método de aproximação e a técnica de uniformização.

representa o ponto onde o termo de *Poisson* é truncado, **d** é o número médio de entradas não-nulas em uma coluna da matriz de transição de probabilidade **P** e **M** é o número total de estados do modelo. Além disso, o valor de **N** é proporcional ao valor de Λ obtido no processo de uniformização da cadeia de Markov de tempo contínuo e ao tempo de observação do processo.

Tendo feito estas considerações, podemos efetuar o custo computacional da técnica de uniformização para o cálculo do valor esperado da recompensa de taxa acumulada no tempo de observação $t = [0s, 1s]$. Temos, para este modelo, $\Lambda = 1,6$x10^6 e $t = 1s$, fazendo $N \approx 1,6$x10^6. O número total de entradas não-nulas da matriz **P** é de 22.512. Assim, o custo total da técnica de uniformização é da ordem de 10^{10} multiplicações, ou seja, duas ordens de grandeza acima do custo do método de aproximação, sendo que o custo do método de aproximação apresentado está relacionado a um intervalo de observação 10 vezes maior que o intervalo de observação abrangido pela técnica de uniformização.

Vale ressaltar que o custo do método de uniformização é diretamente proporcional ao tempo de observação. Assim, se calculássemos o comportamento da medida para um intervalo de tempo até $t = 200s$, o custo computacional aumentaria duas ordens de grandeza, chegando a 10^{12} multiplicações.

Os resultados obtidos pelo método de aproximação e a técnica de uniformização, no instante de tempo $t = 1s$, podem ser visualizados na tabela 3.5. Podemos observar que o resultado obtido pelo método proposto está bem próximo ao valor obtido pela técnica de uniformização. A diferença entre os resultados é da ordem de 10^{-2}, o que mostra a eficácia do método de aproximação.

Parâmetros	Valores
Tempo médio de serviço do servidor erlangiano	0,00125s
Intervalo médio entre pacotes (fonte *ONOFF*)	0,0011s
Tempo médio período ativo	1.000s
Tempo médio período de silêncio	333,33s
Total de estágios do servidor erlangiano	30 estágios
Tamanho do *buffer*	50 pacotes

Tabela 3.6: Parâmetros do modelo *Servidor Erlangiano*.

3.3.3 Modelo 3: Servidor Erlangiano

Neste modelo estamos representando uma fonte *ONOFF* que envia pacotes para serem servidos por um servidor erlangiano. Para esta categoria de servidores, o pacote de dados é servido em R estágios distintos e consecutivos. Cada estágio possui um tempo de serviço exponencialmente distribuído. Se o número de estágios for grande o suficiente, estaremos aproximando o tempo de serviço por um tempo determinístico [16].

O comportamento do modelo é descrito a seguir. O período ativo da fonte é, em média, maior que o seu período de silêncio. Este tipo de fonte pode ser utilizado para representação de tráfego com períodos de rajada de pacotes. Além disso, uma outra característica é que a taxa de envio de pacotes é maior que a taxa de serviço. Este cenário foi proposto para que pudéssemos estudar o comportamento do *buffer* quando estamos sobrecarregando o sistema. Os parâmetros do modelo podem ser visualizados na tabela 3.6.

Para este estudo duas medidas foram calculadas: tamanho médio do *buffer* e proporção de tempo em que o *buffer* possui uma ocupação acima de 70% da sua ocupação total. Estas medidas, dentro da análise do modelo, podem parecer redundantes. No entanto, a medida de proporção de tempo acima de 70% de ocupação é extremamente atraente se estamos interessados, por exemplo, em promover algum tipo de controle de congestionamento em uma rede de dados sem estarmos interessados no número médio de pacotes que estão no *buffer*. Sem a utilização de recompensas, a obtenção deste tipo de medida seria bem mais complicada.

As recompensas foram assim definidas:

```
rate_reward = média_buffer
  condition = (TRUE)
  value     = buffer;

rate_reward = buffer_acima_35
  condition = (buffer>= 35)
  value     = 1;
```

O principal objetivo deste modelo é apresentar um estudo comparativo do erro entre a solução fornecida pelo método proposto e a solução fornecida pela técnica de uniformização.

Analisemos primeiramente a medida que calcula o tamanho médio do *buffer* em um instante t do tempo, sendo que $t \in [0s, 10s]$. Cada subintervalo de tempo observado foi aproximado através de 15 estágios exponenciais. A figura 3.9 mostra o gráfico com as curvas obtidas.

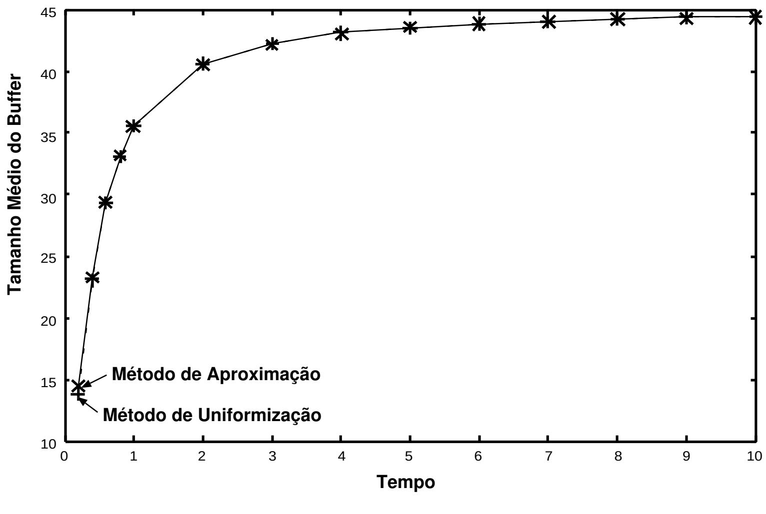


Figura 3.9: Tamanho médio do *buffer* em um instante de tempo t.

Observando o gráfico, podemos notar que o resultado obtido pelo método de aproximação possui uma margem de erro muito baixa em relação ao resultado da técnica de uniformização. Os valores para os erros absoluto e relativo são mostrados na tabela 3.7.

Tempo	Erro Absoluto	Erro Relativo
0.2	6.2142e-01	4.4845e-02
0.4	2.8907e-01	1.2502e-02
0.6	8.8030e-02	3.0020e-03
0.8	2.3037e-02	6.9500e-04
1	5.5850e-03	1.5700e-04
2	1.3970e-03	3.4000e-05
3	1.4760e-03	3.5000e-05
4	1.4480e-03	3.4000e-05
5	1.4480e-03	3.3000e-05
6	1.4410e-03	3.3000e-05
7	1.4430e-03	3.3000e-05
8	1.4330e-03	3.2000e-05
9	1.4260e-03	3.2000e-05
10	5.4470e-03	1.2200e-04

Tabela 3.7: Erros absoluto e relativo (tamanho médio do *buffer*) - método de aproximação.

Mesmo com um erro absoluto na ordem de 10^{-1} nos dois primeiros instantes de tempo, podemos dizer que este fato não compromete a eficácia do método, já que este erro representa um valor menor que 5% do valor real da medida. Além disso, podemos notar que para os demais intervalos de tempo, o método de aproximação possui um erro absoluto da ordem de 10^{-3}, chegando a um erro relativo da ordem de 10^{-5} em alguns intervalos de observação. Estes resultados comprovam a eficácia do método proposto.

Apesar dos erros absoluto e relativo diminuírem com o aumento do número de estágios exponenciais (conforme vimos no modelo do canal multiplexador), observamos na tabela 3.7 que, por exemplo, no tempo $t = 10s$ o erro absoluto é numericamente maior do que o erro produzido no tempo $t = 9s$. Este fato ocorre porque o erro relativo e absoluto estão relacionados tanto com aumento do número total de estágios quanto com a variação da medida nas proximidades do tempo em que estamos observando. No entanto, apesar da variação da medida, podemos dizer que o aumento de **R** tende a produzir, na grande maioria dos casos, resultados cada vez mais próximos dos resultados obtidos com a técnica de uniformização.

Passemos a análise da medida de interesse onde calculamos a proporção de tempo em que passamos com o *buffer* acima de 70% da sua capacidade total. O tempo t de observação está definido em $[0s, 10s]$. Cada subintervalo de tempo observado foi aproximado através de 20 estágios exponenciais. A figura 3.10 mostra o gráfico com as curvas obtidas.

Os resultados obtidos através do método proposto e através da técnica de uniformização podem ser visualizados no gráfico da figura 3.10. Mais uma vez podemos observar, com o auxílio da tabela 3.8, que os erros absoluto e relativo variam entre 10^{-2} e 10^{-5}, provando a eficácia do método de aproximação.

3.4 Estimativa do Erro do Método de Aproximação

Para exemplificar o cálculo da estimativa do erro do método de aproximação para a obtenção do valor esperado da recompensa de taxa acumulada, utilizamos a cadeia de

Tempo	Erro Absoluto	Erro Relativo
0.2	1.3669e-02	4.4845e-02
0.4	8.2340e-03	1.2502e-02
0.6	2.5060e-03	3.0020e-03
0.8	6.5100e-04	6.9500e-04
1	1.5900e-04	1.5700e-04
2	2.2000e-05	3.4000e-05
3	2.3000e-05	3.5000e-05
4	2.3000e-05	3.4000e-05
5	2.3000e-05	3.3000e-05
6	2.3000e-05	3.3000e-05
7	2.3000e-05	3.3000e-05
8	2.3000e-05	3.2000e-05
9	2.3000e-05	3.2000e-05
10	2.2000e-05	3.3000e-05

Tabela 3.8: Erros absoluto e relativo (proporção de tempo acima de 70%) - método de aproximação.

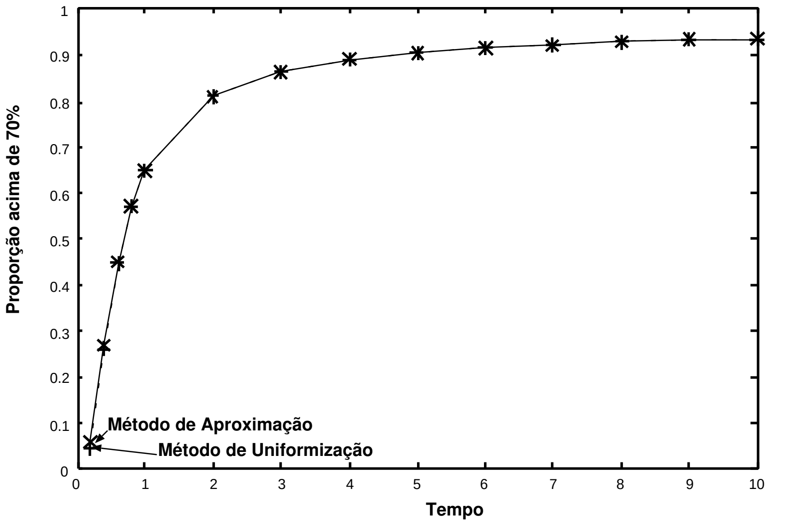


Figura 3.10: Proporção de tempo com o *buffer* acima de 70% da sua ocupação total.

Markov apresentada na figura 3.11. Esta cadeia representa os estágios de uma distribuição erlangiana, e a recompensa calculada foi a proporção de tempo que o processo passa nos cinqüenta primeiros estágios da distribuição, considerando como estado inicial cada um dos duzentos estados da cadeia. Para este exemplo foi utilizado o parâmetro R com valor igual a 1.

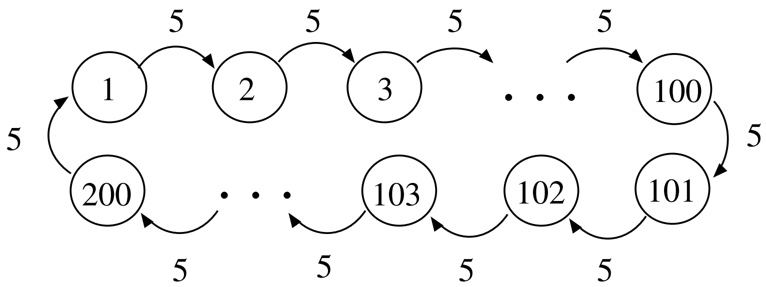


Figura 3.11: Cadeia de Markov Erlangiana.

O gráfico 3.12 apresenta os resultados obtidos. Nele, apresentamos o valor real da recompensa, obtido através da utilização do método de uniformização, o valor aproximado da recompensa, obtido através do método de aproximação proposto, o valor do erro estimado e o valor do erro absoluto entre as medidas. Podemos notar que o erro estimado possui comportamento similar ao erro real, ou seja, para valores de t onde a medida possui uma grande variabilidade o erro é maior. Assim, caso seja utilizado um valor de **R** que não proporcione uma boa aproximação para o tempo determinístico t, obteremos para

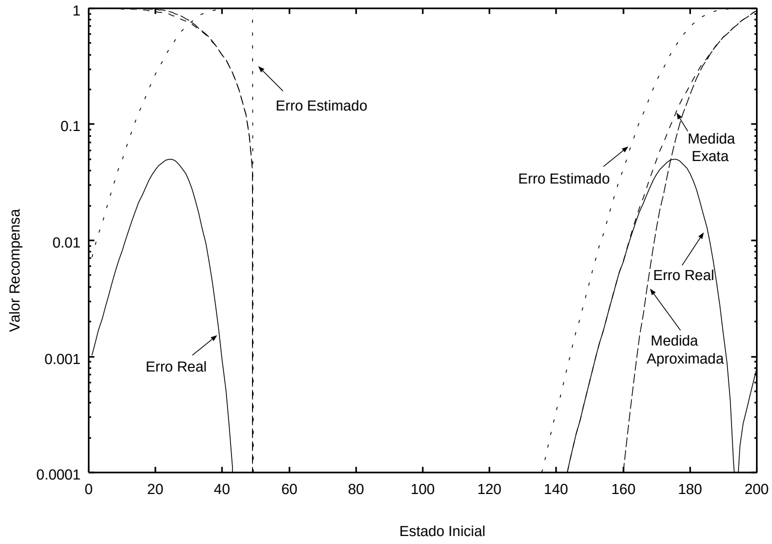


Figura 3.12: Erro Estimado para o Método de Aproximação.

o erro estimado um valor significativo. Dessa maneira, podemos recalcular a medida de interesse aumentando o valor de **R** e obter um resultado mais aproximado do resultado real da medida de interesse.

Capítulo 4

Método de Aproximação para o Cálculo da Distribuição da Ocupação do *Buffer* em Tempo Finito

4.1 Introdução

UMA das maiores restrições encontradas durante o processo de modelagem está relacionada à cardinalidade do espaço de estados que representa o modelo alvo de estudo. Assim, modelos que exibem um amplo conjunto de possíveis estados requerem métodos eficientes para obtenção do espaço de estados, bem como métodos matemáticos eficientes para a solução dos mesmos [9].

Consideremos, por exemplo, a obtenção da distribuição de probabilidade, em estado transiente, de um *buffer* alimentado por uma fonte de tráfego. Tradicionalmente esta medida é obtida através da especificação de um modelo onde tanto a fonte quanto o *buffer* são representados, o que faz com que a cardinalidade do espaço de estados seja diretamente proporcional ao tamanho do *buffer* e das diferentes fases da fonte. Após a especificação do modelo e a geração do modelo matemático, utilizamos um método de solução, como por exemplo a técnica de uniformização, para a obtenção da distribuição de probabilidade. Caso tenhamos uma fonte que possua cinco diferentes níveis de taxa de transmissão e um roteador com capacidade para 20.000 pacotes, o espaço de estados seria aproximadamente igual a 100.000. Este valor em conjunto com os parâmetros do modelo, como por exemplo altas taxas de transmissão, podem ocasionar um alto custo computacional para a obtenção de diversas medidas de interesse.

Uma maneira de amenizar o problema descrito acima é o estudo de métodos de solução que proporcionam a obtenção de diversas medidas de interesse sem que haja a descrição de todo o espaço de estados ou a realização de um grande número de operações. Estudos nesta direção podem ser encontrados em [6] e [15].

Neste capítulo mostramos como a distribuição transiente do *buffer* pode ser obtida usando o paradigma de recompensas de impulso juntamente com os conceitos de limites inferior e superior impostos a recompensa máxima acumulada. Desta forma, podemos modelar *buffers* com várias capacidades de armazenamento, sem nos preocuparmos com a modelagem do seu tamanho total. Um outra particularidade do método é que este pode ser utilizado para a resolução de modelos de fluido [24] de uma maneira bem mais eficiente. Os modelos de fluido se adequam perfeitamente às redes multimídia, onde a quantidade de informação transportada é bem elevada.

A organização deste capítulo é descrita a seguir. Na seção 4.2 apresentamos os principais conceitos de modelos de fluido e o mapeamento destes para os modelos Markovianos com recompensas. Na seção 4.3 apresentamos a utilização de limites para as recompensas atribuídas ao modelo. Na seção 4.4 comentamos brevemente o método proposto em [6, 15] para o cálculo da distribuição da recompensa de taxa acumulada, limitada superiormente e inferiormente. Na seção 4.5 apresentamos a equação proposta em [6] e a modelagem sugerida para obtenção da distribuição de probabilidade da ocupação do *buffer* em um tempo finito t. Calculamos o custo computacional do método e avaliamos em que condições ele apresenta ganho computacional em relação ao método baseado em recompensa de taxa. Na seção 4.6 apresenta alguns exemplos onde foi estudado a precisão do método e o seu custo computacional.

4.2 Modelos de Fluido

Consideremos o tráfego gerado pelas aplicações multimídia, tais como vídeo-conferência, ensino à distância, mundos virtuais entre outras. Este tipo de tráfego se caracteriza por uma grande quantidade de informação e uma grande variabilidade na taxa de transmissão [6].

Para o estudo do comportamento de um *buffer* alimentado pelo tipo de tráfego descrito acima, os modelos de fluido (*fluid flow models*) têm sido muito utilizados [7]. Neste caso, o fluxo de chegada e saída de pacotes em um roteador é aproximado por um fluido contínuo. Este tipo de modelo se adequa bem ao estudo da variação do tamanho do *buffer*, já que o tempo entre chegadas de informações é pequeno em relação à duração de uma rajada e o tempo de serviço e o tempo entre chegada de informações são constantes [6]. Dessa maneira, o tráfego passa a se comportar como um fluido contínuo de informação.

Sob estas condições, seja $b(t)$ a quantidade de pacotes armazenados no *buffer* associado a um canal de saída de um roteador, no instante de tempo t. Seja $X = \{X(t), t \geq 0\}$ o processo que modela a fonte que alimentam o canal, isto é, $X(t)$ fornece a taxa de transmissão no instante de tempo t. Supondo que X é uma cadeia de Markov, seja $P_{ij}(t,x) = P[X(t) = j, b(t) < x | X(0) = i]$. É possível obter a seguinte equação matricial diferencial parcial:

$$\frac{\partial P(t,x)}{\partial t} + \mathbf{D}\frac{\partial P(t,x)}{\partial x} = QP(t,x) \tag{4.1}$$

onde $\mathbf{D}$ é uma matriz diagonal cujo o i-ésimo elemento representa a taxa de transmissão no estado i do processo Markoviano, subtraída da capacidade do canal. Dessa maneira, este elemento representa o quanto de informação está sendo acrescida/decrementada do *buffer*.

A equação 4.1 pode ser resolvida usando análise espectral, envolvendo, também, a inversão numérica da transformada de Laplace de uma equação obtida de 4.1 (ver [6, 24, 19]). Já a solução em estado estacionário envolve o cálculo dos autovalores e autovetores de $\mathbf{D}^{-1}\mathbf{Q}$, o que não é uma tarefa trivial a não ser para casos particulares.

Não é difícil ver que o modelo de fluido pode ser imaginado como um modelo de recompensa. Suponha, como indicado acima, que o conjunto de fontes alimentando o *buffer* de um roteador seja modelado por uma cadeia de Markov $X = \{X(t), t \geq 0\}$, onde a taxa de transmissão (constante) no estado s_i é igual a λ_i (*bits*, células, pacotes, etc). Seja C a capacidade do canal. Assim, se associarmos uma recompensa de taxa $r_i = \lambda_i - C$ ao modelo, a recompensa de taxa acumulada $CR(t)$ é igual ao tamanho do *buffer* em t, desde que os valores de $CR(t)$ estejam **limitados** entre 0 e o tamanho máximo do *buffer*.

4.3 Introdução de Limites à Recompensa Acumulada

Conforme observamos na seção 4.2 podemos mapear o modelo de fluido, que tradicionalmente é resolvido através de métodos de transformada de Laplace ou cálculo de autovalores e autovetores, em modelos que utilizam recompensas que devem ter os seus possíveis valores limitados. No entanto, até o momento estudamos a aplicação de recompensas sem estarmos preocupados em estabelecer limites aos seus possíveis valores. Conseqüentemente, a recompensa acumulada pode atingir diversos valores, aumentando gradativamente, caso somente recompensas positivas estejam presentes no modelo, ou assumindo valores negativos, caso existam recompensas negativas.

Para que seja possível o mapeamento entre o modelo de fluido e o modelo de recompensa sem a modelagem do *buffer*, devemos estabelecer limites para que o *buffer* nunca tenha uma ocupação negativa ou uma ocupação acima da sua capacidade máxima. A figura 4.1 apresenta a utilização de limites à recompensa acumulada, quando consideramos o mesmo caminho para um determinado processo.

Conforme podemos ver, a recompensa acumulada passa a ter valores entre os limites estabelecidos. Conseqüentemente, podemos realizar o estudo do comportamento de um *buffer* sem a sua modelagem explícita, já que podemos garantir, através da utilização de limites, que o *buffer* nunca terá ocupação negativa ou acima do seu valor máximo. Além disso, a partir da introdução de limites, podemos resolver modelos de fluido de uma maneira bem mais simples e eficiente em termos computacionais, diferentemente

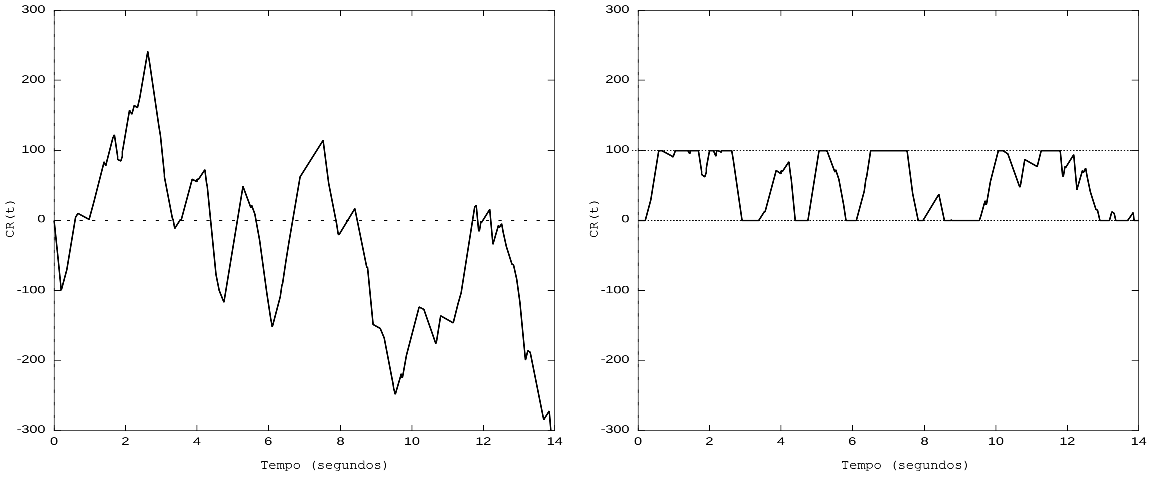


Figura 4.1: Diferença entre processos com recompensa ilimitada e limitada.

da maioria dos métodos propostos que envolvem resoluções de equações diferenciais, possuindo, assim, alta complexidade e problemas numéricos.

4.4 Modelos de Fluido e Recompensas de Taxa

Em [15] foi proposto um algoritmo para determinação da distribuição complementar da recompensa de taxa acumulada, isto é $P[CR(t) > x]$. Em [6] foi mostrado que com a utilização de argumentos probabilísticos, é possível limitar tanto inferiormente quanto superiormente a recompensa acumulada de taxa acumulada e com isso pode-se calcular a distribuição transiente da ocupação de um *buffer*.

Dois aspectos devem ser ressaltados em relação ao método de [6, 15]. O primeiro é relativo à restrição que deve ser satisfeita quando são impostos limites à variável aleatória $CR(t)$. A seguinte condição deve ser satisfeita neste caso [6]:

$$B \leq r^+ t \tag{4.2}$$

onde r^+ é a menor recompensa positiva do modelo. Isto significa que, para os modelos onde o tamanho do *buffer* é muito grande em relação a menor recompensa positiva e o tempo de observação t é pequeno, não conseguimos determinar a distribuição complementar exata da ocupação do *buffer*. Nestes casos, somente conseguimos valores que sejam limites inferior e superior para a medida original. Para estabelecermos o intervalo em que a medida verdadeira se encontra, devemos executar o algoritmo por duas vezes, aumentando seu custo computacional. Um outro problema é que nem sempre os limites inferior e superior estão próximos um do outro, não fornecendo uma boa precisão para a medida real [6].

Um outro ponto a ser ressaltado está relacionado a complexidade computacional do método. A ordem de complexidade é de $O(N^2dM^2)$, onde N é o ponto em que a série de *Poisson* é truncada, d é o número médio de saídas de um estado e M é o total de estados que representam a fonte. O custo computacional pode ser elevado nos casos em que N tem valor elevado, ou seja, modelos onde a taxa de uniformização Λ é alta ou que demoram para atingir o estado estacionário.

Na próxima seção apresentamos um método de solução que utiliza recompensa acumulada de impulso, limitada superiormente e inferiormente, para o cálculo da distribuição de ocupação de um *buffer* em um instante t. Este fato somente é possível através de uma aproximação à recompensa acumulada de taxa.

4.5 Modelos de Fluido e Recompensas de Impulso

Definimos modelos Markovianos com recompensas de impulso os modelos em que a uma dada transição particular ocorrida de um estado i para um estado j, uma recompensa $\rho(i, j)$ é atribuída ao processo.

Como a recompensa de impulso está disassociada do tempo, tradicionalmente esta é utilizada para contabilizar eventos que ocorrem durante o processo, já que são os eventos que desencadeiam as transições. No entanto, apresentamos nesta seção uma maneira de aplicarmos a recompensa acumulada de impulso para obtenção da distribuição da ocupação do *buffer* em um instante de tempo t.

4.5.1 Definição do Modelo usando Recompensa de Impulso

No modelo a ser utilizado iremos associar a cada transição uma recompensa de impulso que deve ser igual à variação do *buffer* em um curto intervalo de tempo t.

Consideremos o modelo definido anteriormente onde associa-se uma recompensa de taxa $r_i = \lambda_i - C$, sendo λ_i a taxa da fonte e C a capacidade do canal de comunicação. Neste modelo, se o processo permanece no estado i t unidades de tempo, o *buffer* aumenta de $(\lambda_i - C)t$ unidades de informação, caso $(\lambda_i - C)$ seja positiva. Caso $(\lambda_i - C)$ seja negativa, o *buffer* decresce de $(\lambda_i - C)t$ unidades de informação.

Considere o processo $M = \{M(t), t \geq 0\}$ da figura 4.2 que representa uma fonte de tráfego *ONOFF*. Neste modelo associamos uma recompensa de impulso $\rho(i, j)$ a uma transição do estado i para o estado j, sendo que esta deve ser igual a variação do *buffer* em um intervalo t.

Suponha agora que M tenha sido uniformizado com taxa Λ. Logo, o tempo de permanência em cada estado do modelo é exponencialmente distribuído com média $1/\Lambda$. No estado *ON* temos a transmissão de informação a uma taxa de $\lambda_i > C$. Neste mesmo estado, o canal, no qual as informações estão sendo enviadas, "serve" o tráfego com uma taxa C. Dessa forma, podemos dizer que o *buffer* está aumentando em média o seu tamanho de $(\lambda_i - C)/\Lambda$, a cada visita ao estado *ON*. Este valor deve ser atribuído a recompensa de impulso $\rho(ON, OFF)$. No caso do estado *OFF*, o tamanho do *buffer* decresce, em média, de $-C/\Lambda$. Assim, este valor deve ser atribuído a recompensa de impulso $\rho(OFF, ON)$. Podemos observar que o comportamento do *buffer* é totalmente representado no modelo descrito acima, o que possibilita a diminuição do espaço de estados do modelo e, conseqüentemente, a diminuição, na grande maioria dos casos, do custo computacional.

Generalizando o exemplo descrito acima, considere um processo $M = \{M(t), t \geq 0\}$, uniformizado, que representa uma fonte de tráfego. Para cada transição de um estado i do modelo para um estado j, associa-se uma recompensa de impulso $\rho(i, j) = (\lambda_i - C)/\Lambda$, onde λ_i é a taxa da fonte no estado i, C é a capacidade do canal e Λ é a taxa de uniformização do processo.

Desta forma a variável aleatória $CI(t)$, recompensa de impulso acumulada em $(0, t)$, representa o tamanho do *buffer* no instante de tempo t.

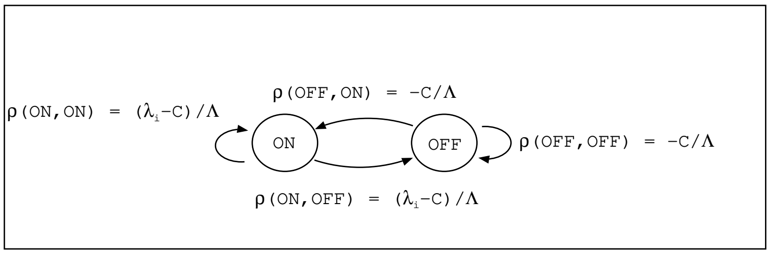


Figura 4.2: Representação da fonte *ONOFF* para modelos com recompensa de impulso.

4.5.2 A Modelagem Proposta

Em [8] é apresentada uma equação para o cálculo de distribuição da recompensa acumulada de impulso. No trabalho apresentado em [6] é mostrado que a recompensa pode ser limitada inferiormente e superiormente de forma simplificada.

Limitando a variável aleatória $CI(t)$ e utilizando o modelo descrito na seção 4.5.1, é possível calcularmos a distribuição da ocupação de um *buffer* em um instante de tempo t, sem nos preocuparmos com a modelagem de sua capacidade. Para obtermos a distribuição de $CI(t)$, condicionamos a $P[CI(t) \leq r]$ no número de transições n ocorridas no processo M e no valor da recompensa acumulada $\hat{r}$. Desta forma, obtemos [8]:

$$P[CI(t) \leq r] = \sum_{n=0}^{\infty} e^{-\Lambda t} \frac{(\Lambda t)^n}{n!} \sum_{\hat{r}} \hat{\Gamma}[n, \hat{r}] P[CI(t) \leq r | n, \hat{r}] \tag{4.3}$$

onde $\hat{\Gamma}[n, \hat{r}] = P[\hat{r}|n \text{ transições}]$. O termo de *Poisson* da equação 4.3 é facilmente obtido [10], restando determinar $P[CI(t) \leq r|n, \hat{r}]$ e $\hat{\Gamma}[n, \hat{r}]$. Claramente temos que:

$$P[CI(t) \leq r|n, \hat{r}] = \begin{cases} 1 & \text{, se } \hat{r} \leq r \\ 0 & \text{, caso contrário} \end{cases} \tag{4.4}$$

Desta forma, podemos reescrever 4.3:

$$P[CI(t) \leq r] = \sum_{n=0}^{\infty} e^{-\Lambda t} \frac{(\Lambda t)^n}{n!} \sum_{\hat{r} \leq r} \hat{\Gamma}[n, \hat{r}] \tag{4.5}$$

Consideremos $\hat{\Gamma}[n, \hat{r}]$. Seja $\hat{\Gamma}_s[n, \hat{r}]$ a probabilidade da recompensa total acumulada ser igual a $\hat{r}$ dado a ocorrência de *n* transições e que o estado visitado após a última transição seja o estado *s*. Podemos calcular $\hat{\Gamma}_s[n, \hat{r}]$ através da seguinte recursão, sabendo que $l.i$ e $l.s$ representam, respectivamente, os limites inferior e superior impostos ao valor da recompensa acumulada de impulso [6]:

$$\hat{\Gamma}_s[n, \hat{r}] = \begin{cases} \displaystyle\sum_{s' \in \mathcal{S}, \rho(s',s) \leq 0} \; \sum_{r' \leq l.i. - \rho(s',s)} \hat{\Gamma}_{s'}[n-1, r'] p_{s's} & \text{, se } \hat{r} = l.i. \\ \displaystyle\sum_{s' \in \mathcal{S}} \hat{\Gamma}_{s'}[n-1, \hat{r} - \rho(s', s)] p_{s's} & \text{, se } l.i. < \hat{r} < l.s. \\ \displaystyle\sum_{s' \in \mathcal{S}, \rho(s',s) \geq 0} \; \sum_{r' \geq l.s. - \rho(s',s)} \hat{\Gamma}_{s'}[n-1, r'] p_{s's} & \text{, se } \hat{r} = l.s. \end{cases} \tag{4.6}$$

onde *s'* é o estado visitado pelo processo antes da última transição, *r'* é a recompensa acumulada anteriormente a última transição, $\rho(s', s)$ é a recompensa de impulso relacionada com a transição do estado *s'* para o estado *s* e $p_{s's}$ representa a probabilidade de transição de *s'* para *s*. Esta recursão deve ser aplicada para valores de n maiores que 1.

Para iniciar a recursão:

$$\hat{\Gamma}_s[0, r] = \begin{cases} \pi_s(0) & \text{, se } r \text{ é igual a recompensa inicial} \\ 0 & \text{, caso contrário} \end{cases} \tag{4.7}$$

onde $\pi_s(0)$ é a probalidade do processo se encontrar inicialmente no estado *s*.

Para finalizar, definimos o vetor $\Upsilon[n, r] = (\Upsilon_1, ..., \Upsilon_N)$, onde $\Upsilon_i = \sum_{\hat{r} \leq r} \hat{\Gamma}_i[n, \hat{r}]$, considerando para cada estado do modelo a probabilidade de acumularmos todos os possíveis valores de recompensa menores que o valor r. Dessa forma, podemos estabelecer o seguinte teorema [6]:

Teorema 1 *A distribuição da recompensa acumulada com limites durante (0,t) é dada por*

$$P[CI(t) \leq r] = \sum_{n=0}^{\infty} e^{-\Lambda t} \frac{(\Lambda t)^n}{n!} \|\Upsilon[n, r]\| \tag{4.8}$$

sendo $\|\Upsilon[n, r]\|$ a soma de todos os elementos do vetor $\Upsilon[n, r]$.

Algumas considerações podem ser feitas com relação a recursão 4.6, principalmente quando a recompensa $\hat{r}$ atinge os valores do limite inferior e do limite superior:

1. **Limite Inferior** - Para que a recompensa $\hat{r}$ atinja o valor do limite inferior na n-ésima transição, o valor da recompensa r' acumulada até a transição $n-1$ deve assumir valores maiores que $l.i.$ e a recompensa recebida através da transição do estado s' para o estado s deve ser, obrigatoriamente, menor que zero. Este fato explica a restrição $\rho(s', s) \leq 0$ do primeiro somatório e a restrição $r' \leq l.i. - \rho(s', s)$ imposta ao segundo somatório.

2. **Limite Superior** - Da mesma maneira, para que a recompensa $\hat{r}$ atinja o valor do limite superior na n-ésima transição, o valor da recompensa r' acumulada até a transição $n-1$ deve assumir valores menores que $l.s.$ e a recompensa recebida através da transição do estado s' para o estado s deve ser, obrigatoriamente, maior que zero. Este fato explica a restrição $\rho(s', s) \geq 0$ do primeiro somatório e a restrição $r' \geq l.s. - \rho(s', s)$ imposta ao segundo somatório.

Para completarmos a compreensão da recursão 4.6, consideremos, como exemplo, a cadeia hipotética mostrada na figura 4.3. Nesta, ocorrem recompensas de impulso que assumem valores positivos, negativos e nulos. Para esse processo, limites superior e inferior para o total da recompensa acumulada pelo processo são estabelecidos. Podemos notar que os únicos valores que precisam ser mantidos durante a recursão são aqueles que se encontram entre os limites estabelecidos. Caso a recompensa acumulada $\hat{r}$ atinja valores fora desses limites, esta é descartada.

Observando a recursão 4.6 podemos verificar que um dos parâmetros a ser determinado são os possíveis níveis *r'*.

Considere o modelo da figura 4.4. Neste modelo os possíveis valores de recompensa

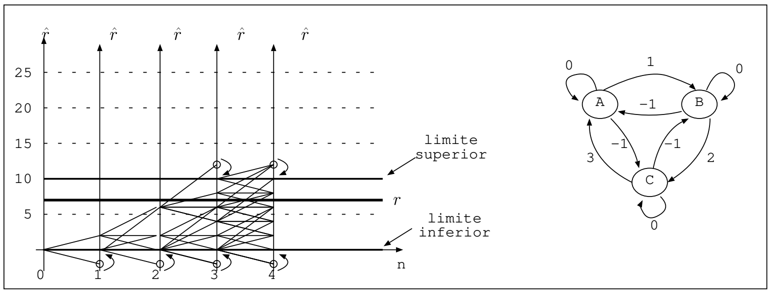


Figura 4.3: Desenvolvimento da recursão 4.6.

de impulso que podem ser adicionados/subtraídos da recompensa acumulada cada vez que o processo faz uma transição são $\rho(i,j) = \{(\lambda_1 - C)/\Lambda, (\lambda_2 - C)/\Lambda, (\lambda_3 - C)/\Lambda\}$.

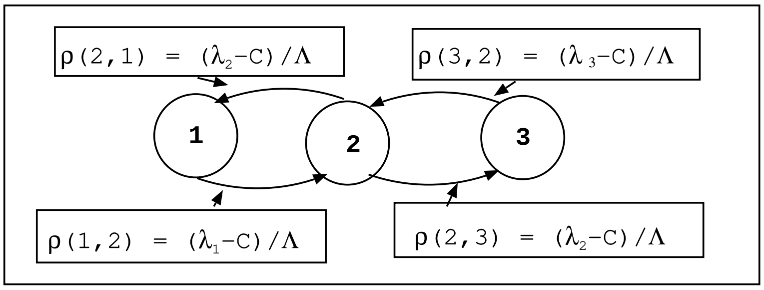


Figura 4.4: Modelo de fonte com três estados.

Dois casos devem ser considerados:

1. $\lambda_i - C >> \Lambda$ $(i = 1, 2, 3...)$

   Este é o caso dos modelos de fluido onde a taxa da fonte e a taxa de serviço são muito maiores que as taxas de transição entre os estados da fonte. Assim, $(\lambda_i - C)/\Lambda$ é maior do que uma unidade de informação o que significa que o *buffer* poderá ser acrescido/decrementado de valores elevados. Logo, o custo computacional deve ser reduzido sensivelmente.

2. $\lambda_i - C < \Lambda$ $(i = 1, 2, 3...)$

   Para esta classe de modelos, $(\lambda_i - C)/\Lambda$ é menor do que uma unidade de informação. Logo, não devemos ter ganho computacional em relação aos modelos em que o *buffer* é explicitamente representado.

Consideremos, agora, como obter os possíveis valores de *r'*, baseado no modelo apresentado na figura 4.4. Denominamos granularidade o valor mínimo que pode ser adicionado/subtraído da recompensa acumulada cada vez que o processo faz uma transição. Desta forma, todos os possíveis valores de *r'* serão múltiplos da granularidade escolhida.

Para que possamos determinar a granularidade, os seguintes procedimentos podem ser utilizados:

1. Podemos fazer com que a granularidade seja igual ao mínimo entre todas as recompensas de impulso atribuídas ao processo.

   No caso do modelo da figura 4.4 a granularidade é $G = min(\quad |(\lambda_1 - C)/\Lambda|, |(\lambda_2 - C)/\Lambda|, |(\lambda_3 - C)/\Lambda|)$. Se o tamanho do *buffer* a ser modelado é igual a B, teremos que o total de níveis R' é igual a B/G.

2. Podemos fazer com que a granularidade seja igual ao máximo divisor comum das recompensas recebidas pelo processo. Neste caso, estaremos aumentando o número total de níveis *r'* e estabelecendo um universo maior de possíveis múltiplos pelos quais as recompensas reais devem ser aproximadas. Este procedimento faz com que o resultado calculado fique bem próximo ao resultado real da medida. O custo computacional atinge valores proporcionais ao tamanho do *buffer* e do máximo divisor comum das recompensas associadas ao processo.

   No caso do modelo da figura 4.4 a granularidade é $G = MDC(\quad |(\lambda_1 - C)/\Lambda|, |(\lambda_2 - C)/\Lambda|, |(\lambda_3 - C)/\Lambda|)$. Se o tamanho do *buffer* a ser modelado é igual a B, teremos que o total de níveis R' é igual a B/G.

3. Podemos fazer com que a granularidade seja igual a média ponderada das diversas recompensas pela proporção do tempo que o processo permanece em cada fase da fonte, considerando o instante t de observação. A média ponderada é obtida através da multiplicação entre as recompensas associadas a cada fase da fonte e a proporção de tempo em que o processo permanece em cada fase da fonte. Para se obter o segundo fator da multiplicação, devemos definir, para cada fase da fonte, uma recompensa de taxa onde atribuimos o valor 1 quando estamos na fase considerada e 0, caso contrário. Ao final do tempo de observação t, temos o tempo total passado em cada fase da fonte e a divisão deste valor pelo tempo t resulta na proporção de tempo procurada. Este resultado pode ser obtido através do método

de aproximação apresentado no capítulo 2. O fato de utilizarmos este artifício para o cálculo da proporção de tempo em cada fase da fonte requer um custo computacional bastante pequeno, devido ao espaço de estados reduzido do modelo da fonte. Se o tamanho do *buffer* a ser modelado é igual a B, teremos que o total de níveis R' é igual a B/G.

A partir dos resultados obtidos, duas situações podem ocorrer:

1. $G \approx B$

   Neste caso, a granularidade é proporcional ao tamanho do *buffer* (mesma ordem de grandeza). O fato da granularidade assumir valores maiores é um indicativo de que não será necessário um grande número de níveis de recompensa. No entanto, um novo valor de Λ deve ser calculado para que o crescimento/decrescimento do *buffer* assuma um comportamento mais contínuo. Este novo valor de Λ deve ser, por exemplo, tal que $(\lambda_i - C)/\Lambda' \approx 0.1B \; \forall i$.

   Assim, consideremos um processo onde as recompensas $\lambda_i - C$ assumam valores da ordem de grandeza 10^3. Se o tamanho do *buffer* é da ordem de grandeza de 10^3, por exemplo igual a 5000, podemos escolher um novo Λ', tal que o crescimento/decrescimento seja da ordem de 10^2. A partir dessas condições, Λ' será da ordem de 10^1, sendo o seu novo valor igual, por exemplo, a 50, que representa $0.01B$.

2. $G << B$

   Neste caso, o valor do Λ da uniformização pode ser mantido.

As considerações citadas acima se mostraram eficientes para determinação do número total de níveis R' para um conjunto de modelos estudado.

4.5.3 Custo Computacional

Para o cálculo do custo computacional do método proposto devemos observar que o somatório em 4.8 terá um total de elementos igual ao N utilizado para truncar a série infinita. A cada transição do processo, devemos calcular a probabilidade da recompensa acumulada ser menor que um valor r qualquer, representado pelo termo $\Upsilon[n, r]$. Para a obtenção deste termo, devemos considerar todos os possíveis estados do processo como sendo o último estado visitado após a n-ésima transição. Dessa forma para todo $\rho(s', s)$ diferente de zero, devemos calcular o termo $\Gamma[n, \hat{r}]$, considerando todos os possíveis níveis de recompensa acumulada. Assim, o custo do método proposto é da ordem de $O(NdMR')$, onde N é o ponto onde o termo de *Poisson* é truncado, d é o número médio de transições ocorridas a partir de um estado qualquer do processo, M é o número de estados da fonte modelada e R' é o número total de níveis de recompensa. O termo dM pode ser considerado como o número total de entradas não-nulas da matriz de probabilidades.

Nos casos em que o *buffer* é modelado, a cadeia de Markov relacionada com o modelo possui $M(B+1)$ estados, onde M é o total de fases da fonte e B representa o tamanho do *buffer*. Como já citado anteriormente (ver capítulo 2), o custo da resolução deste modelo usando, por exemplo, a uniformização é da ordem de $O(NdM(B+1))$. Assim, o custo computacional do método proposto será menor quando $R' < B$. Esta condição não é difícil de ser alcançada, já que para os modelos de fluido, o número total de níveis de recompensa é, em geral, bem menor que o tamanho B do *buffer*.

Como foi citado na seção 4.4, o custo computacional do método que utiliza recompensas de taxa para calcular a distribuição da ocupação do *buffer* é da ordem de $O(N^2dM^2)$. Para a comparação com o método de impulsos, devemos considerar dois casos específicos:

1. Nos casos onde um novo valor Λ' deve ser considerado para o cálculo de G, temos que o método de recompensa de impulso será mais vantajoso se e somente se $NR' < N^2M$. O valor do parâmetro N passa a ter valores diferentes para os dois métodos fato este decorrente dos valores distintos do parâmetro Λ.

2. Nos casos onde é usado o Λ da uniformização para o cálculo de G, temos que o método de impulsos será mais vantajoso se e somente se $R' < NM$. O ganho computacional é imediato, caso o valor de N seja elevado. Este fato ocorre quando o Λ de uniformização possui valor elevado ou quando o modelo atinge o estado estacionário para valores elevados de t, já que $N \approx \Lambda t$.

O fato de estarmos aplicando este método para resolução de modelos onde as taxas de recompensas são bem mais elevadas que as taxas de transição entre os estados torna possível estabelecermos valores de R' bem menores que o tamanho real do *buffer*, produzindo, além de uma boa aproximação para o resultado, um custo computacional reduzido.

4.6 Exemplos

Esta seção apresenta resultados numéricos da utilização do algoritmo de recompensa de impulso com limites para o cálculo da distribuição transiente do *buffer*. Para tal, três exemplos de modelagem de tráfego são utilizados.

No primeiro exemplo temos a modelagem de uma fonte *ONOFF* que alimenta um *buffer*. Os demais exemplos representam fontes de dados que modelam seqüências reais de vídeos e que serão utilizados para o estudo do custo computacional, bem como do erro alcançado quando o método de recompensa de impulso é utilizado. Em todos os exemplos o erro foi estimado em relação ao método de recompensas de taxa encontrado em [6, 15].

4.6.1 Fonte *ONOFF*

Considere uma fonte *ONOFF* que alimenta um *buffer* de capacidade igual a 20.000. O tempo de permanência no estado *ON* e no estado *OFF* são exponencialmente distribuídos com média igual a $8ms$ e a $16ms$, respectivamente. A taxa de transmissão da fonte, no estado *ON*, é de $2.621.440$bits/s e a capacidade do canal de comunicação é de $1.310.720$bits/s.

O principal objetivo deste exemplo é avaliar o erro obtido quando variamos o número total de níveis de recompensa. Para tal, estabelecemos o instante $t = 1s$ e variamos o número total de níveis de recompensa entre 10 e 100. A definição da recompensa de impulso, utilizando a sintaxe da ferramenta TANGRAM-II é a seguinte:

```
impulse_reward = buffer
```

```
bounds= 0, 20000
  event = On_Off,1
  value = taxa_transmissão - taxa_serviço;
  event = Off_On,1
  value = - taxa_serviço;
```

A figura 4.5 mostra o comportamento do erro absoluto e do erro relativo, resultante da variação do número total de níveis de recompensa, para o cálculo da probabilidade do *buffer* atingir o seu valor máximo. Conforme esperado, o aumento do número de níveis possibilita uma maior precisão da medida resultante. Podemos observar que o erro absoluto sempre se encontra abaixo de 5% da medida real, chegando a quase 1% para R' igual a 100. Mesmo com um número reduzido de níveis, o método de recompensas de impulso proporciona uma boa aproximação em relação a solução exata (resultante da utilização do método de recompensa de taxa).

A razão de um número reduzido de níveis proporcionar uma boa aproximação se deve ao fato da fonte modelada possuir taxas de envio e serviço de informações bem maiores que as taxas de mudança entre os estados *ON* e *OFF*, situação que representa uma típica fonte de fluido, favorecendo, assim, a utilização do algoritmo de recompensas de impulso.

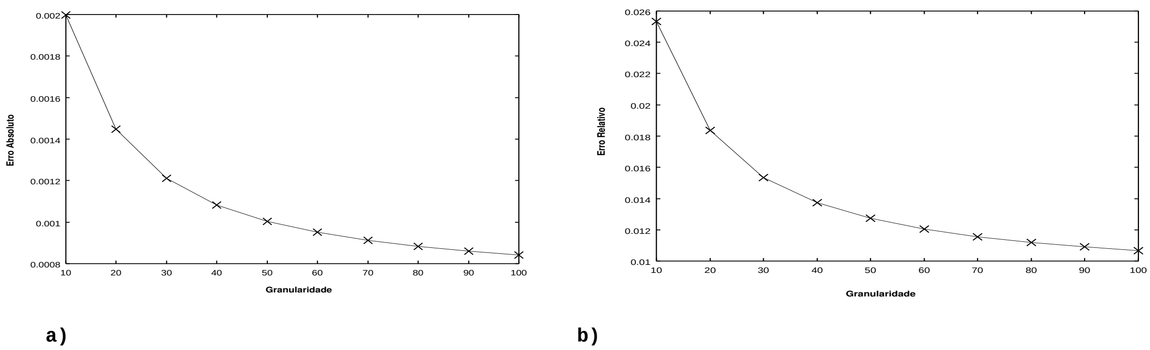


Figura 4.5: Variação do número total de níveis de recompensa: a) Erro absoluto; b) Erro relativo.

Estado	Taxa
1	18.293
2	15.882
3	13.482
4	11.072
5	8.672
6	6.261
7	3.861
8	1.450

Tabela 4.1: Parâmetros - Modelagem de tráfego - seqüência I.

4.6.2 Modelagem de Tráfego - Seqüência I

Para este exemplo de modelagem de tráfego utilizamos o trace de uma partida de futebol, disponibilizada em [20]. Esta seqüência está codificada em MPEG-I e para a modelagem da fonte de tráfego todos os três tipos de quadros (I,P,B) foram utilizados. Além disso, consideramos que o tráfego total é, na verdade, um agregado de 40 seqüências individuais, que, hipoteticamente estão sincronizadas no tempo. Assim, o tamanho de cada frame da seqüência original passa a ser multiplicado por 40, representando a transmissão simultânea de todas as seqüências individuais. No próximo exemplo é considerado o caso da não sincronização no tempo da apresentação de uma seqüência de filmes agregados.

O modelo de fonte de tráfego utilizado para representar a seqüência é o apresentado em [23]. A parametrização foi feita confome sugerido em [23], onde os valores para as taxas de transmissão em cada estado da fonte são apresentados na tabela 4.1. As taxas estão em pacotes, onde cada pacote tem 48 *bytes*. A capacidade considerada para o canal de comunicação é de 10.000 pacotes/s.

Cenário I - *Buffer* com Capacidade de 5.000 pacotes

Em um primeiro cenário, foi considerado um *buffer* com capacidade total de $B = 5.000$ pacotes, cada um com o tamanho igual a 48 *bytes*. Primeiramente, obtivemos a distribuição complementar da ocupação do *buffer* para o instante de tempo $t = 5s$. Em um segundo teste, variamos o intervalo de observação de $5s$ a $15s$ e obtivemos a probabilidade da ocupação do *buffer* ser maior que 90% da sua ocupação total.

Em relação ao parâmetro total de níveis de recompensa (R'), os seguintes valores foram utilizados:

1. Para um primeiro valor foi utilizado o cálculo da média ponderada entre as diversas taxas de transmissão de dados. Neste caso, calculamos a proporção de tempo em cada fase da fonte e obtivemos o valor da média ponderada, utilizando, primeiramente, o Λ da uniformização ($\Lambda = 1$). No entanto, este valor era da mesma ordem de grandeza do tamanho do *buffer*, fato este que não proporciona uma boa aproximação para o resultado obtido pelo método de recompensa de impulso. Assim, um novo valor Λ' foi calculado conforme apresentado na seção 4.5.2. Neste caso, para o valor de $\Lambda' = 50$, a média ponderada foi de 129 (valor do parâmetro G) e o total de níveis R' foi de 39;

2. Como segundo valor foi considerado o parâmetro G como sendo igual a divisão da recompensa mínima (1072) por $\Lambda = 50$. Neste caso, o total de níveis R' foi de 233;

3. Finalmente atribuímos um valor aleatório a R' (igual a 35) menor do que o calculado em 1.

A utilização de diversos valores para o parâmetro G tem como finalidade realizar um estudo comparativo entre as diversas propostas mostradas na seção 4.5.2.

O gráfico 4.6 mostra a distribuição complementar da ocupação do *buffer* para o instante de tempo $t = 5s$, utilizando os métodos de recompensa de taxa e de impulso. Conforme podemos observar, a utilização do método de recompensa de impulso para a determinação da distribuição complementar oferece uma boa aproximação para o método de recompensa de taxa. Além disso, podemos também observar que o fato de aumentarmos o total de níveis resulta em uma melhor aproximação em relação a solução exata.

O erro absoluto da medida variou entre $[0.002, 0.0008]$, considerando todos os níveis apresentados. Assim, o erro máximo alcançado foi menor que 10% da medida original, chegando aproximadamente a 5% da medida original para a probabilidade do *buffer* estar cheio, fato este que reafirma a eficácia do método proposto.

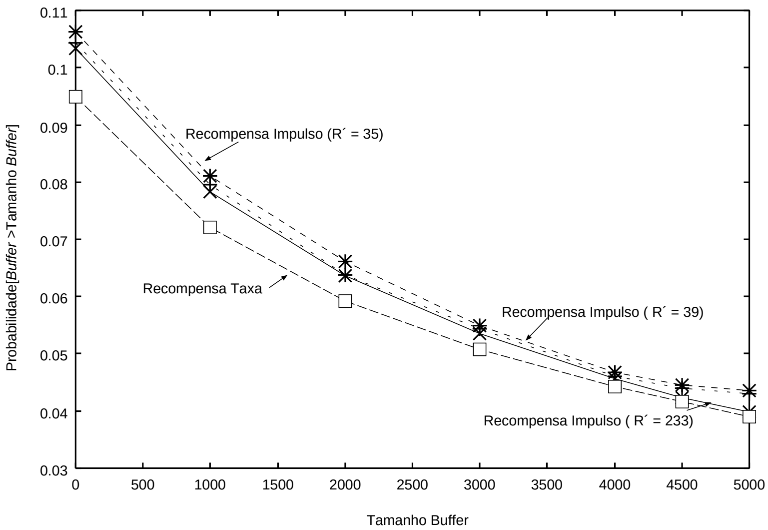


Figura 4.6: Distribuição complementar da ocupação do *buffer* no instante $t = 5s$.

Para uma segunda análise do método de recompensa de impulso, foi considerado o cálculo da probabilidade da ocupação do *buffer* ser maior que 90% do seu tamanho total, em diversos instantes de tempo. A figura 4.7 mostra os resultados obtidos.

No gráfico 4.7 podemos observar que a probabilidade do *buffer* ultrapassar 90% da sua ocupação diminui com o tempo, aproximando-se do valor de $3.352444e^{-3}$ (para R'= 35), $3.371964e^{-3}$ (para R'= 39) e $3.88856804e^{-3}$ considerando o método de recompensa de taxa, quando em estado estacionário (o que ocorre aproximadamente no tempo $t = 15s$). Mais uma vez, o método de recompensas de impulso obteve resultados bem próximos ao resultado exato fornecido pelo método de recompensas de taxa, fato este confirmado pela quase sobreposição das curvas no gráfico 4.7.

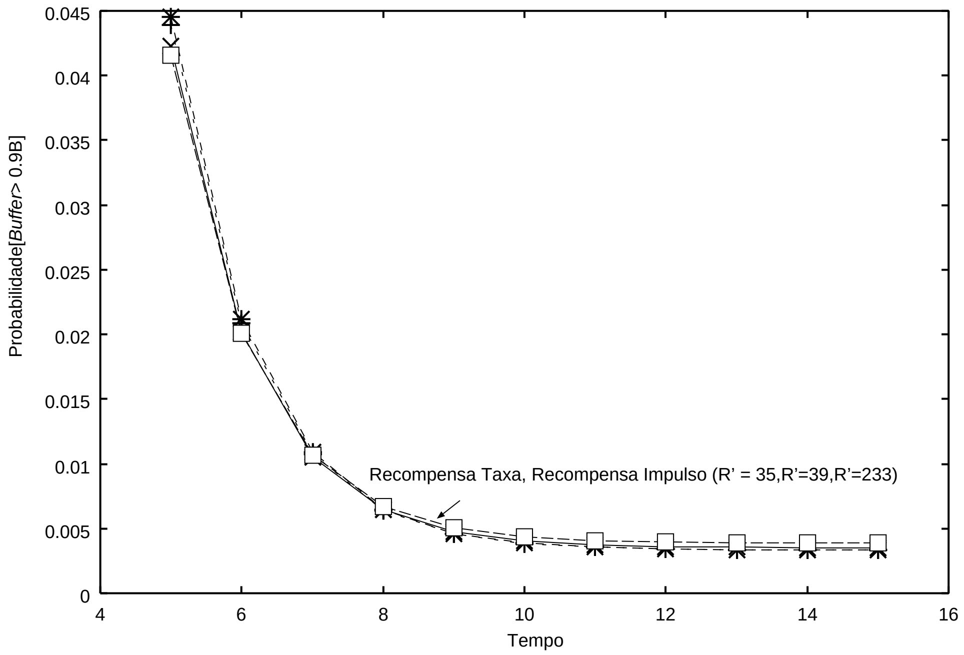


Figura 4.7: Probabilidade [*Buffer* $> 0.9B$].

Custo Computacional

Para finalizar a análise deste primeiro cenário, a figura 4.8 mostra o gráfico com o custo computacional, expresso em número total de multiplicações, dos métodos de recompensa de impulso e de taxa. O gráfico a) apresenta o custo computacional considerando $R' = 35$ e $R' = 39$. Podemos observar que o custo computacional é até cinco vezes menor no caso da utilização do método de recompensa de impulso. Além disso, a variação do custo do método de recompensa de taxa é exponencial em relação ao parâmetro N, enquanto que para o método de recompensa de impulso essa variação é linear.

O gráfico b) da figura 4.8 apresenta o custo computacional para o caso em que o total de níveis R' é igual a 233. Para este valor de R', o custo do método de recompensas de impulso atinge valores maiores que o método de recompensas de taxa, proporcionando, no entanto, uma solução mais aproximada da solução exata.

Em relação ao método de uniformização, o ganho computacional é imediato. Esta afirmação se deve por dois motivos. O primeiro é que a taxa de uniformização, Λ, que influencia diretamente no valor final de N, é proporcional as taxas do modelo. Assim, altas taxas de transmissão de informação (uma das características principais do modelo

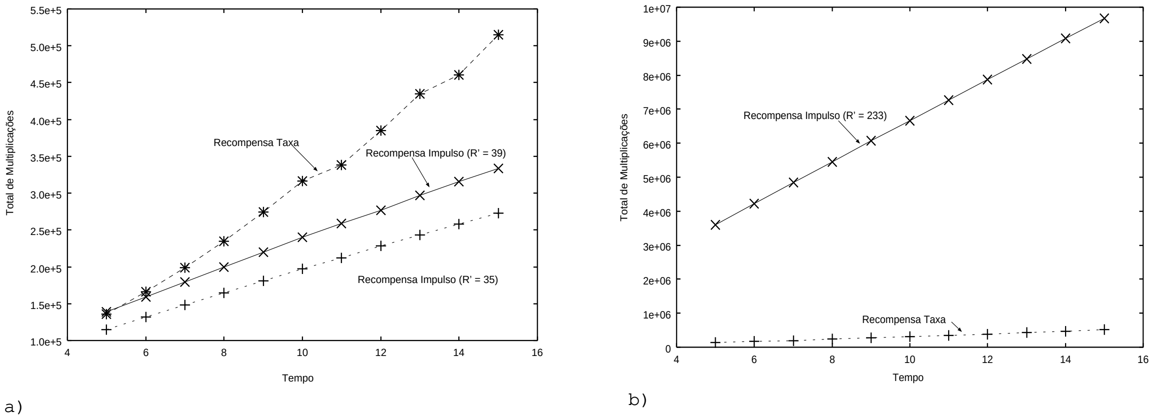


Figura 4.8: Custo computacional do método de recompensa de taxa e impulso: a) Recompensa de taxa e recompensa de impulso para $R' = 35$ e $R' = 39$; b) Recompensa de taxa e recompensa de impulso para $R' = 233$.

de fluido) resultam em altas taxas de uniformização. O segundo ponto é que o fato de modelarmos o *buffer* gera uma matriz de probabilidade com diversas entradas não-nulas, influenciando diretamente no custo computacional do método. Para este exemplo em específico, caso o *buffer* seja modelado, a taxa Λ seria aproximadamente igual a 28.323, enquanto o número de entradas não nulas da matriz **P** seria igual a 335.051. Estes parâmetros resultam em alto custo computacional, da ordem de 10^9, para o método de uniformização.

Finalizando, vale ressaltar que o custo da obtenção da medida para um determinado t' menor que o tempo final de observação t é o custo para o cálculo de t, e não a soma do custo em t' e t.

Cenário II - *Buffer* com capacidade de 30.000 pacotes

Em um segundo cenário, iremos considerar um *buffer* com capacidade bastante grande de forma que a probabilidade do *buffer* atingir a sua capacidade total seja bem pequena. Desta forma podemos verificar a precisão do método para valores pequenos de probabilidade. Assim, consideramos um *buffer* com capacidade total de 30.000 pacotes de tamanho igual a 48 *bytes* cada e calculamos a probabilidade do *buffer* atingir a sua capacidade total no instante de tempo $t = 15s$. Neste caso o método de recompensa de taxa teve que ser executado por duas vezes, em razão das restrições do próprio algoritmo. O resultado da me-

Estado	Taxa
1	13.435
2	11.677
3	9.906
4	8.141
5	6.377
6	4.612
7	2.848
8	1.083

Tabela 4.2: Parâmetros - Modelagem de tráfego - seqüência II.

dida, pelo algoritmo de recompensa de taxa, varia entre $(2.56899085e^{-5}, 2.62694243e^{-5})$. Neste caso, os limites são bem próximos. O resultado da medida, pelo algoritmo de recompensa de impulso foi de $2.607634e^{-5}$, o que mostra a eficácia do método, mesmo para valores pequenos de probabilidade. Em relação aos custos, o custo do método de recompensa de taxa ficou em $1.029.488$ multiplicações, enquanto o custo do método de recompensa de impulso ficou em 230.300 multiplicações (havendo redução de uma ordem de grandeza).

4.6.3 Modelagem de Tráfego - Seqüência II

Como uma segunda aplicação real para a utilização do método de recompensas de impulso, modelamos um agregado de filmes codificados em MJPEG e que podem ser encontrados em [3, 4]. Neste segundo agregado, diferentemente do primeiro apresentado na seção 4.6.2, foi considerado a não-sincronização temporal da demonstração dos filmes. Deste modo, os filmes começam em pontos aleatórios do tempo, representando melhor sistemas como vídeo-sob-demanda, por exemplo. Como fonte modeladora do tráfego utilizamos novamente a fonte apresentada em [23]. Os parâmetros da fonte são apresentados na tabela 4.2. As taxas são em pacotes por segundo, onde cada pacote tem 48 *bytes*. A capacidade do canal de comunicação é de 10.000 pacotes/s.

Para a determinação da probabilidade do *buffer* estar com a sua ocupação total, foi considerado um *buffer* com capacidade $B = 10.000$ pacotes. O intervalo total de observação considerado foi de $[10s, 3.000s]$. Este amplo intervalo é justificado pelo fato do modelo demorar a atingir o estado estacionário. Comparações entre o método de recompensas de impulso e o método de uniformização foram realizadas.

Para o valor do parâmetro total de níveis de recompensa, foram consideradas três situações:

1. Para um primeiro valor foi utilizado o cálculo da média ponderada entre as diversas taxas de transmissão de dados. Neste caso, calculamos a proporção de tempo em cada fase da fonte e obtivemos o valor da média ponderada, utilizando, primeiramente, o Λ da uniformização ($\Lambda = 0.04$). Este valor de Λ resultou em uma média ponderada elevada, sendo necessário o cálculo de um novo valor Λ' que proporcione um crescimento/decrescimento mais contínuo do *buffer*. Assim, um novo valor Λ' foi calculado conforme apresentado na seção 4.5.2. Neste caso, para o valor de $\Lambda' = 10$, considerando que as recompensas atribuídas sejam da ordem de 10^2, a média ponderada foi de 308 (valor do parâmetro G) e o total de níveis R' foi de 32;
2. Em um segundo caso consideramos o parâmetro G como sendo igual a divisão da recompensa mínima (94) pelo $\Lambda' = 10$. Neste caso, o total de níveis R' foi de 106;
3. Finalmente atribuímos aleatoriamente um valor para R' como sendo igual a 30, que é inferior ao valor definido em 1.

Comparação I - Recompensas de Taxa e Impulso

Consideremos o cálculo da probabilidade da ocupação do *buffer* ser igual a sua capacidade total. A figura 4.9 mostra os resultados dos métodos de recompensas de taxa e impulso, que, conforme podemos ver, estão bem próximos. A tabela 4.3 apresenta o erro da medida em questão, reforçando a boa aproximação oferecida pelo método de recompensas de impulso. Foram considerados os erros para os valores de $R' = 30$ e $R' = 106$. O valor do erro absoluto para $R' = 32$ não foi apresentado devido a ser aproximadamente igual ao erro absoluto para $R' = 30$.

Tempo	Erro Absoluto (R'= 30)	Erro Absoluto (R'= 106)
10	3.6698222e-02	4.598e-03
20	3.3573957e-02	1.3627e-02
30	4.0919892e-02	1.5780e-02
40	4.0679517e-02	1.6554e-02
50	3.8421269e-02	1.6473e-02
60	3.5505914e-02	1.5864e-02
70	3.2448391e-02	1.4949e-02
80	2.945881e-02	1.3882e-02
90	2.6652685e-02	1.2768e-02
100	2.4113682e-02	1.1677e-02
200	1.30565e-02	6.058e-03
300	1.1299734e-02	4.964e-03
400	1.0710849e-02	4.537e-03
500	1.0347996e-02	4.273e-03
600	1.0070094e-02	4.083e-03
700	9.840046e-03	3.934e-03
800	9.642029e-03	3.814e-03
900	9.467298e-03	3.712e-03
1000	9.31047e-03	3.624e-03
1500	8.701313e-03	3.312e-03
2000	8.272004e-03	3.111e-03
3000	7.712888e-03	2.867e-03

Tabela 4.3: Erro absoluto - Probabilidade do *buffer* = 10.000.

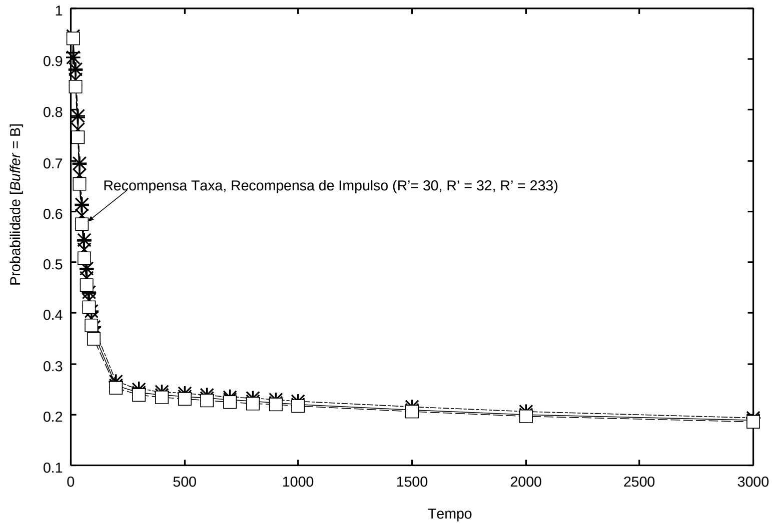


Figura 4.9: Probabilidade da ocupação do *buffer* ser igual ao tamanho total.

Para finalizar a comparação com o método de recompensas de taxa, apresentamos na figura 4.10 o custo computacional, em total de multiplicações. Conforme podemos observar, o custo computacional do método de recompensas de impulso é superior somente nos dois primeiros intervalos de tempo. Nos demais, o custo do método de recompensas de impulso sempre é menor, proporcionando diferenças extremamente significativas nos intervalos de tempo maiores que $t = 200s$, chegando a uma ordem de grandeza de diferença. No ponto $t = 3.000s$, o custo do método de recompensas de impulso foi de 997.560 multiplicações (para $R' = 30$), enquanto o custo total alcançado pelo método de recompensas de taxa foi de $10.880.000$ multiplicações. Conforme podemos ver, o custo do método de recompensas de impulso foi duas ordens de grandeza inferior ao custo obtido pelo método de recompensas de taxa. Este resultado apresenta um ganho computacional considerável.

Comparação II - Método de Recompensa de Impulso e Uniformização

Para completar o estudo do método proposto, realizamos uma comparação entre a utilização do método da uniformização, com a especificação e modelagem explícita do *buffer*, e o método de recompensas de impulso.

Cabe lembrar que para que o método de uniformização seja usado, todos os eventos

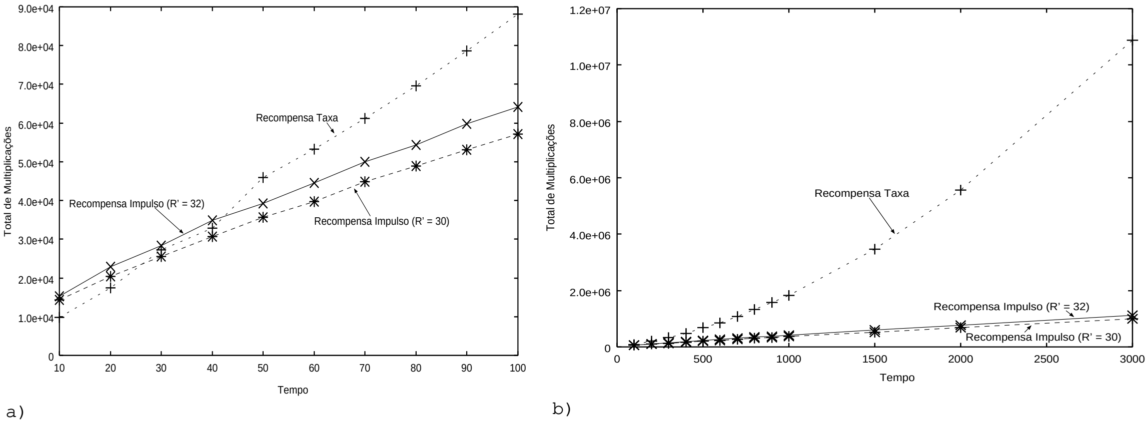


Figura 4.10: a)Custo computacional para os instantes de tempo $[10, 100]$; b)Custo computacional para os instantes de tempo $[100, 3.000]$.

do modelo devem ser exponencialmente distribuídos. Logo, no modelo resolvido com a uniformização, tanto o intervalo entre chegadas quanto o tempo de serviço dos pacotes têm distribuição exponencial, o que o diferencia do modelo considerado nos métodos de recompensa, onde os tempos (entre chegadas e serviço dos pacotes) são determinísticos.

Neste cenário consideramos dois modelos: o modelo **A** onde todos os eventos são exponenciais e o modelo **B** onde alguns eventos são determinísticos. Para que a comparação seja válida, tanto o modelo **A** quanto o modelo **B** devem fornecer resultados aproximados para a distribuição complementar da ocupação do *buffer*.

A figura 4.11 apresenta os resultados da distribuição complementar do *buffer* tanto para o método de uniformização (modelo **A**) quanto para o método de recompensas de taxa (modelo **B**), para $t = 10s$.

Neste exemplo, a diferença entre os resultados obtidos com o método de recompensas de taxa e de uniformização sempre foram inferiores a 4%. Podemos concluir que os modelos **A** e **B** fornecem resultado semelhante para a distribuição complementar do *buffer*.

Consideremos agora que ao invés da chegada/serviço de um pacote de cada vez no modelo **A**, ocorram chegadas/serviço de G pacotes. Com isso, estaremos reduzindo o custo da uniformização (que é diretamente proporcional ao tamanho do *buffer*). A motivação para a realização deste experimento é que no método de impulso o *buffer* é

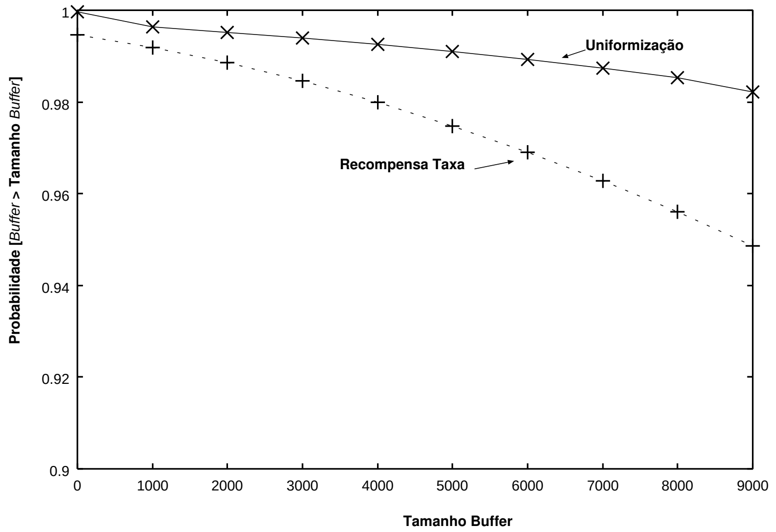


Figura 4.11: Distribuição Complementar da Ocupação do *buffer* para $t = 10s$.

dividido em R' níveis, onde $R' = tamanhobuffer/granularidade$. Isto significa que estamos considerando, no modelo **B**, a chegada/serviço de G pacotes. Com base neste fato podemos supor que se considerássemos a chegada/serviço de G pacotes no modelo **A**, poderíamos obter resultados semelhantes para diferentes valores de G. Concluindo, a utilização de níveis para a execução do método de recompensas de impulso pode nos levar a pensar que se utilizarmos o método de uniformização, devidamente escalonado, teremos o mesmo resultado. Para verificar o comportamento dos dois modelos quando o *buffer* é truncado, realizamos o teste descrito a seguir.

Consideramos um *buffer* de tamanho 10.000 e o total de níveis de recompensa igual a 30, temos que $G = 333$. Logo, o *buffer* truncado é igual a 30 pacotes para o modelo **B**.

No caso do modelo **A**, o tamanho do *buffer* também será 30 e as taxas de chegada e serviço de pacotes foram divididas por G, o que significa que estaremos considerando a chegada/serviço de G pacotes de cada vez.

Conforme podemos observar na figura 4.12, o tamanho reduzido do *buffer*, bem como a redução das taxas de envio/serviço de pacotes, fornece um resultado bastante diferente do resultado obtido quando o *buffer* é representado em sua totalidade.

Em relação ao custo computacional, mesmo que o modelo **A** truncado produzisse um resultado semelhante ao modelo com o *buffer* total, o custo do método de uniformização é

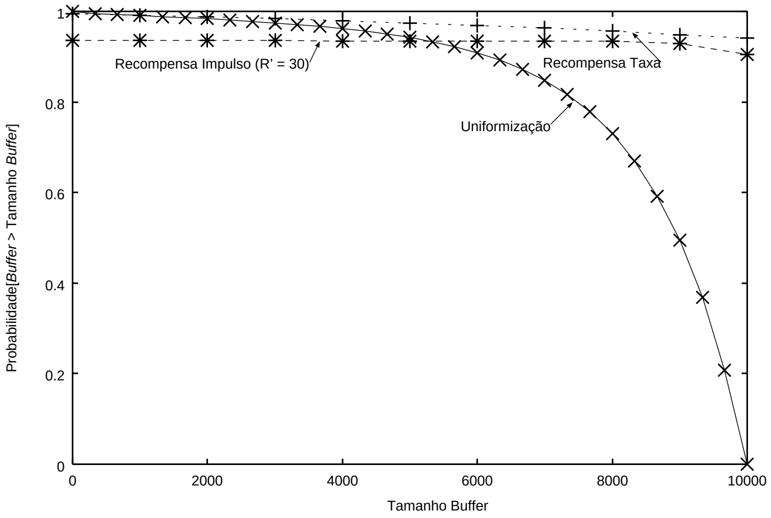


Figura 4.12: Distribuição complementar da ocupação do *buffer* para $t = 10s$, utilizando o método de impulso, método de taxa e o método de uniformização.

quase 90 vezes maior (1.273.220 multiplicações) do que o custo do método de impulsos.

4.7 Conclusão

A partir dos resultados dos exemplos, podemos observar que o fato de utilizarmos um número de níveis R', baseado na recompensa mínima, bastante superior ao R' escolhido baseado na média ponderada não apresenta redução significativa do erro obtido. Isto indica que o valor de R' baseado na média ponderada fornece a melhor relação entre o custo computacional e a precisão obtidos no cálculo da medida.

Os exemplos mostram também que o método de recompensas de impulso é extremamente eficiente para um conjunto de modelos onde as taxas de envio de informação são extremamente elevadas e o tempo de permanência em cada estado é bem maior que o intervalo entre chegadas/partidas de unidades de informação.

Capítulo 5

Método de Aproximação para o Cálculo da Distribuição da Ocupação do *Buffer* em Estado Estacionário

5.1 Introdução

CONFORME apresentado nos capítulos anteriores, a análise de sistemas computacionais em estado transiente é de extrema importância para o estudo do comportamento em tais sistemas. No entanto, não podemos deixar de prever também o seu comportamento em estado estacionário, principalmente quando estamos analisando sistemas em que as medidas de interesse se estabilizam em tempos considerados curtos.

Diversos métodos de solução para obtenção de distribuição de probabilidades dos estados de um sistema em regime estacionário são apresentados na literatura [18, 14, 26]. Estes métodos visam a resolução do sistema linear $\pi\mathbf{P} = \pi$, ou, em alguns casos, uma pequena variação deste ($\pi\mathbf{A} = 0$, onde $\mathbf{A} = \mathbf{I} - \mathbf{P}$). Alguns destes métodos exploram a estrutura especial da matriz de probabilidades, dividindo-a em blocos, manipulando estes através de eliminações e encontrando, finalmente, a solução desejada.

Para os modelos de fluido, a análise em estado estacionário também é importante, apesar de não ser muito explorada na literatura, fato este devido a complexidade da análise destes modelos. Considerando modelos de fonte de tráfego encontrados na literatura, podemos explorar a estrutura da matriz de probabilidades e estabelecer novos métodos de solução que se beneficiem de todas as suas características.

O principal objetivo deste capítulo é apresentar um novo método de solução para obtenção da distribuição em estado estacionário de um *buffer* baseado em modelos com recompensas de impulso e na estrutura da matriz gerada a partir desses modelos. O método proposto é baseado em [18] e não se restringe à aplicação de modelagem de tráfego, podendo ser usado em modelos que possuam a matriz de probabilidade de transição com uma estrutura especial.

A organização deste capítulo é descrita a seguir. Na seção 5.2 são apresentados alguns métodos de solução utilizados para o estudo de modelos em estado estacionário. Estes métodos são baseados na redução da matriz **P** através da eliminação de blocos. Na seção 5.3 é apresentada a modelagem utilizada, bem como um novo método de solução cujo o principal objetivo é a obtenção da solução do sistema $\pi\mathbf{P} = \pi$, sendo a matriz **P** uma matriz com estrutura típica de sistemas de redes multimídia. Nesta seção também são apresentadas algumas considerações adicionais sobre o novo método proposto. Finalmente, a seção 5.4 apresenta alguns exemplos.

5.2 Métodos de Solução

Diversos métodos de solução para obtenção de resultados em estado estacionário estão presentes na literatura. A grande maioria deles é utilizada para a resolução da equação $\pi\mathbf{P} = \pi$, onde π representa o vetor de probabilidades e **P** a matriz de probabilidades em uma transição e são aplicados em modelos Markovianos[14] ou não-Markovianos [15].

Entre os diversos métodos, estamos dando ênfase particularmente a dois métodos, já que estes são os mais próximos ao novo método de solução apresentado neste trabalho. Estes métodos podem ser utilizados para resolução de modelos de tráfegos com as seguintes características: o *buffer* é explicitamente modelado, o intervalo entre chegadas de pacotes da fonte tem distribuição exponencial (as taxas da fonte deixam de ser um fluido) e o tempo de serviço tem distribuição exponencial. Podemos observar que os métodos descritos nesta seção resolvem uma classe de modelos de tráfego onde todos os eventos tem distribuição exponencial. Contudo, em alguns casos, tanto o modelo onde todos os eventos tem distribuição exponencial quanto o modelo de fluido são uma boa aproximação para a seqüência de tráfego sendo modelada.

O primeiro método é o proposto por Grassman, Taksar e Heyman, conhecido como GTH ou método de eliminação de estados [26]. O segundo é o método proposto por Michella Meo, Edmundo de Souza e Silva e M. Ajmone Marsan, que propõe uma solução mais eficiente para uma classe de modelos Markovianos encontrados em sistemas de telecomunicações [18]. O método proposto neste capítulo é baseado em [18].

5.2.1 GTH

Grassman, Taksar e Heyman propuseram este método direto para solução de cadeias de Markov em geral. Este método é, na verdade, uma inteligente modificação da eliminação de Gauss e possui diversas características positivas. Uma delas é que somente adições e multiplicações são utilizadas, ou seja, não são realizadas subtrações que são as maiores responsáveis por erros de arredondamento. Outra característica presente no método é a interpretação probabilística do procedimento de eliminação.

Para a comparação com o método proposto neste trabalho, estamos utilizando a versão para solução de matrizes divididas em blocos. Neste caso, subconjuntos de estados são eliminados a cada passo, ao invés de um único estado, como no algoritmo original. Para cada passo da redução, o complemento estocástico para um subconjunto de estados é obtido.

Basicamente, o GTH é dividido em duas etapas, cada uma com $N - 1$ passos, sendo **N** o número total de blocos no qual a matriz original foi dividida. Na primeira etapa, a cada passo do algoritmo um subconjunto de estados é eliminado e as probabilidades de transição entre os demais blocos são recalculadas de forma que as probabilidades estacionárias dos estados da nova cadeia de Markov correspondam as probabilidades condicionais dos mesmos estados na cadeia original para o subconjunto restante.

Na segunda etapa, as probabilidades para os estados de cada bloco são calculadas a partir da obtenção de probabilidades não-normalizadas do bloco resultante do passo $N-1$ do algoritmo. Como último passo, todas as probabilidades são normalizadas resultando na distribuição de probabilidades estacionária do modelo estudado.

Formalmente, consideremos a matriz **P** $(M$ x $M)$. Assumindo que esta matriz está dividida em **N** B x B blocos diagonais:

$$\mathbf{P} = \begin{bmatrix} \mathbf{P}_{11} & \mathbf{P}_{12} & \cdots & \mathbf{P}_{1N} \\ \mathbf{P}_{21} & \mathbf{P}_{22} & \cdots & \mathbf{P}_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{P}_{N1} & \mathbf{P}_{N2} & \cdots & \mathbf{P}_{NN} \end{bmatrix}$$

Eliminando o último bloco, $\mathbf{P}_{NN}$, através do cálculo do complemento estocástico de **P** e considerando $\mathbf{P}^{(N-1)}$ este complemento, temos:

$$\mathbf{P}_{ij}^{(N-1)} = \mathbf{P}_{ij} + \mathbf{P}_{iN}(\mathbf{I} - \mathbf{P}_{NN})^{-1}\mathbf{P}_{Nj}$$

O algoritmo continua calculando o complemento estocástico de $\mathbf{P}^{(N-1)}$ para o bloco $(N-1, N-1)$, até o cálculo do complemento (B x B) de $\mathbf{P}^{(1)} = \mathbf{P}_{11}^{(1)}$.

As probabilidades não-normalizadas para os estados do primeiro subconjunto de estados são obtidas a partir de $\mathbf{P}^{(1)}$, através da resolução do seguinte sistema linear:

$$\pi_1 = \pi_1 \mathbf{P}^{(1)} \tag{5.1}$$

onde $\pi_1 =< \pi_{11}, \cdots, \pi_{1B} >$. Os demais $N-1$ vetores π_k são obtidos a partir de:

$$\pi_k = \sum_{i=1}^{k-1} \pi_i \mathbf{P}_{ik}^{(k)} (\mathbf{I} - \mathbf{P}_{kk}^{(k)})^{-1}$$

O último passo consiste na normalização de π.

O custo total do método GTH pode ser expresso através do total de blocos **N** e da ordem, neste caso quadrada e de valor igual a **B**, de cada bloco. Assim, a expressão de custo do algoritmo é de, considerando que cada inversão de $(\mathbf{I}-\mathbf{P}_{NN})^{-1}$ possui um custo na ordem de $O(2/3B^3)$ [14]:

$$T(N) = (N^3 + 5/3N)B^3 + (N^2/2)B^2 + C(\pi_1) \tag{5.2}$$

sendo $C(\pi_1)$ o custo para solução do sistema 5.1.

É evidente que o custo computacional expresso em 5.2 pode ser reduzido, principalmente quando este método é aplicado a matrizes com estruturas especiais, do tipo banda, por exemplo. Um novo custo computacional, considerando a estrutura da matriz gerada por um modelo de fonte de tráfego alimentando um *buffer*, será apresentado na seção 5.3.3.

5.2.2 Método Proposto em [18]

O método proposto por Michella Meo, Edmundo de Souza e Silva e M. Ajmone Marsan também pode ser visto como uma pequena modificação da eliminação de Gauss, de maneira similar ao GTH. No entanto, para a classe de modelos que são típicos das redes de telecomunicações de alta velocidade este método se mostra extremamente atraente, já que somente uma única inversa de matriz é calculada, em oposição ao GTH onde **N** inversas são necessárias. Isto é possível já que neste novo método as matrizes abaixo da diagonal principal é que são invertidas, ao passo que no método GTH as matrizes da diagonal é que sofrem inversão. Em [18] mostra-se que as matrizes abaixo da diagonal não se modificam no decorrer da execução do algoritmo, proporcionando uma única inversão matricial inicial.

O sistema a ser resolvido por este método é o sistema $\pi\mathbf{A} = 0$, onde $\mathbf{A} = \mathbf{I} - \mathbf{P}$. A matriz **A** é dividida em **N** blocos de ordem B x B cada:

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} & \cdots & \mathbf{A}_{1N} \\ \mathbf{A}_{21} & \mathbf{A}_{22} & \cdots & \mathbf{A}_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}_{N1} & \mathbf{A}_{N2} & \cdots & \mathbf{A}_{NN} \end{bmatrix}$$

Para cada passo k da redução, as seguintes multiplicações são efetuadas:

$$\mathbf{A}_{ik}^{k+1} = -\mathbf{A}_{ik}^{(k)}(\mathbf{A}_{k+1,j}^{(k)})^{-1}$$

$$\mathbf{A}_{ij}^{(k+1)} = \mathbf{A}_{ij}^{(k)} + \mathbf{A}_{ik}^{(k+1)}\mathbf{A}_{k+1,j}^{(k)}$$

sendo que o processo de redução é realizado $N-1$ vezes.

Ao final, similarmente ao método GTH, teremos o seguinte sistema de equações a ser resolvido:

$$\pi_0 = \pi_0 \mathbf{A}_{0N-1}^{(N-1)} \tag{5.3}$$

Os demais $N-1$ vetores π_k são obtidos a partir de:

$$\pi_k = \pi_0 \mathbf{A}_{0,k-1}^{(k)} + \sum_{j=k+1}^{(N-1)} \pi_j \mathbf{A}_{j,k-1}^{(k)}$$

O último passo consiste na normalização do vetor π.

O custo deste método também está relacionado com o número total de blocos em que a matriz **A** foi dividida, bem como da ordem de cada partição. Devemos ressaltar que para este método somente uma única inversão, $(\mathbf{A}_{k+1,j}^{(k)})^{-1}$, deve ser realizada. Assim, o custo computacional pode ser expresso como:

$$T(N) = (N^3 + N + 2/3)B^3 + (N^2/2 + N)B^2 + C(\pi_0) \tag{5.4}$$

sendo $C(\pi_0)$ o custo para resolução do sistema 5.3.

Mais uma vez, o custo computacional de 5.4 pode ser reduzido através da utilização de matrizes com estruturas especiais. Para estas matrizes, um novo custo computacional será mostrado na seção 5.3.3.

5.3 Método Proposto

Conforme sabemos, os modelos de fluido se caracterizam pelas altas taxas de envio/serviço de informações, em oposição as baixas taxas de mudanças entre as fases da fonte geradora de informações. Mais precisamente, temos que a fonte fica um tempo grande enviando altas taxas de informação, enquanto estas também são retiradas das filas dos servidores a altas taxas. Sabemos também que a variação proporcionada ao tamanho do *buffer* é sempre uma variação constante, que depende da taxa de envio/serviço de informações e do tempo passado em uma determinada fase da fonte. Realizamos um estudo em estado estacionário da distribuição de um *buffer* submetido a uma fonte que possui as características descritas acima. Novamente estamos utilizando o paradigma de recompensas de impulso, aproximando estas de um comportamento contínuo, ou seja, ao comportamento similar aos modelos onde são utilizadas recompensas de taxa.

5.3.1 A Modelagem Utilizada

Iremos basear o nosso modelo no modelo previamente definido no capítulo 4, onde a cada transição entre o estado j da fonte e o estado i é associada uma recompensa de impulso $\rho(j,i) = (\lambda_j - C)/\Lambda$. Neste modelo, as recompensas de impulso podem assumir valores diferentes já que o valor de Λ é igual para todos os estados.

Nosso objetivo é definir um modelo onde, em todas as fases da fonte, independentemente da recompensa $\lambda_j - C$ associada a uma determinada fase, o *buffer* 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 **P** com estrutura especial.

Nos parágrafos seguintes iremos mostrar, através de um exemplo, como o novo modelo é construído a partir de um modelo de uma fonte de tráfego.

Considere o modelo da fonte de tráfego representado na figura 5.1. Seja C a capacidade do canal de comunicação (igual a 15 unidades de informação/s). Assim, a taxa de aumento/decréscimo do *buffer* é de $\lambda_j - C$. O nosso objetivo é que o *buffer* seja sempre incrementado ou decrementado da mesma quantidade de informação. Para isso iremos definir uma taxa Λ_j para cada estado tal que:

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

onde K é a quantidade de informação que será acrescida/retirada do *buffer* cada vez que o processo fizer uma transição a partir do estado j.

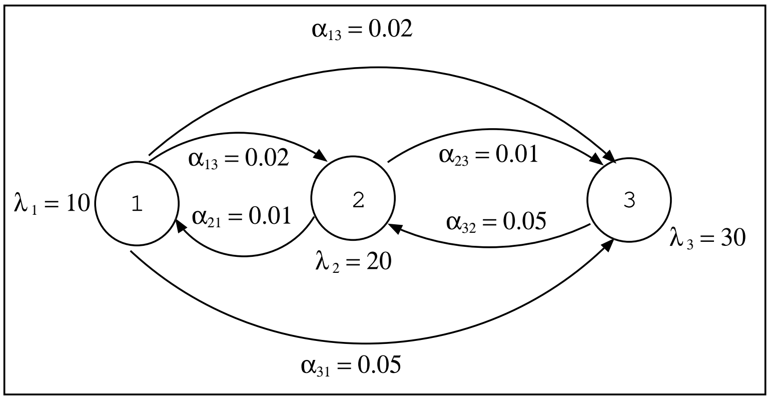


Figura 5.1: Modelo de uma fonte de tráfego.

Para o cálculo de Λ_j a seguinte restrição dever ser imposta: $\Lambda_j \geq \alpha_j$ onde α_j é a taxa de saída do estado j. Esta restrição não deve acarretar problemas para os modelos de fluido, já que para esta classe de modelos $\lambda_j - C >> \Lambda$ (onde $\Lambda = \max_{\forall j}\{\alpha_j\}$).

O estado do novo modelo será definido como o par ordenado $S_i^j = (i, j)$, onde j é a fase da fonte e i é o nível de recompensa (no caso do exemplo, o estado do *buffer*). Considerando para o modelo da figura 5.1 um *buffer* igual a 3 e $K = 1$, 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 5.2 apresenta a matriz **P** do modelo da figura 5.1.

As transições entre os estados da matriz e os impulsos associados às transições são obtidos da forma descrita abaixo.

Seja $i = \{1, 2, ..., B\}$, $n = \{1, 2, ..., B\}$, $j = \{0, 1, ..., N\}$ e $m = \{0, 1, ..., N\}$, onde N é o número de fases da fonte e B é o tamanho do *buffer* (valor máximo que a recompensa pode atingir):

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

$$
\begin{array}{c|cccccccccccc|}
 & S_0^1 & S_0^2 & S_0^3 & S_1^1 & S_1^2 & S_1^3 & S_2^1 & S_2^2 & S_2^3 & S_3^1 & S_3^2 & S_3^3 \\
\hline
S_0^1 & 0.992 & 0.004 & 0.004 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
S_0^2 & 0 & 0 & 0 & 0.002 & 0.996 & 0.002 & 0 & 0 & 0 & 0 & 0 & 0 \\
S_0^3 & 0 & 0 & 0 & 0.003 & 0.003 & 0.994 & 0 & 0 & 0 & 0 & 0 & 0 \\
S_1^1 & 0.992 & 0.004 & 0.004 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
S_1^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0.002 & 0.996 & 0.002 & 0 & 0 & 0 \\
S_1^3 & 0 & 0 & 0 & 0 & 0 & 0 & 0.003 & 0.003 & 0.994 & 0 & 0 & 0 \\
S_2^1 & 0 & 0 & 0 & 0.992 & 0.004 & 0.004 & 0 & 0 & 0 & 0 & 0 & 0 \\
S_2^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.002 & 0.996 & 0.002 \\
S_2^3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.003 & 0.003 & 0.994 \\
S_3^1 & 0 & 0 & 0 & 0 & 0 & 0 & 0.992 & 0.004 & 0.004 & 0 & 0 & 0 \\
S_3^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.002 & 0.996 & 0.002 \\
S_3^3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.003 & 0.003 & 0.996 \\
\hline
\end{array}
$$

Figura 5.2: Matriz de probabilidade de transições do modelo 5.1.

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

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

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

2. Se $\lambda_j - C < 0$ e $i = n = 0$ ou $\lambda_j - C > 0$ e $i = n = B$

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

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

$$\rho(j, m) = 0$$

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

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

Nesta nova modelagem, o processo passa em cada fase da fonte/nível de recompensa um tempo médio diferente. Desta forma, a matriz **P** passa a ser definida como sendo **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.

Como vimos anteriormente, cada estado do modelo passa a ser representado pela fase da fonte e pelo nível de recompensa acumulada. Como poderemos ver na seção 5.4, o número de níveis pode ser bem menor que o tamanho real do *buffer* alimentado pela fonte de informação, o que diminui sensivelmente a complexidade computacional do método apresentado.

5.3.2 O Método

O nosso objetivo é o cálculo da distribuição em estado estacionário da ocupação de um *buffer* alimentado por uma fonte de dados quando tanto a taxa da fonte quanto a taxa de serviço são representadas por um fluido.

Iremos considerar os estados ordenados da seguinte forma: fixamos o nível de recompensa e variamos as fases da fonte, iniciando com as fases da fonte com recompensa negativa (conforme figura 5.2). A matriz **P** apresentada na figura 5.2 pode ser organizada em blocos conforme a figura 5.3.

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

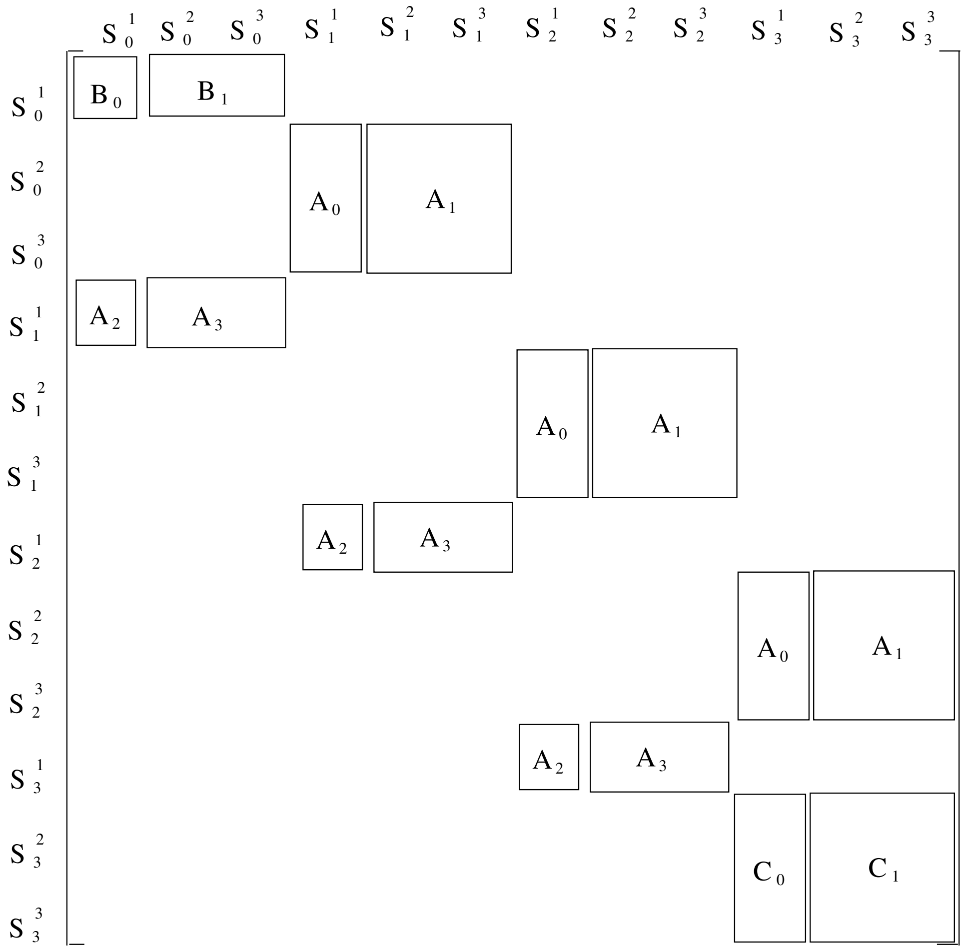


Figura 5.3: Matriz de probabilidades **P** particionada em blocos.

1. Os blocos $\mathbf{B}_0$ e $\mathbf{B}_1$ representam, respectivamente, as transições entre estados de recompensa negativa e as transições de um estado em que a recompensa é negativa para um estado em que a recompensa é positiva. Em ambos os casos, o *buffer* está com o nível igual ao limite inferior. Portanto, não é possível a mudança de nível (a recompensa associada às transições entre estados pertencentes a estes blocos é zero).

2. Os blocos $\mathbf{C}_0$ e $\mathbf{C}_1$ representam, respectivamente, as transições de um estado em que a recompensa é positiva para um estado em que a recompensa é negativa e a as transições entre estados em que a recompensa é positiva. Em ambos os casos, o *buffer* está com o nível igual ao limite superior. Portanto, não é possível mudança de nível (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. Nos blocos $\mathbf{A}_0$ e $\mathbf{A}_1$ parte-se de estados em que a recompensa é positiva para estados em que, respectivamente, a recompensa é negativa e para estados em que a recompensa é positiva (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 a recompensa é negativa para estados em que, respectivamente, a recompensa é negativa e estados em que a recompensa é positiva (a recompensa associada às transições entre estados pertencentes a estes blocos tem valor $-K$).

4. Os blocos diagonais são todos nulos, excetuando o primeiro e o último blocos. Este fato ocorre devido as características do modelo especificado, onde à 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 a mesma fase da fonte sem a mudança do nível de recompensa.

O objetivo do método proposto é resolver o sistema de equações dado por $\pi\mathbf{P} = \pi$. Consideremos o número total de níveis de recompensa igual a $\mathbf{L}$ e o vetor π particionado em $L+2$ blocos π_i^j onde j representa a fase da fonte e i representa o nível de recompensa acumulada ($i = \{1, 2, ..., L\}$). Definiremos $j = 0$ para as fases onde a recompensa é negativa (decremento do *buffer*) e $j = 1$ para as fases onde a recompensa é positiva (incremento do *buffer*).

A figura 5.4 apresenta os blocos que devem ser eliminados a cada passo do algoritmo, bem como o vetor π devidamente particionado.

Como característica deste novo método, a cada passo da redução da matriz $\mathbf{P}$ 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.

Consideremos as duas primeiras equações obtidas a partir das duas primeiras colunas de 5.4:

$$\pi_0^0\mathbf{B}_0 + \pi_1^0\mathbf{A}_2 = \pi_0^0 \tag{5.5}$$

$$\pi_0^0\mathbf{B}_1 + \pi_1^0\mathbf{A}_3 = \pi_0^1 \tag{5.6}$$

$$
< \pi_0^0, \pi_0^1, \cdots, \pi_L^0, \pi_L^1 > \begin{bmatrix} \boxed{B_0} & B_1 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & A_0 & A_1 & 0 & 0 & \cdots & 0 \\ A_2 & A_3 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & A_0 & A_1 & \cdots & 0 \\ 0 & 0 & A_2 & A_3 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & C_0 & \boxed{C_1} \end{bmatrix} = < \pi_0^0, \pi_0^1, \cdots, \pi_L^0, \pi_L^1 >
$$

Figura 5.4: Blocos que devem ser eliminados a cada passo do algoritmo.

Resolvendo a equação 5.5 para π_1^0 e a equação 5.6 para π_0^1, teremos:

$$\pi_1^0 = \pi_0^0(\mathbf{I} - \mathbf{B}_0)(\mathbf{A}_2)^{-1} \tag{5.7}$$

$$\pi_0^1 = \pi_0^0[\mathbf{B}_1 + (\mathbf{I} - \mathbf{B}_0)(\mathbf{A}_2)^{-1}\mathbf{A}_3] \tag{5.8}$$

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:

$$\pi_0^0(\mathbf{B}_0^{(L)} + \mathbf{B}_1^{(L)}(\mathbf{I} - \mathbf{C}_1)^{-1}\mathbf{C}_0) = \pi_0^0 \tag{5.9}$$

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 3.

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.

Para o algoritmo apresentado, deveremos calcular a inversa da matriz $(\mathbf{I} - \mathbf{C}_1)$. Para tal, o seguinte teorema é utilizado para provar a existência da inversa:

Teorema 2 *Toda matriz da forma* $(\mathbf{I} - \mathbf{S})$ *sendo* $\mathbf{S}$ *uma matriz subestocástica, possui inversa.*

Prova: *Temos pelo próprio algoritmo, que a matriz* $\mathbf{C}_1$ *é uma partição da matriz* **P** *de probabilidades que representa o modelo estudado. Desta forma, cada linha da matriz* $\mathbf{C}_1$ *quando somada resulta em um valor inferior a 1, o que a caracteriza como uma matriz subestocástica. Assim, a matriz* $(\mathbf{I} - \mathbf{C}_1)$ *é da forma* $(\mathbf{I} - \mathbf{S})$. *Em [14] é mostrado que toda a matriz da forma* $(\mathbf{I} - \mathbf{S})$ *é uma matriz não singular, possuindo inversa. Logo, a inversa de* $(\mathbf{I} - \mathbf{C}_1)$ *sempre existe.*

A inversa da matriz $\mathbf{A}_2^{(0)}$ também existe, conforme mostrado em [18].

Como descrito na seção 5.3.1, o processo modelado passa em cada estado S_i^j um tempo médio de permanência $1/\Lambda_j$. Assim, como o processo fica um tempo médio $1/\Lambda_j$ quando este visita um determinado estado S_i^j, podemos obter π_i^j a partir de [22]:

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

o que resulta na função distribuição de probabilidade final.

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 $N = L$. Além disso, cada partição da matriz **P** é de ordem B x B, sendo esta proporcional ao número de fases da fonte em que a recompensa é maior que zero e ao número de fases da fonte em que a recompensa é menor que zero. Assumiremos que o número de fases em que a recompensa é maior que zero é igual o número de fases em que a recompensa é menor que zero. Portanto, temos sempre o mesmo valor para B. O custo pode ser expresso através da seguinte expressão:

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

sendo $C(\pi_0)$ o custo para resolução do sistema 5.9. 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) reduz o custo do algoritmo proposto.

A respeito do custo computacional do método proposto, devemos ressaltar que somente uma única inversão é realizada. As demais inversões, relacionadas com o termo $(\mathbf{A}_2^{(k)})^{-1}$ não são realizadas, devido a seguinte propriedade matricial:

1. Se **A** e **B** são matrizes quadradas de mesma ordem, ambas inversíveis (isto é, existem $\mathbf{A}^{-1}$ e $\mathbf{B}^{-1}$), então **AB** é inversível e $(\mathbf{AB})^{-1} = \mathbf{B}^{-1}\mathbf{A}^{-1}$ [1].

5.3.3 Comparação dos Custos Computacionais

Nas seções 5.2.1 e 5.2.2 foram apresentados os custos computacionais para o método GTH e o método proposto em [18], respectivamente. No entanto, as equações de custo apresentadas não levavam em conta a estrutura especial da matriz a ser resolvida.

Nos casos em que o *buffer* é modelado em sua totalidade, a matriz **P** é uma matriz tridiagonal, fato este que reduz sensivelmente o custo computacional dos métodos citados. Além deste fator, podemos também considerar a característica esparsa da matriz que representa o modelo. No caso do GTH, a nova expressão de custo passa a ser:

$$T(N) = (2/3N)B^3 + 9NB^2 + C(\pi_0)$$

sendo **N** o tamanho total do *buffer* e **B** a ordem das matrizes multiplicadas, sendo esta ordem proporcional ao número total de fases da fonte.

A queda no custo computacional se deve principalmente ao fato de que a estrutura tridiagonal da matriz **P** resulta no máximo em três multiplicações matriciais para cada linha considerada durante o processo de redução. No caso das inversas a serem calculadas, o fator computacional relacionado se mostra invariante, visto que os blocos diagonais continuam presentes.

No caso do método proposto em [18], a estrutura tridiagonal da matriz favorece ainda mais a queda do custo computacional. Isto ocorre dado que a matriz abaixo da diagonal, a única matriz que é invertida é uma matriz diagonal, não necessitando de nenhum processo de inversão [18]. A estrutura tridiagonal também resulta em no máximo três multiplicações matriciais para cada linha considerada no procedimento de redução. Considerando a característica esparsa da matriz **P** temos:

$$T(N) = 8NB^2 + C(\pi_0)$$

sendo **N** o tamanho total do *buffer* e **B** a ordem das matrizes multiplicadas, sendo esta ordem proporcional ao número total de fases da fonte.

Um outro ponto a ser ressaltado é a relação entre os parâmetros **N** e **B** dos quais o custo computacional dos três algoritmos analisados é proporcional. No caso do algoritmo proposto, o parâmetro **N** representa o total de níveis da recompensa de impulso acumulada e o parâmetro **B** representa o número de fases da fonte em que a recompensa é positiva ou negativa. Nos outros algoritmos apresentados, o parâmetro **N** representa o tamanho do *buffer* enquanto o parâmetro **B** representa o total de fases da fonte. Considerando que podemos reduzir o tamanho do *buffer* através da definição de níveis, o custo do método proposto se reduzirá bastante, o que o torna extremamente atraente para os modelos de fluido considerados. Além disso, se considerarmos que o número total de fases da fonte em que a recompensa é positiva é igual ao número total de fases da fonte em que a recompensa é negativa, temos que o parâmetro **B** passa a ser a metade do valor considerado para o algoritmo GTH e para o algoritmo proposto em [18].

Finalizando, as condições para que o custo do método proposto neste trabalho seja menor que os custos do método GTH e do método encontrado em [18], podem ser definidas como:

1. Método proposto em [18]:

$$\frac{7NB^3}{8/3} + \frac{7B^3}{24} + \frac{2NB^2}{4/6} < \frac{8NB^2}{1/24}$$
$$21NB^3 + 7B^3 + 12NB^2 < 192NB^2$$
$$21NB + 7B + 12N < 192N$$

   como B¡¡N:

$$21NB + 12N < 192N$$
$$21B < 180$$
$$B \leq 8$$

2. GTH:

   Considerando B¡¡N:

$$\frac{7NB^3}{8} + \frac{2NB^2}{4} < \frac{2NB^3}{3} + 9NB^2$$
$$\frac{7B}{8/3} + \frac{2}{4/6} < \frac{2B}{3/8} + \frac{9}{1/24}$$
$$21B + 12 < 16B + 216$$
$$B \leq 40$$

5.3.4 Considerações Adicionais

Para utilizarmos o método proposto devemos mapear a fonte original de dados para um novo modelo, onde o estado é definido pela fase da fonte e o nível da recompensa acumulada. Conforme vimos anteriormente na seção 5.3.1, devemos calcular a probabilidade de mudança entre as fases da fonte utilizando a taxa original de mudança entre fases e a nova taxa de uniformização que possui um valor diferente para cada estado do modelo.

Mediante estas especificações, ressaltamos que o método proposto somente pode ser utilizado quando as taxas originais de transição entre os estados da fonte são menores que a nova taxa de uniformização para cada estado da cadeia, proporcionando, assim, após a divisão destes parâmetros, valores válidos para a probabilidade.

Esta restrição não causa grandes problemas para o método proposto, já que a característica principal dos modelos de fluido está fundamentada no fato da fonte a ser modelada possuir taxas de envio e serviço de informações bem maiores que as taxas de transição entre as suas fases. Assim, na grande maioria dos modelos a serem estudados, esta restrição não acontece, o que permite a utilização do método proposto.

Uma outra consideração a ser feita é no caso em que existam fases da fonte em que a recompensa acumulada é igual a zero. Para tratarmos destes casos, podemos considerar que estes estados se enquadram na categoria das fases onde se ganha recompensa, sendo que este ganho é igual a zero. Analogamente, podemos considerar como sendo da categoria onde se perde recompensa, sendo que esta perda é igual a zero. O valor para Λ_j pode ser considerado igual a taxa de uniformização original do modelo. Esta maneira particular de tratar modelos onde a fonte acumula recompensa igual a zero em uma das suas fases, não altera a definição do método proposto para estudo de modelos de fluido em estado estacionário visto que ao fim da execução do método, sempre consideramos a média ponderada de permanência em cada estado do modelo. Podemos, assim, aplicar este tratamento particular, sem nos preocuparmos se estamos infringindo a corretude do método.

Finalmente, devemos ressaltar que a estrutura da matriz de probabilidades **P** proposta na seção 5.3.1 não pode ser resolvida pelos métodos apresentados na seção 5.2. No caso do método GTH, é necessária a inversão das matrizes diagonais. A estrutura da matriz proposta possui na diagonal matrizes nulas e, portanto, não possuem inversas. No caso do método encontrado em [18], é necessária a inversão das matrizes abaixo da diagonal. Da mesma forma, as matrizes abaixo da diagonal também não podem ser invertidas, o que inviabiliza a utilização deste segundo método para a resolução dos modelos estudados.

5.4 Exemplos

Esta seção apresentará alguns resultados provenientes de estudos de modelos de tráfego em estado estacionário. Serão utilizados dois exemplos com o objetivo de avaliar o custo computacional e o erro obtidos com o método proposto.

No primeiro exemplo estaremos utilizando a fonte *ONOFF* alimentando um *buffer* conforme apresentado no capítulo 4. O principal objetivo deste primeiro exemplo é mostrar o erro obtido com este novo método. O segundo exemplo apresenta uma fonte geradora de tráfego que modela uma seqüência de vídeo alimentando um *buffer* de dados. O principal objetivo deste segundo exemplo é realizar um estudo comparativo do custo computacional com relação a outros métodos da literatura.

5.4.1 Fonte *ONOFF*

O exemplo apresentado nesta seção é o mesmo exemplo referenciado no capítulo 4, com os mesmos parâmetros e considerando o mesmo tamanho de *buffer*, sendo este igual a 20.000 posições.

Para comparar e avaliar os resultados obtidos, estaremos utilizando o algoritmo de recompensa de taxas apresentado no capítulo 4, considerando um tempo t bastante superior aos demais tempos de ocorrências de eventos. Esta não é a melhor maneira de se obter resultados em estado estacionário, no entanto, este é o método mais adequado para o propósito apresentado.

Primeiramente, foi encontrada a matriz de probabilidades **P**, onde cada linha representa um estado formado por duas variáveis aleatórias: fase da fonte de dados e nível de recompensa acumulada. Para tal, consideramos que o tempo de permanência em cada fase da fonte (*ON* ou *OFF*) é diferente e está relacionado com o parâmetro K especificado anteriormente.

A partir dessas considerações, foi calculada a probabilidade do *buffer* se encontrar com a ocupação igual a sua capacidade total. A quantidade de níveis de recompensa acumulada variou dentro do conjunto $R' = \{50, 100, 200, 300, 400\}$. Isto significa que os valores considerados para K foram $K = \{400, 200, 100, 67, 50\}$, já que $R' = B/K$. Esta variação tem como objetivo apresentar o erro encontrado pela aplicação do método e a queda do custo computacional quando podemos considerar a representação do *buffer* através de níveis cada vez menores em relação ao seu tamanho original.

O gráfico da figura 5.5 apresenta o erro obtido para diferentes níveis de recompensa de impulso acumulada. Conforme podemos observar, o erro absoluto se encontra na quarta casa decimal. Considerando que a medida original tem o seu valor na ordem de 10^{-2}, podemos concluir que o método proposto é uma boa aproximação para o método que utiliza recompensa de taxa.

A redução do tamanho original do *buffer* em 400 vezes se tornou possível devido as características do modelo apresentado. Assim, mesmo com a redução do tamanho do *buffer*, o valor do erro absoluto foi menor que 1% do valor da medida real.

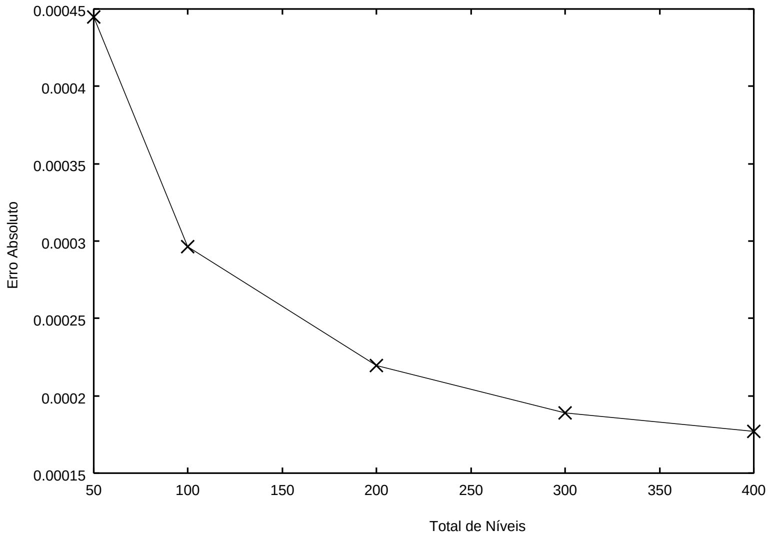


Figura 5.5: Erro absoluto da probabilidade do *buffer* estar com a capacidade total ocupada.

Finalizando, consideremos este *buffer* dividido em 200 níveis sendo $K = 100$. A figura 5.6 mostra a distribuição complementar da ocupação do *buffer* em estado estacionário. Conforme podemos observar, o método proposto é uma boa aproximação para cálculo da distribuição complementar em estado estacionário.

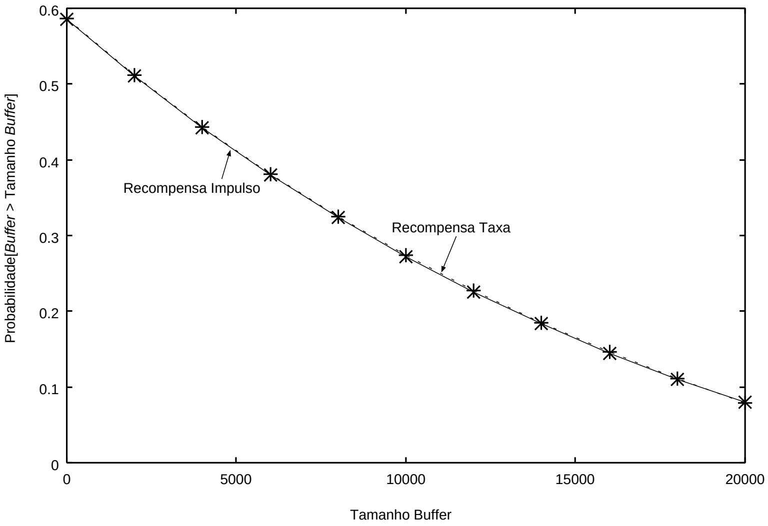


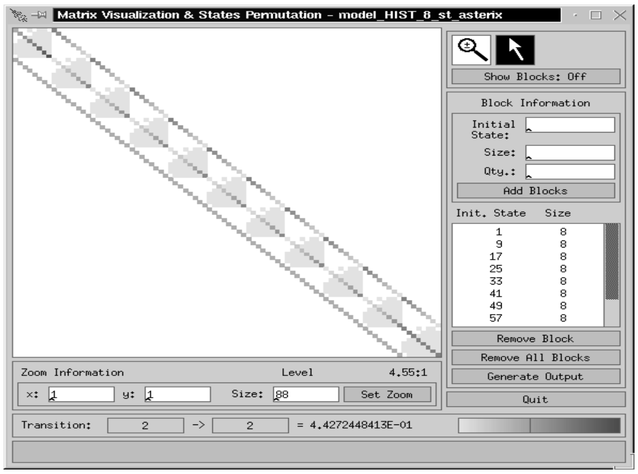
Figura 5.6: Distribuição complementar da ocupação do *buffer*.

5.4.2 Fonte de Vídeo

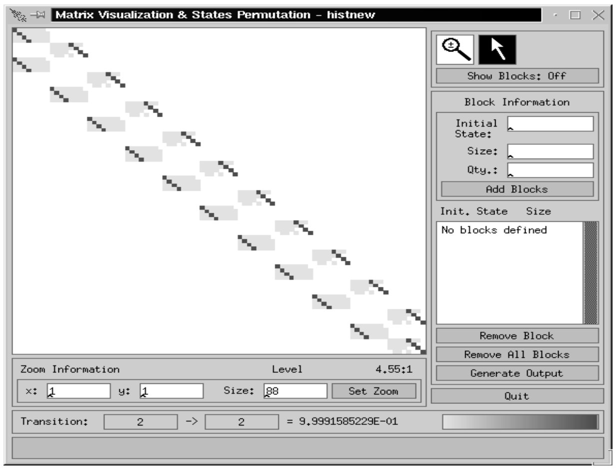
Para o estudo do custo computacional do algoritmo proposto e comparação com o custo do método GTH e do método encontrado em [18], foi modelada uma fonte composta por oito fases distintas de envio de dados, sendo que na metade delas o processo ganha recompensa e na outra metade o processo perde recompensa. A seqüência modelada é a seqüência do filme Asterix e pode ser encontrada em [20]. O modelo foi parametrizado como em [23] e os parâmetros (em bits/s) estão apresentados na tabela 5.1. A capacidade do canal é de 76.800 bits/s e consideraremos, originalmente, que o *buffer* possui 10.000 posições. A matriz **P** de dois modelos diferentes pode ser visualizada na figura 5.7. A matriz $a)$ representa a matriz **P** do modelo onde o *buffer* é representado. A matriz $b)$ representa a matriz **P** do modelo proposto.

Através da figura 5.7 podemos visualizar, de maneira mais clara, o motivo pelo qual o método GTH e o método proposto em [18] não podem ser utilizados para a resolução de matrizes com a estrutura especial apresentada em **b)**. A utilização do método GTH se torna impossível devido as matrizes diagonais serem matrizes nulas (deveriam ser invertidas). Em relação ao método proposto em [18], a restrição se deve ao fato das matrizes abaixo da diagonal principal não serem inversíveis pois são nulas.

A tabela 5.2 mostra o custo computacional de cada método apresentado. Para o



a)



b)

Figura 5.7: Comparação das diferentes estruturas de matrizes utilizadas: a) Matriz **P** do modelo onde o *buffer* é representado; b) Matriz **P** do modelo proposto.

cálculo, redimensionamos o *buffer* para 10 níveis diferentes $[1.000, 10.000]$ e consideramos $B = 8$, para o método GTH e o método encontrado em [18], e $B = 4$ para o método proposto. Podemos notar que o custo do método proposto sempre se encontra abaixo do custo dos outros métodos apresentados. O fator principal, entre outros, que possibilita este quadro está relacionado com a multiplicação de matrizes de ordem menor do que as matrizes do método GTH e do método [18]. Além disso, mesmo que em alguns casos o método proposto necessite de um número maior de multiplicações do que os outros métodos, devemos estar atentos para o fato de que a modelagem de uma seqüência de vídeo alimentando um *buffer* usando somente taxas exponenciais nem sempre fornece uma boa aproximação do comportamento do sistema. Nestes casos, os modelos de fluido podem fonecer um resultado melhor, sendo, portanto, importante a investigação de métodos para resolver essa classe de modelos.

Finalizando, consideremos a distribuição complementar da ocupação do *buffer* sendo que este é redimensionado de 10.000 para 50 níveis, cada um com $K = 200$. Assim, o Λ_j de cada fase da fonte passa a ter o valor tal que o crescimento/descrescimento do *buffer* seja igual a K. A tabela 5.3 apresenta os resultados obtidos. Podemos observar que o método de aproximação apresenta resultados bem próximos ao resultado exato, com erros menores que 10% da medida exata.

Estado	Taxa
1	67.304
2	48.920
3	30.536
4	12.152
5	6.532
6	24.616
7	43.000
8	61.384

Tabela 5.1: Parâmetros - Fonte de Vídeo.

Total Níveis	GTH	Método Proposto em [18]	Método Proposto
1000	918.250	512.512	480.629
2000	1.835.584	1.024.512	960.629
3000	2.752.917	1.536.512	1.440.629
4000	3.670.250	2.048.512	1.920.629
5000	4.587.584	2.560.512	2.400.629
6000	5.504.917	3.072.512	2.880.629
7000	6.422.250	3.584.512	3.360.629
8000	7.339.584	4.096.512	3.840.629
9000	8.256.917	4.608.512	4.320.629
10000	9.174.250	5.120.512	4.800.629

Tabela 5.2: Custo Comparativo entre os métodos estudados.

Tamanho *Buffer*	Método Recompensa Taxa	Método Recompensa Impulso
0	4.27010644e-02	4.0502995098e-02
1000	3.93131733e-02	3.7265560296e-02
2000	3.62205220e-02	3.4310287777e-02
3000	3.33951209e-02	3.1610880282e-02
4000	3.08116835e-02	2.9143547094e-02
5000	2.84473634e-02	2.6886764432e-02
6000	2.62815175e-02	2.4821058814e-02
7000	2.42954915e-02	2.1194079749e-02
8000	2.24724268e-02	2.2928811171e-02
9000	2.07970861e-02	1.9602439974e-02
10000	1.92556959e-02	1.8423340376e-02

Tabela 5.3: Distribuição complementar da ocupação do *buffer*.

Capítulo 6

Conclusões

O estudo de métodos de solução para obtenção de medidas de interesse que visam a análise de sistemas de computação e comunicação é uma área de pesquisa extremamente importante. Definir e propor métodos que explorem características especiais de uma classe de modelos possibilita resolver de maneira mais eficiente e, conseqüentemente, com um menor custo computacional o sistema em questão. Em alguns casos o algoritmo proposto pode proporcionar maneiras mais simples de definir e de obter as medidas de interesse a serem estudadas.

O objetivo principal desta tese foi apresentar um conjunto de métodos para solução de modelos baseados em recompensa. Dois métodos previamente propostos na literatura, um para obtenção do valor esperado da recompensa de taxa acumulada em $(0, t)$ e outro para o cálculo da distribuição da recompensa de impulso em $(0, t)$ foram implementados e foram avaliados o custo computacional e erro obtidos (capítulos 2,3,4). Além desse estudo, apresentamos um novo método de solução para matrizes com uma certa estrutura que visa a resolução de modelos de fluido em estado estacionário (capítulo 5). Todos os métodos apresentados visam a resolução de modelos Markovianos que utilizam o paradigma de recompensa, o que possibilita definirmos com maior flexibilidade diversas medidas de interesse relevantes ao sistema em estudo.

No capítulo 2 é apresentado um método para o cálculo do valor esperado da recompensa de taxa acumulada em um intervalo $(0, t)$. A utilização da distribuição erlangiana para aproximar o tempo de observação determinístico t, neste método apresentado, proporciona ganhos computacionais amplamente significativos para a classe especial de modelos do tipo *stiff*. Conforme mostrado, em alguns casos, a técnica de uniformização possui complexidade extremamente elevada, o que, em termos práticos, a torna inviável. Dessa forma, utilizar métodos de aproximação torna possível o estudo de sistemas com características consideradas desfavoráveis para os métodos de solução tradicionais. Um outro ponto inerente ao método proposto é que a estrutura da matriz de probabilidades pode ser explorada, mais precisamente na técnica direta, como meio de diminuição do custo computacional. Esta flexibilidade não se aplica ao método tradicional de uniformização. Com relação ao erro obtido com o método, foi proposta uma estimativa do erro que mostra ser uma boa indicação do erro real.

A definição de novos métodos de solução de modelos requer a consideração de dois aspectos: o custo computacional, relacionado com a capacidade de processamento para a resolução do modelo, e o espaço de mémoria e a capacidade de processamento necessários para a geração do espaço de estados do modelo que representa o sistema a ser analisado. O método de solução apresentado no capítulo 4 considera estes dois aspectos e possibilita o estudo do comportamento de *buffers* com alta capacidade de armazenamento e alimentados por um tráfego de fluido. A modelagem introduzida permite o mapeamento de modelos com recompensa de taxa em modelos com recompensa de impulso. O método apresentado é aplicado na obtenção da distribuição de probabilidades em estado transiente da ocupação de um *buffer*, sem que este seja modelado e representado no espaço de estados. Concluimos, através do estudo realizado, que o método de recompensa de impulso, tradicionalmente utilizado em medidas discretas e que foi mapeado para uma medida contínua, é extremamente atraente nos casos em que o período transiente da medida é longo, o que acarreta um alto custo computacional quando o método de recompensas de taxa é utilizado. O método também apresenta vantagens computacionais quando o método de recompensa de taxa somente permite a obtenção de limites para a medida considerada. Através dos exemplos analisados, pôde-se constatar que mesmo com um número reduzido de níveis de impulso (o que garante baixo custo computacional) obtém-se um erro bastante razoável.

Como continuidade dos estudos dos modelos de fluido, propomos um novo método de solução para a obtenção da distribuição de probabilidades da ocupação do *buffer* em estado estacionário. A modelagem utilizada é baseada na atribuição de recompensas de impulso às transições entre os estados do modelo de forma que a matriz de probabilidade de transição obtida possua uma estrutura especial. O método proposto é baseado na eliminação de blocos e possibilita a resolução de uma classe de modelos que possuem a matriz de probabilidades de transição com uma certa estrutura. Os modelos de fluido são um exemplo de modelos desta classe. Para estes modelos, os métodos tradicionais como o GTH [26] e o método proposto em [18] não são aplicáveis, já que se destinam a resolução de uma outra classe de modelos. Os exemplos estudados mostram que o erro obtido é bem pequeno e o custo computacional é inferior quando comparado ao custo de métodos tradicionais como o GTH. Este novo método é uma contribuição ao estudo de modelos de fluido, não havendo, do nosso conhecimento, métodos similares na literatura.

6.1 Trabalhos Futuros

A partir do trabalho desenvolvido nesta tese podemos relacionar alguns estudos que poderiam ser realizados.

1. Poderíamos investigar o cálculo de um outra medida de interesse a partir do método apresentado no capítulo 4 para obtenção da distribuição transiente do *buffer*. A medida de interesse é a distribuição do tempo em que o *buffer* fica acima do seu valor total. Esta medida é importante pois permite estimar, entre outros parâmetros, a quantidade de informação descartada pelo *buffer* durante um intervalo $(0, t)$.

2. Em relação ao método proposto para análise em estado estacionário de modelos de fluido, podemos investigar outras classes de modelos que possuam a matriz $\mathbf{P}$ com a estrutura utilizada pelo método, favorecendo a diminuição do custo computacional para a obtenção da medida. Um outro estudo que pode ser realizado é uma possível extensão do método para o caso de matrizes infinitas, baseado no trabalho apresentado em [12]. Estudos prévios mostram que o método apresentado no capítulo 5 possui todas as características que possibilitam a sua aplicação para esta classe de problemas.

Referências Bibliográficas

[1] BOLDRINI, J. L., COSTA, S. I. R., E E HENRY G. WETZLER, V. L. F. *Álgebra Linear*. Harper Row do Brasil, 1980.

[2] CARMO, R., DE CARVALHO, L., DE SOUZA E SILVA, E., DINIZ, M., E MUNTZ, R. Performance/Availability Modeling with the TANGRAM-II Modeling Environment. *Performance Evaluation 33* (1998), 45–65.

[3] CHI FENG, W. URL http://www.cis.ohio-state.edu/ wuchi/video/mjpeg/. MJPEG traces.

[4] CHI FENG, W. *Vídeo-on-Demand Services: Efficient Transportation an Decompression of Variable Bit Rate Video*. PhD thesis, University of Michigan, 1996.

[5] DE ANÁLISE E DESEMPENHO DE SISTEMAS DE COMPUTACÃO E COMUNICACÃO, L. User's Guide - TANGRAM-II.

[6] DE BRITO, C. E. F. Medidas Transientes para Recompensa Acumulada Limitada: Novas Técnicas de Solução e Aplicações para Redes de Alta Velocidade. Tese de Mestrado, Programa de Pós-Graduação de Engenharia de Sistemas e Computa ção da COPPE/UFRJ, 1999.

[7] DE BRITO, C. E. F., DE SOUZA E SILVA, E., DINIZ, M. C., E LEAO, R. M. Análise Transiente de Modelos de Fonte Multimidia. In *Anais do Simpósio Brasileiro de Redes de Computadores* (2000).

[8] DE SOUZA E SILVA, E., E GAIL, H. An Algorithm to Calculate Transient Distributions of Cumulative Rate and Impulse based Reward. *Communications in Statistics - Stochastic Models 14*, 3 (1998), 509–536.

[9] DE SOUZA E SILVA, E., E GAIL, H. *Transient Solutions for Markov Chains*. Kluwer, 2000, pp. 44–79.

[10] DE SOUZA E SILVA, E., E GAIL, H. R. The Uniformization Method in Performability Analysis. Relatório Técnico 2, IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598, 1996.

[11] DE SOUZA E SILVA, E., E LEÃO, R. M. M. The TANGRAM-II Environment. In *Computer Peformance Evaluation - Modelling Techniques and Tools* - $11^{\underline{th}}$ *International Conference (TOOLS2000)*, vol. 1786. Springer, Março 2000, pp. 366–369.

[12] DE SOUZA E SILVA, E., MEO, M., E MARSAN, M. A. A Method for Calculating Sucessive Approximate Solutions for a Class of Block Banded M/G/1 Type Markovian Models. *Performance Evaluation 44* (2001), 97–119.

[13] DE SOUZA E SILVA, E., M.LEÃO, R. M., E MARIE, R. Efficient Solutions for an Approximation Technique for the Transient Analysis of Markovian Models. Relatório Técnico 11, Institut National de Recherche en Informatique et en Automat ique, Centre de Diffusion, INRIA, BP 105 - 78153 Le Chesnay Cedex, France, 1996.

[14] DE SOUZA E SILVA, E., E MUNTZ, R. R. Computational Solution Methods for Markov Chains:Applications to Computer an Communication Systems.

[15] DINIZ, M. C. *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.

[16] KLEINROCK, L. *Queueing Systems* . Wiley-Interscience Publication, 1975.

[17] LAND. URL http://www.land.ufrj.br.

[18] M.MEO, E SILVA, E. S., E MARSAN, M. A. Efficient Solution for a Class of Markov Chain Models of Telecommunication Systems. *Performance Evaluation 27&28* (Outubro 1996), 603–625.

[19] Q.R.PATTIPATI, E H.KOBAYASHI. Transient Solutions for the Buffer Behavior in Statistical Multiplexing. *Performance Evaluation 23* (1995), 65–87.

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

[21] Ross, S. M. Approximating Transition Probabilities and Mean Ocupation Times in Continuos-time Markov Chains. *Probability in the Engineering and Informational Sciences* (1987), 251–264.

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

[23] Skelly, P., Schwartz, M., e Dixit, S. A Histogram-Based Model for Video Traffic Behavior in an ATM Multiplexer. *IEEE/ACM Transactions on Networking 1*, 4 (August 1993), 445–459.

[24] Tanaka, T., O.Hashida, e Y.Takahashi. Transient Analysis of Fluid Model for ATM Statistical Multiplexer. *Performance Evaluation 23* (1995), 145–162.

[25] Trivedi, K. S. *Probability and Statistics with Reliability, Queuing and Computer Science Applications*. Prentice-Hall, 19XX.

[26] W.K.Grassmann, M.I.Taksar, e D.P.Heyman. Regenerative Analysis and Steady State Distributions for Markov Chains. *Operation Research 33* (1985), 1107–1116.

Apêndice A

Modelos

Neste apêndice apresentamos os modelos utilizados nos diversos exemplos apresentados no decorrer da tese. Todos os sistemas foram modelados utilizando a ferramenta TANGRAM-II.

A.1 Canal Multiplexador

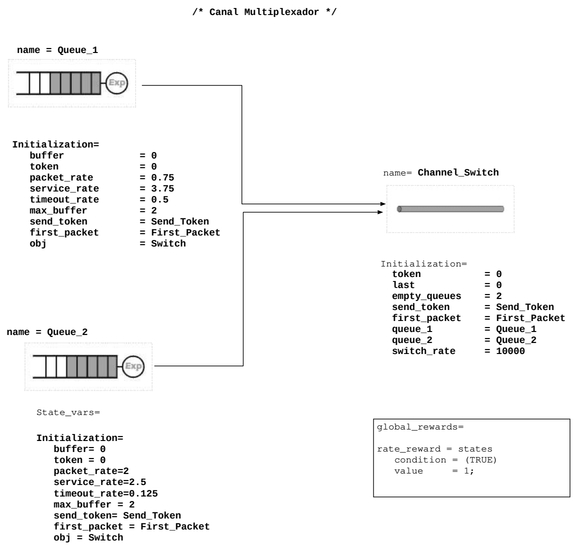


Figura A.1: Modelo 1: Canal Multiplexador.

A.2 *Vacation Servers*

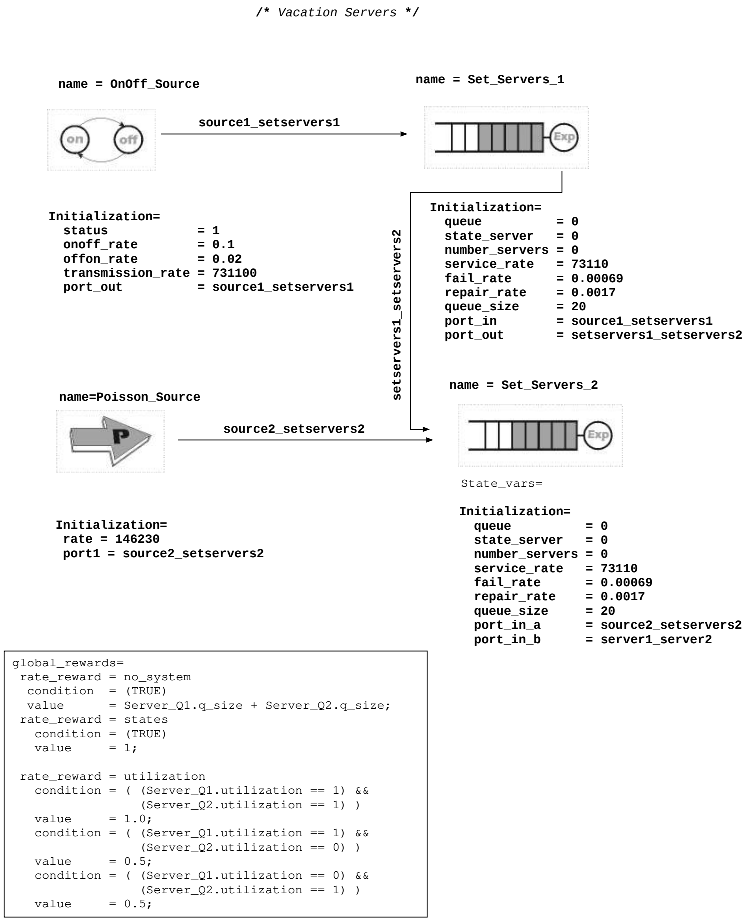


Figura A.2: Modelo 2: *Vacation Servers*.

A.3 Servidor Erlangiano

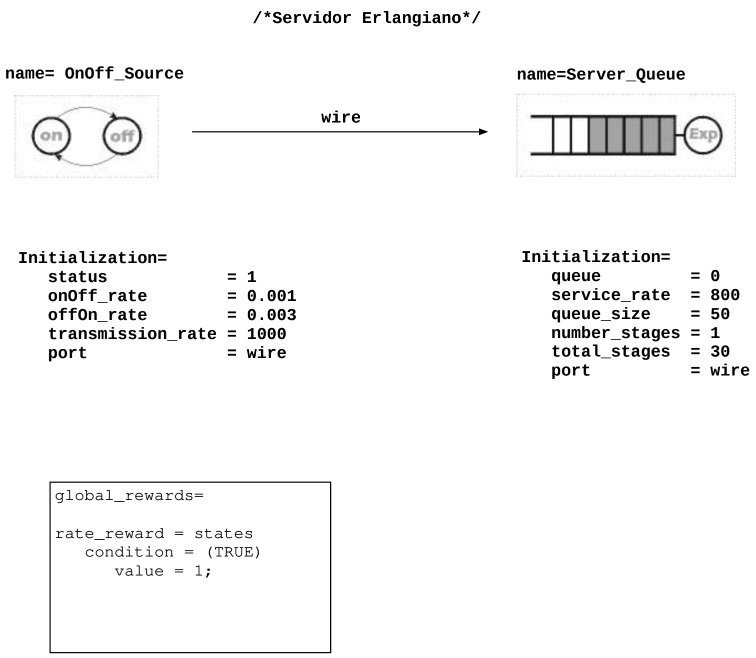
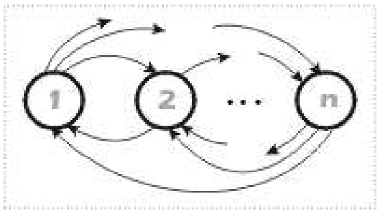


Figura A.3: Modelo 3: Servidor Erlangiano.

A.4 Modelagem de Tráfego - Seqüência I

```
name = Source
```



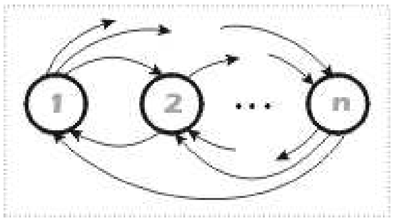

```
Initialization=

 level        = 1
 uniform_rate = 30
 out_port     = link
 obj_dest     = Server_Exp_Queue
 service_rate = 10000
 rate_1       = 18293
 rate_2       = 15882
 rate_3       = 13482
 rate_4       = 11072
 rate_5       = 8672
 rate_6       = 6261
 rate_7       = 3861
 rate_8       = 1450
```

Figura A.4: Modelo 4:Modelagem de Tráfego - Seqüência I .

A.5 Modelagem de Tráfego - Seqüência II

```
name = Source
```




```
Initialization=

 level        = 1
 uniform_rate = 30
 out_port     = link
 obj_dest     = Server_Exp_Queue
 service_rate = 10000
 rate_1       = 13435
 rate_2       = 11677
 rate_3       = 9906
 rate_4       = 8141
 rate_5       = 6377
 rate_6       = 4612
 rate_7       = 2848
 rate_8       = 1083
```

Figura A.5: Modelo 5:Modelagem de Tráfego - Seqüência II .