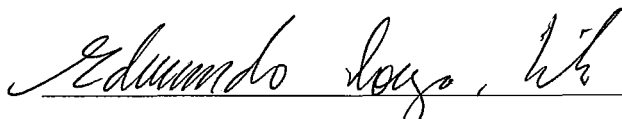


# Técnicas de Solução para Modelos Provenientes de Redes Multimídias

Morganna Carmem Diniz

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 DOUTOR EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

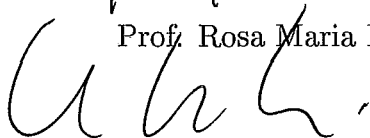
Aprovada por:



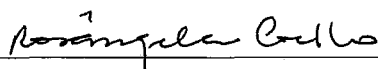
Prof. Edmundo de Souza e Silva, Ph.D.



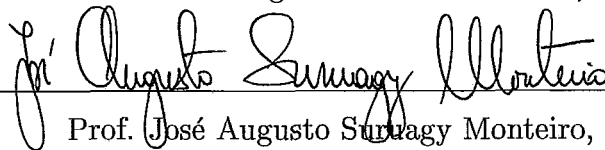
Prof. Rosa Maria Meri Leão, Dr.



Prof. Paulo Henrique de Aguiar Rodrigues, Ph.D.



Prof. Rosângela Fernandes Coelho, Dr.



Prof. José Augusto Surragy Monteiro, Ph.D.

RIO DE JANEIRO, RJ - BRASIL

FEVEREIRO DE 2000

DINIZ, MORGANNA CARMEM

Técnicas de Solução para Modelos Provenientes de Redes Multimídias [Rio de Janeiro] 2000

XIII, 180 p. 29,7 cm (COPPE/UFRJ, D.Sc., Engenharia de Sistemas e Computação, 2000)

Tese - Universidade Federal do Rio de Janeiro, COPPE

1. Redes Multimídias
2. Jitter
3. Métodos de Solução para Modelos Markovianos e Não Markovianos

I. COPPE/UFRJ      II. Título (Série)

*Aos meus pais.*

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

# Técnicas de Solução para Modelos Provenientes de Redes Multimídias

Morganna Carmem Diniz

Fevereiro/2000

Orientador: Edmundo de Souza e Silva

Programa: Engenharia de Sistemas e Computação

A modelagem de sistemas de computação/comunicação é uma das tarefas mais importantes no processo de análise e desenvolvimento de novas tecnologias. Por este motivo, é fundamental o desenvolvimento de modelos que aproximem o comportamento de sistemas reais e de métodos de solução eficientes que possibilitem a obtenção de medidas de interesse destes sistemas. O objetivo desta tese é estudar novos modelos provenientes de redes multimídia e propor soluções gerais ou particulares para esses modelos. Dividimos este trabalho em três partes. A primeira parte focaliza o estudo de *jitter* como um exemplo de problema comum para a transmissão de voz e vídeo em tempo real. Neste trabalho é feita a modelagem e a análise de duas propostas de controle do *jitter* no destino. A segunda parte estuda métodos de solução em estado estacionário de modelos de rede e apresenta uma aproximação para o método de solução GTH de forma que ele possa ser usado na solução de matrizes com uma determinada estrutura especial e com milhares de estados. A terceira parte faz o estudo transiente de modelos com recompensas que têm aplicação em modelos de fluido para redes. Neste trabalho é feito o estudo dos algoritmos propostos em [52] e [54] para o cálculo da distribuição da recompensa acumulada em um tempo finito. O objetivo é fornecer uma interpretação probabilística para o algoritmo de [54] usando a mesma metodologia de [49]. Esta interpretação é importante de forma a entender a aplicabilidade do algoritmo. Além disso, apresentamos dois novos algoritmos para combinação linear de estatística de ordem obtidos como subproduto do estudo feito.



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

# Solution Techniques for Models Arising from Multimedia Networks

Morganna Carmem Diniz

February/2000

Advisor: Edmundo de Souza e Silva

Department: Computer and System Engineering

The modeling of computer and communication systems is a fundamental step in the analysis and development of new technologies. Therefore, it is important to develop not only mathematical models which try to approximate the behavior of real systems but also efficient solution methods to obtain the measures of interest. The aim of this thesis is to study new models arising from multimedia networks and to develop efficient solutions for these. The work is divided in three parts. First, we study the jitter as an example of a common problem in real-time video and voice transmission. We present two models for analyzing jitter control schemes at user end hosts, and study the efficacy of the approaches. We want to know if the mechanism is efficacy for jitter control. Second, we study steady-steady solution techniques for network models and present an approximation for the GTH algorithm suitable for solving large models that have a special characteristic. Third, we study transient analysis methods for Markovian reward models arising from network models. In this work we study the algorithms [52] e [54] for calculating the Cumulative Reward Distribution over a finite observation period. The aim is to obtain a probabilistic interpretation for the algorithm of [54] from the methodology of [49]. The results provided the basis to extend the applicability of the algorithm. Furthermore, we obtain two new algorithms for linear combination of uniform order statistics.

# Conteúdo

<b>Resumo</b>	<b>iv</b>
<b>Abstract</b>	<b>v</b>
<b>1 Introdução</b>	<b>1</b>
1.1 Objetivos da Tese . . . . .	4
<b>2 Modelagem e Análise de Mecanismos de Controle do <i>Jitter</i></b>	<b>7</b>
2.1 Introdução . . . . .	7
2.2 Controle do Jitter na Rede . . . . .	11
2.3 Controle do Jitter no Destino . . . . .	12
2.3.1 Modelo 1 . . . . .	13
2.3.2 Modelo 2 . . . . .	26
2.4 Resultados . . . . .	30
2.5 Conclusões . . . . .	36

<b>3</b>	<b>Aproximação para o Algoritmo GTH</b>	<b>40</b>
3.1	Introdução . . . . .	40
3.2	Métodos de Solução . . . . .	43
3.2.1	Solução de Modelos Markovianos . . . . .	43
3.2.2	Solução de Modelos Não Markovianos . . . . .	47
3.3	Métodos de Aproximação para a Solução de Modelos Markovianos	49
3.3.1	Cadeias de Markov $\epsilon$ - <i>quasi-lumpable</i> . . . . .	50
3.3.2	Solução para um Sub-conjunto de Estados de um Modelo Markoviano . . . . .	52
3.4	Novo Método: uma aproximação para o algoritmo GTH . . . . .	56
3.4.1	Transições com valores $\epsilon$ . . . . .	59
3.4.2	Algoritmo Proposto . . . . .	64
3.4.3	Melhorando o Algoritmo Proposto . . . . .	73
3.5	Exemplos . . . . .	77
3.5.1	Modelo 1: Jitter . . . . .	78
3.5.2	Modelo 2: Sistema de Polling . . . . .	83
3.5.3	Modelo 3: Servidores de Vídeo com Carga Mista . . . . .	97
3.6	Conclusões . . . . .	108
<b>4</b>	<b>Distribuição da Recompensa Acumulada em um Tempo Finito</b>	<b>110</b>

4.1	Introdução . . . . .	110
4.2	Algoritmos . . . . .	114
4.2.1	Técnica de Uniformização . . . . .	114
4.2.2	Combinação Linear de Estatísticas de Ordem . . . . .	115
4.2.3	Algoritmo 1 . . . . .	120
4.2.4	Processo de Renovação . . . . .	124
4.2.5	Algoritmo 2 . . . . .	125
4.3	Novos Algoritmos de Combinação Linear de Estatísticas de Ordem	128
4.3.1	Algoritmo 1 . . . . .	129
4.3.2	Algoritmo 2 . . . . .	137
4.4	Recursão para $P[ACR(t) > r]$ . . . . .	152
4.4.1	Interpretação Probabilística . . . . .	153
4.4.2	Alteração na Recursão para $P[ACR(t) > r]$ . . . . .	163
4.4.3	Aplicação . . . . .	166
4.5	Conclusões . . . . .	169
<b>5</b>	<b>Conclusões</b>	<b>171</b>

# Lista de Tabelas

3.1	$\Pr[\textit{Jitter} > x]$ para $T_1 = 800$ e $B = 50$ . . . . .	82
3.2	$\Pr[\textit{Jitter} > x]$ para $T_1 = 800$ e $B = 50$ . . . . .	89
3.3	Retardo médio de um pacote de dados. . . . .	94
3.4	Retardo médio de um pacote de voz. . . . .	94
3.5	Retardo médio de um pacote de dados. . . . .	96
3.6	Retardo médio de um pacote de voz. . . . .	97
3.7	Tempo médio de resposta de um pedido NC quando $\varepsilon = 10e - 05$ . . . . .	104
3.8	Tempo médio de resposta de um pedido NC quando $\varepsilon = 10e - 03$ . . . . .	104

# Lista de Figuras

2.1	Transmissão de pacotes de voz através de uma rede. . . . .	8
2.2	Controle do <i>jitter</i> no destino. . . . .	12
2.3	Conexão com $N_{min} = 2$ e $B = 4$ . . . . .	16
2.4	Processo de chegada modelado por um MMPP. . . . .	18
2.5	Processo de chegada modelado por um processo erlangiano. . . . .	24
2.6	Conexão com $r = 3$ , $N_{min} = 2$ e $B = 4$ . . . . .	27
2.7	Conexão com <i>timeout</i> e $B = 4$ . . . . .	28
2.8	Conexão com <i>timeout</i> , $r = 3$ e $B = 4$ . . . . .	29
2.9	Transmissão de pacotes de voz através de uma rede de dados . . .	31
2.10	Distribuição do <i>jitter</i> na entrada do mecanismo de controle. . . .	32
2.11	Distribuição de <i>jitter</i> com $\sigma = 14$ . . . . .	33
2.12	Distribuição de <i>jitter</i> com $\sigma = 21$ . . . . .	33
2.13	Distribuição de <i>jitter</i> com $\sigma = 36$ . . . . .	34
2.14	Dados reais $\times$ Distribuição de <i>jitter</i> com $\sigma = 21$ . . . . .	36

3.1	Grupos de estados. . . . .	45
3.2	Cálculo do elemento $p_{ij}^{(M-1)}$ . . . . .	46
3.3	Matriz $P_s$ . . . . .	51
3.4	Novo Processo $\mathcal{X}'$ . . . . .	53
3.5	Modelo com 2 filas de pacotes. . . . .	53
3.6	Novo processo $\mathcal{X}''$ . . . . .	55
3.7	Algoritmo GTH. . . . .	58
3.8	Transições com valores $\varepsilon$ . . . . .	58
3.9	O estado $s_i$ possui uma única transição de entrada. . . . .	60
3.10	Exemplo de cadeia com transição $\varepsilon$ . . . . .	60
3.11	O estado $s_i$ com transição de saída $\varepsilon$ . . . . .	61
3.12	Todas as transições de saída de $s_2$ têm valor $\varepsilon$ . . . . .	62
3.13	Cadeia de Markov quase particionada. . . . .	63
3.14	Algoritmo GTH. . . . .	64
3.15	Nova matriz $P_s$ . . . . .	65
3.16	Transições $\varepsilon$ são desviadas para um novo estado. . . . .	66
3.17	Processo markoviano descrito por $\mathcal{X}''$ . . . . .	69
3.18	Estados clones do modelo com 2 filas de pacotes. . . . .	71
3.19	Novos estados clones do modelo com 2 filas de pacotes. . . . .	71

3.20	Estrutura da matriz de $\mathcal{X}''$ . . . . .	74
3.21	Estados clones do modelo de <i>jitter</i> . . . . .	80
3.22	Estrutura da Matriz $\mathbf{P}$ do modelo de <i>jitter</i> . . . . .	84
3.23	Estrutura da Matriz $\mathbf{P}'$ do modelo de <i>jitter</i> . . . . .	85
3.24	Estrutura da Matriz $\mathcal{H}$ do modelo de <i>jitter</i> . . . . .	86
3.25	Estrutura da Matriz $\mathcal{H}'$ do modelo de <i>jitter</i> . . . . .	87
3.26	Número de operações requeridas na solução da matriz exponencial para $B = 50$ . . . . .	88
3.27	Número de operações requeridas na solução da matriz exponencial para diferentes valores de $B$ . . . . .	88
3.28	Número de operações requeridas na solução da matriz de pontos embutidos. . . . .	90
3.29	Multiplexador $T_1 - T_2$ . . . . .	90
3.30	Estados clones do modelo de <i>polling</i> . . . . .	93
3.31	Estados clones do modelo de <i>polling</i> . . . . .	96
3.32	Estrutura da Matriz do modelo do Multiplexador $T_1 - T_2$ . . . . .	98
3.33	Estrutura da Matriz $\mathbf{P}$ do modelo do Multiplexador $T_1 - T_2$ . . . . .	99
3.34	Estrutura da Matriz $\mathcal{H}$ do modelo do Multiplexador $T_1 - T_2$ . . . . .	100
3.35	Estrutura da Matriz $\mathcal{H}'$ do modelo do Multiplexador $T_1 - T_2$ . . . . .	101
3.36	Número de operações requeridas na solução da matriz exponencial. . . . .	102



3.37	Número de operações requeridas na solução da matriz de pontos embutidos. . . . .	102
3.38	Matriz $\mathbf{P}$ do Modelo do Servidor de Disco. . . . .	105
3.39	Matriz $\mathcal{H}$ do Modelo do Servidor de Disco. . . . .	106
3.40	Matriz $\mathcal{H}'$ do Modelo do Servidor de Disco. . . . .	107
3.41	Número de operações requeridas na solução da matriz $\mathcal{H}$ . . . . .	108
4.1	Exemplo de estatísticas de ordem com 5 variáveis. . . . .	116
4.2	Combinação linear de estatísticas de ordem $U_{(k)}$ . . . . .	117
4.3	Cálculo de $\mathbf{b}^j(n, k)$ . . . . .	128
4.4	Cálculo dos $\Upsilon_s(n, m, j)$ . . . . .	155
4.5	Ocupação do <i>Buffer</i> . . . . .	169

# Capítulo 1

## Introdução

A modelagem de sistemas de computação/comunicação é uma das tarefas mais importantes no processo de análise e desenvolvimento de novas tecnologias. A possibilidade de estudar o comportamento de um sistema que ainda não existe é uma das grande vantagens do uso da modelagem. Sistemas podem ser propostos e avaliados ainda na fase de projeto, antes de serem implementados. Por este motivo, é fundamental o desenvolvimento de modelos que aproximem o comportamento de sistemas reais e de métodos de solução eficientes que possibilitem a obtenção de medidas de interesse destes sistemas.

Um modelo é na realidade uma visão simplificada do sistema sendo estudado, mas é definido de forma a representar o mais próximo possível o comportamento do sistema real. O nosso interesse neste trabalho é a análise de desempenho de sistemas de comunicação multimídia. Por exemplo, analisar o comportamento do sistema diante da contenção de recursos, ou seja, um ou mais recursos podem ser solicitados por mais de um usuário em um mesmo intervalo de tempo.

Os modelos estocásticos são os mais usados na modelagem e análise de desempenho de sistemas. Um modelo estocástico é definido como um conjunto de variáveis aleatórias indexadas por um parâmetro de tempo. Por exemplo, no

processo estocástico  $\{\mathcal{X}(t) : t \geq 0\}$ , o índice  $t$  é interpretado como tempo e  $\mathcal{X}(t)$  corresponde ao estado do processo no instante  $t$ .

Um processo estocástico é dito markoviano se 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. Os modelos markovianos podem ser de tempo discreto ou de tempo contínuo. No primeiro caso uma transição de estado só ocorre em tempos discretos, enquanto no segundo caso uma transição de estado pode ocorrer em qualquer tempo.

Uma tarefa importante na análise de um modelo é a definição das medidas de interesse a serem calculadas. Medidas de interesse refletem o comportamento do modelo e são utilizadas para avaliar e caracterizar o sistema. Podemos classificar as medidas em dois tipos: medidas em estado estacionário e medidas transientes.

No cálculo das medidas em estado estacionário, o tempo de observação do modelo é muito grande, ou seja,  $t \rightarrow \infty$ . Estas medidas fornecem o comportamento médio do sistema ao longo do tempo. Existem na literatura vários métodos de solução de modelos markovianos em estado estacionário como Gauss, Jacobi, Power, SOR, etc [28]. Essas medidas porém não captam o comportamento do sistema em um intervalo finito de tempo.

Considere, por exemplo, o caso de transmissão de pacotes de voz sobre uma rede de dados. Uma análise do sistema em estado estacionário pode concluir que a probabilidade de perda de pacotes é pequena, o que permite garantir uma melhor qualidade de serviço (QoS) para o usuário. Entretanto, as perdas de pacotes podem se concentrar em determinados instantes da transmissão. Nestes instantes, a QoS oferecida pela rede é baixa, apesar da probabilidade de perda de pacotes se manter em níveis aceitáveis. A alternativa para evitar este tipo de problema é realizar um estudo sobre o comportamento transiente do sistema.

No cálculo das medidas transientes o tempo de observação não é muito grande

e não é possível usar a aproximação  $t \rightarrow \infty$ . Estas medidas fornecem o comportamento do sistema após um período curto de observação e supondo um determinado estado inicial do sistema [27]. Uma maneira de se obter medidas transientes é usar o método de uniformização [28, 34].

Em geral, pode-se atribuir diferentes tipos de recompensa à cadeia de Markov. Por exemplo, pode-se associar a cada estado da cadeia uma taxa de recompensa. Isto significa que por cada unidade de tempo em que o sistema permanece em um determinado estado, ele ganha a recompensa associada a esse estado. Medidas de recompensa são muito usadas na análise de desempenho de sistemas. Para o modelo de transmissão de pacotes de voz sobre uma rede de dados discutido acima, podemos associar a recompensa zero aos estados que possuem zero pacotes na fila e associar recompensa 1 aos outros estados. A distribuição do tempo que o sistema está ocupado transmitindo pacotes pode ser então calculada a partir do modelo de recompensas.

Infelizmente, nem sempre é simples a obtenção de medidas de interesse para um modelo. Um fator que pode dificultar o cálculo é a grande quantidade de estados do modelo. Por exemplo, suponha um modelo com  $10^6$  estados. Solucionar sistemas com tal magnitude não é uma tarefa trivial. Principalmente, se a matriz de taxas de transição do modelo não possui alguma estrutura especial. Uma maneira é fazer uso de truncagem do espaço de estados do modelo [28]. O problema é como calcular o erro introduzido pela truncagem. Na literatura é possível encontrar alguns métodos onde o cálculo do erro introduzido pela truncagem é fornecido. Em geral, cada método proposto fornece a solução apenas para uma classe específica de modelos.

Temos também outro problema relacionado ao uso de cadeias de Markov na modelagem de sistemas. Existem vários exemplos onde alguns dos eventos do modelo não são exponenciais, isto é, a distribuição do tempo transcorrido entre a ativação do evento até a sua execução é *geral*. Por exemplo, modelos de *polling* com períodos de temporização determinísticos e modelos de fila com tempo de

serviço determinístico. Em geral, a técnica de cadeias de Markov embutida é usada para a solução destes modelos. Por exemplo, em [17] é apresentada uma solução para modelos não markovianos que possuem, no máximo, uma única transição não exponencial habilitada em um determinado momento. Em linhas gerais, pontos embutidos são identificados e é construída uma cadeia de Markov embutida  $\mathcal{Y}$  que representa o estado do sistema nestes instantes de tempo. Um algoritmo é apresentado para a obtenção das probabilidades de transição de estados de  $\mathcal{Y}$  usando a solução transiente de modelos. Finalmente as medidas de interesse podem ser calculadas a partir da solução de  $\mathcal{Y}$ .

## 1.1 Objetivos da Tese

O objetivo desta tese é estudar novos modelos provenientes de redes multimídia e propor soluções gerais ou particulares a esses modelos. Por isso, dividimos este trabalho em três partes: a primeira parte focaliza o estudo de *jitter* como um exemplo de um problema que pode ocorrer numa rede multimídia; a segunda parte estuda métodos de solução para modelos em estado estacionário; e a terceira parte faz o estudo transiente de modelos com recompensas. A seguir iremos discutir brevemente cada uma dessas partes e apresentar a organização da tese. Uma discussão mais detalhada dos problemas estudados neste trabalho é fornecida no início de cada capítulo.

As redes multimídias devem suportar uma variedade de serviços com características de tráfego distintas e garantir a QoS requisitada. Um importante serviço solicitado é a especificação do retardo fim-a-fim para os pacotes de uma conexão. Pacotes de uma mesma conexão possuem retardos distintos principalmente devido ao armazenamento nos nós da rede. Limitar o retardo fim-a-fim é um serviço importante principalmente para aplicações em tempo-real, mas não é suficiente para conexões com forte dependência entre pacotes consecutivos como, por exemplo, pacotes de voz. Para esta classe de tráfego, o ideal é limitar a variação do

retardo dos pacotes e não somente o seu valor máximo. Esta variação é chamada de *jitter*.

Basicamente, existem duas propostas para o controle do *jitter*. A primeira proposta define que o controle do *jitter* é um serviço da rede e portanto, cabe à rede garantir a QoS definida durante o estabelecimento da conexão. A segunda proposta define que a rede deve entregar os pacotes o mais rápido possível e que a reorganização do tráfego é da responsabilidade do aplicativo do usuário.

No Capítulo 2 fazemos a modelagem e a análise de duas propostas de controle do *jitter* no destino existentes na literatura. Através da análise desses modelos estudamos se o controle do *jitter* feito apenas no destino é suficiente para recuperar o tráfego de pacotes e garantir a QoS solicitada pelo usuário.

No início deste capítulo dissemos que existem vários métodos de solução para cadeias de Markov em estado estacionário como Gauss, Jacobi, Power, SOR, etc [28]. Um método de solução é dito direto quando fornece as probabilidades exatas dos estados após um número finito de passos. Em geral, métodos diretos são usados quando o número de estados do sistema sendo modelado não é muito grande (da ordem de algumas centenas de estados) e quando a matriz de transição de estados do modelo não é esparsa (poucas transições com valores zero). Um dos métodos diretos mais importantes na solução de cadeia de Markov é o algoritmo GTH [29]. Um problema do algoritmo GTH é o custo computacional elevado para matrizes que não possuem estruturas especiais.

No Capítulo 3 apresentamos uma aproximação para o método de solução GTH de forma que ele possa ser usado na solução de matrizes com uma determinada estrutura especial (a ser discutida no Capítulo 3) e com milhares de estados. Este algoritmo faz uso de conceitos e definições de dois outros algoritmos de aproximação encontrados na literatura. Visando estudar a eficácia do algoritmo proposto, iremos comparar as soluções de alguns modelos markovianos e não markovianos obtidas com o uso do algoritmo GTH e com o novo algoritmo de

aproximação.

Como vimos, o estudo transiente de modelos com recompensa é importante na análise de sistemas [27]. A recompensa ganha pelo modelo corresponde a uma abstração do rendimento obtido pelo sistema durante o intervalo de observação. Uma importante medida de interesse na análise transiente é a distribuição da recompensa acumulada.

No Capítulo 4 estudamos os algoritmos propostos em [52] e [54] para o cálculo da distribuição da recompensa acumulada em um tempo finito. O objetivo é fornecer uma interpretação probabilística para o algoritmo de [54] usando a mesma metodologia de [49]. Além disso, vamos discutir um algoritmo para a distribuição de recompensa acumulada com limites que foi apresentado em [61] e que se tornou possível a partir da interpretação probabilística aqui mostrada. Neste capítulo também são apresentados dois novos algoritmos de combinação linear de estatísticas de ordem que foram obtidos como subproduto do estudo feito. Exemplos do uso destes algoritmos podem ser encontrados em [55] e [58].

Finalmente, no Capítulo 5 são feitas as considerações finais sobre a tese e são propostos alguns temas para trabalhos futuros.

# Capítulo 2

## Modelagem e Análise de Mecanismos de Controle do *Jitter*

### 2.1 Introdução

Aplicativos multimídia estão se tornando cada vez mais populares em redes de computadores. Recursos como vídeo, voz e áudio já são comuns em aplicativos de rede. Portanto, as redes multimídias devem suportar uma variedade de serviços com características de tráfego distintas e oferecer uma qualidade de serviço apropriada para seus aplicativos.

Um importante serviço solicitado é a especificação do retardo fim-a-fim para os pacotes de uma conexão. Esse serviço tem por objetivo garantir que os pacotes sejam entregues ao seu destino dentro de um certo intervalo de tempo, e caso o tempo de permanência de um pacote na rede seja maior que o retardo máximo especificado para a conexão a qual ela pertence, o pacote é considerado perdido. Limitar o tempo de permanência de um pacote na rede não é um serviço fácil de implementar, pois pacotes experimentam retardos aleatórios na rede devido principalmente à espera em filas nos comutadores da rede (nós).



Limitar o retardo fim-a-fim é um serviço importante principalmente para aplicações em tempo-real, mas não é suficiente para conexões com forte dependência entre pacotes consecutivos. Por exemplo, considere um tráfego de voz passando através de uma rede como mostrado na Figura 2.1. Note que a fonte possui períodos ativos e períodos de silêncio [1]. Durante um período ativo pacotes de voz são gerados com taxa  $1/T$ , e durante um período de silêncio nenhum pacote é gerado. Como consequência do retardo aleatório, o intervalo entre pacotes consecutivos na saída da rede pode ser maior ou menor que o intervalo de emissão dos pacotes na fonte. Assim, para recuperar o sinal de voz, os pacotes de voz digitalizada devem ser recebidos a intervalos regulares, com a mesma duração dos intervalos de transmissão, ou seja, o tráfego deve ser reorganizado antes de ser apresentado, ou a voz sairá distorcida. Para esta classe de tráfego, o ideal é que os pacotes da conexão experimentem o mesmo retardo fim-a-fim, ou pelo menos, retardos com valores bem próximos, e é também importante limitar a variação deste retardo e não somente o seu valor máximo. Esta variação é chamada de *jitter*. Quando o intervalo entre pacotes na saída da rede é maior que o intervalo de emissão dos pacotes na fonte, dizemos que o *jitter* é positivo; se é menor, dizemos que o *jitter* é negativo.

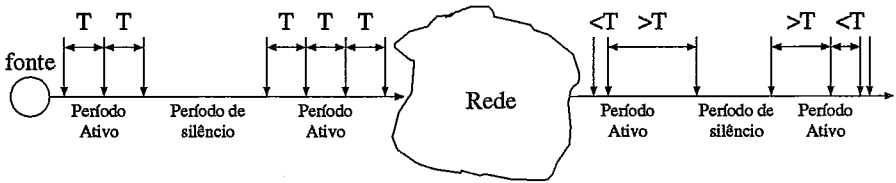


Figura 2.1: Transmissão de pacotes de voz através de uma rede.

Um problema importante consiste em decidir se o controle do *jitter* deve ser ou não um serviço da rede [2]. Argumenta-se que o controle do *jitter* como serviço da rede minimiza a necessidade de *buffers* no destino e simplifica as implementações nos equipamentos dos usuários. Por outro lado, argumenta-se que memória não é cara e que a rede não consegue eliminar completamente o *jitter*, pois pacotes podem experimentar retardos aleatórios no próprio destino, e em consequência,

controle do *jitter* no destino deve ser usado.

Existem na literatura vários artigos que discutem *jitter*. A seguir citaremos alguns exemplos.

- Em [3], Golestani propõe o mecanismo *Stop-and-Go* para controlar o congestionamento de pacotes na rede e garantir um valor máximo para o *jitter*. A idéia é monitorar as conexões em cada nó da rede e garantir que nenhuma conexão exceda o número médio de pacotes definido durante o estabelecimento da conexão.
- Em [4, 5], Guillemin e Roberts estudam o comportamento do *jitter* em uma rede que possui mecanismo *jumping window* [2] ou *leaky bucket* [2] na entrada.
- Em [6], Guillemin e Monin propõem um mecanismo de espaçamento de pacotes, chamado *Spacer-Controller*, nos nós da rede. O objetivo é manter um intervalo mínimo entre os pacotes de um tráfego em cada nó da rede. Os autores mostram através de simulação que o mecanismo diminui o *jitter* negativo e aumenta apenas moderadamente o *jitter* positivo.
- Em [7], Landry e Stavrakakis propõem um mecanismo chamado PORE (*Peak Output Rate Enforcement*) semelhante ao mecanismo *Spacer-Controller* [6] e que também diminui o *jitter* negativo de pacotes de um tráfego periódico. Entretanto, o objetivo do mecanismo não é garantir um intervalo mínimo entre pacotes de uma mesma conexão e sim, garantir que a taxa de entrega dos pacotes no destino não é maior que a taxa de pico definida para o tráfego durante o estabelecimento da conexão.
- Em [8, 9, 10], Ferrari *et al* discutem a implementação de um mecanismo para monitorar o tempo de vida dos pacotes de uma conexão em uma rede ATM. Este mecanismo chamado RCSP (*Rate-Controlled Static-Priority*) é apresentado como uma melhoria no mecanismo JITTER-EDD [2]. Uma

das vantagens atribuídas ao uso do RCSP em relação ao JITTER-EDD é a possibilidade de limitar o valor do *jitter* para os pacotes de uma conexão.

- Em [11], Kröner *et al* estudam o *jitter* de um tráfego não periódico que atravessa um ou mais nós de uma rede ATM. Com este estudo os autores concluíram que, se existe um bom mecanismo de controle do tráfego na entrada da rede, o *jitter* do tráfego no destino é menor que o tempo máximo de permanência dos pacotes do tráfego em um nó da rede. Os resultados analíticos apresentados neste trabalho são validados por simulação.
- Em [12], Matragi *et al* estudam o comportamento de um tráfego periódico após atravessar um comutador ATM, onde o tráfego das outras conexões (tráfego *background*) é modelado como uma sequência de *batches* de pacotes independentes e identicamente distribuídos com uma distribuição qualquer. Em [13, 14], os autores estendem este modelo para um tráfego periódico que atravessa um ou mais comutadores ATM e onde o tráfego *background* tem uma distribuição qualquer. E por último, os autores estudam em [15] o efeito de vários parâmetros de tráfego na distribuição do *jitter*. Estes trabalhos apresentam uma expressão analítica fechada para o cálculo da distribuição do *jitter* dos modelos analisados. O problema é a complexidade da solução apresentada.

O objetivo desse capítulo é modelar e analisar duas propostas de controle do *jitter* no destino existentes na literatura, assumindo que o tráfego sofreu um certo grau de perturbação após atravessar uma rede (veja, por exemplo, [5], [11], [13] e [14] para modelagem do *jitter* na saída de uma rede). Através da análise desses modelos queremos verificar se o controle do *jitter* feito apenas no destino é suficiente para recuperar o tráfego de pacotes e garantir a QoS solicitada pelo usuário. Este trabalho apresenta tem duas contribuições: o desenvolvimento de modelos analíticos para o estudo do controle do *jitter* no destino e o cálculo do *jitter* destes modelos. A organização deste capítulo é a seguinte: na seção 2.2 revemos alguns mecanismos propostos para controle do *jitter* pela rede; na

seção 2.3 analisamos dois modelos de controle do *jitter* pelo destino; na seção 2.4 apresentamos resultados numéricos para os dois modelos; a seção 2.5 conclui o capítulo.

## 2.2 Controle do Jitter na Rede

Como vimos, na literatura existem propostas de mecanismos para controle do *jitter* na rede tais como *Stop-and-Go* [3] *Spacer-Controller* [6], PORE [7] e RCSP [10]. Esses mecanismos monitoram os tráfegos em nós da rede, corrigindo eventuais distorções. A distorção de um tráfego é medida em relação aos parâmetros definidos durante o estabelecimento da correspondente conexão. Por exemplo, pode ser definido um intervalo mínimo  $X_{min}$  entre pacotes consecutivos de uma conexão. Se o tempo entre chegadas de dois pacotes desta conexão for menor que  $X_{min}$ , o segundo pacote é retardado até que o intervalo mínimo seja completado. A monitoração de um tráfego pode ser feita em cada nó da rede por onde o tráfego passa ou em apenas alguns nós.

Embora esses mecanismos consigam manter o *jitter* dentro de certos limites, podemos fazer algumas críticas ao seu uso [3]-[10]:

- *aumento do retardo fim-a-fim*: a idéia básica desses mecanismos consiste em retardar os pacotes de acordo com os parâmetros especificados para a conexão a que pertencem, logo esses mecanismos aumentam o tempo de permanência dos pacotes dentro da rede;
- *desperdício de banda passante*: durante um período de tempo podem não ocorrer transmissões, mesmo que existam pacotes armazenados (as disciplinas não conservam *trabalho*);
- *aumento do tempo de processamento*: a inclusão de mecanismos para verificar o comportamento dos pacotes de uma conexão e de procedimentos

para reconstrução do tráfego original, aumenta o tempo de processamento dos pacotes dentro da rede. No caso de *Stop-and-Go*, PORE e RCSP, a situação é ainda mais crítica, pois a inclusão desses mecanismos é feita em cada nó da rede.

Portanto, mesmo que esses mecanismos sejam eficazes em limitar o *jitter* na saída da rede, o preço a ser pago é o aumento do retardo fim-a-fim e do tempo de processamento dos pacotes de uma conexão dentro da rede.

## 2.3 Controle do Jitter no Destino

Se não existe nenhum mecanismo de controle do *jitter* dentro de uma rede, os pacotes experimentam retardos aleatórios até o destino. A Figura 2.2 ilustra esta situação. Neste exemplo, a fonte gera um pacote a cada intervalo  $T$  durante o período ativo, e é definido que pacotes devem ser entregues ao seu destino com a mesma taxa. Após atravessar a rede, o tráfego não é mais periódico. O intervalo entre pacotes consecutivos pode ser menor ou maior que  $T$ , e é possível ocorrer pequenos períodos com várias chegadas de pacotes, e longos períodos com nenhuma chegada de pacote.

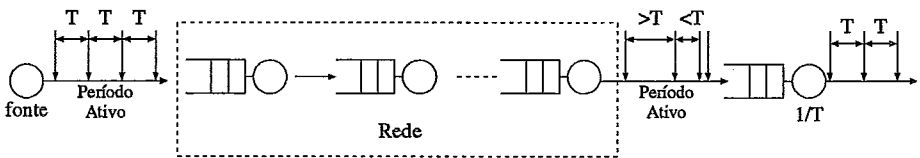


Figura 2.2: Controle do *jitter* no destino.

A idéia básica do controle do *jitter* no destino é armazenar os pacotes da conexão no equipamento do usuário, e entregá-las com taxa  $T$ . Entretanto, armazenar e entregar pacotes com taxa  $T$ , apenas garante que o intervalo entre

pacotes consecutivos não será menor que  $T$ , ou seja, apenas garante que o *jitter* negativo é eliminado. Suponha, por exemplo, que existe um intervalo  $T + k$  entre as chegadas no destino dos dois primeiros pacotes da conexão, onde  $k > 0$ . Se o primeiro pacote encontra a fila de armazenamento vazia, o *jitter* entre os dois pacotes ainda será  $k$ . Para absorver o *jitter* positivo, existem duas possíveis soluções: esperar pela chegada de um dado número de pacotes da conexão no início de um período ativo ou esperar por um certo tempo após a chegada do primeiro pacote de um período ativo, antes de iniciar a entrega dos pacotes ao usuário. A seguir, faremos a modelagem e a análise destas duas propostas de controle do *jitter*.

### 2.3.1 Modelo 1

No primeiro modelo, os pacotes que chegam são colocados em uma fila FIFO que só inicia o serviço após a chegada de  $N_{min}$  pacotes. A escolha de  $N_{min}$  não é trivial. Se  $N_{min}$  é muito pequeno, a fila pode ser esvaziada rapidamente e o *jitter* positivo não será completamente eliminado. Se  $N_{min}$  é muito grande, pode ser necessário alocar uma grande quantidade de espaço em *buffer* para a conexão, causando também um retardo maior na entrega dos pacotes.

Um segundo problema consiste em o que fazer quando a fila é esvaziada após o início da entrega dos pacotes. Neste modelo a entrega dos pacotes é reiniciada com a chegada do próximo pacote. A espera por  $N_{min}$  pacotes só ocorre no início de um período ativo. Esta solução reduz o *jitter* positivo se a probabilidade da fila esvaziar for pequena.

No nosso modelo supomos que é possível detectar a ocorrência de um período de silêncio. Esta suposição é realista, uma vez que existem algoritmos de voz capazes de detectar períodos de silêncio. Após um período de silêncio o algoritmo de *jitter* volta a armazenar  $N_{min}$  pacotes antes de começar a entregar os pacotes à aplicação. Visando simplificar o modelo, também supomos que a detecção de

um período de silêncio é feita logo após a chegada do último pacote do período ativo. Pacotes que estão armazenados quando ocorre um período de silêncio são entregues com taxa fixa logo que o período de silêncio é detectado. Estes pacotes não farão parte dos  $N_{min}$  pacotes do próximo período ativo, caso um novo período ativo seja iniciado antes que todos os pacotes do último período ativo sejam entregues.

O modelo do algoritmo de redução do *jitter* é constituído por uma fila com um único servidor e tempo de serviço determinístico. O servidor desta fila passa a funcionar apenas depois de acumulados  $N_{min}$  pacotes ou após um período de silêncio ser detectado. O processo de chegada caracteriza a distribuição do *jitter* na entrada do mecanismo de controle (saída da rede).

Seja  $B$  o número de *buffers* alocados para a redução de *jitter* da conexão. Um pacote é descartado se, ao chegar, não existir *buffer* disponível. É evidente que  $B$  deve ter um valor adequado de maneira a evitar perda de pacotes por parte do algoritmo de redução do *jitter*. Note que podem existir pacotes de diferentes períodos ativos armazenados no *buffer* em um certo instante: pacotes que chegaram antes do último período de silêncio, e portanto devem ser entregues o mais rápido possível (com taxa fixa  $T$ ), e pacotes do atual período ativo que estão sendo armazenados até que haja uma quantidade igual a  $N_{min}$  na fila.

Vimos que quando o equipamento no destino recebe o primeiro pacote após um período de silêncio, ele armazena o pacote e aguarda a chegada de mais  $N_{min} - 1$  pacotes antes de iniciar a entrega ao usuário. Se um novo período de silêncio é detectado antes que existam  $N_{min}$  pacotes armazenados, a entrega dos pacotes recebidos é iniciada imediatamente. Se a fila é esvaziada após o início da entrega de pacotes, a entrega é reinicializada com a chegada do próximo pacote. O servidor da fila é então modelado por uma cadeia com três estados numerados de 0 a 2. O estado 0 indica que o servidor está ocioso, esperando pelos  $N_{min}$  pacotes. O estado 1 indica que o servidor está ocupado, entregando pacotes à aplicação, onde o tempo de serviço é fixo e igual a  $T$ . O estado 2 indica que

o servidor está ocioso, esperando a chegada de novos pacotes para reiniciar as entregas pois a fila esvaziou durante um período ativo da fonte (não foi detectado o início de um período de silêncio).

Na definição do processo de chegada, precisamos considerar a existência de períodos ativos e períodos de silêncio da fonte. Além disso, também é necessário considerar a variação dos intervalos entre chegadas de pacotes consecutivos em um mesmo período ativo. Esta variação pode ser modelada por uma cadeia de Markov qualquer de maneira a se obter a variância desejada para o intervalo entre chegadas de pacotes de um mesmo período ativo. Por exemplo, podemos escolher a distribuição exponencial. Neste caso, supondo que pacotes são gerados com taxa  $1/T$  nos períodos ativos, a distribuição dos intervalos entre pacotes consecutivos na entrada do mecanismo de redução do *jitter* tem desvio padrão igual a  $T$ .

Objetivando estudar a eficácia do algoritmo de armazenamento no destino, escolhemos dois modelos para representar o processo de chegada de forma a obter uma variância para o *jitter* maior e menor do que a variância de uma distribuição exponencial. O primeiro modelo é um MMPP (*Markov Modulated Poisson Process*) com três estados e será usado para modelar um tráfego cujo *jitter* na entrada do mecanismo de controle tem desvio padrão igual ou maior ao de uma distribuição exponencial. O segundo modelo é um processo Erlangiano com  $r$  estágios e será usado para modelar um tráfego cujo *jitter* na entrada do mecanismo de controle tem desvio padrão menor ou igual ao desvio padrão de uma distribuição exponencial. A seguir estudaremos o comportamento do *jitter* antes e após atravessar o mecanismo de controle para estes dois casos.

### **Processo de chegada de pacotes representado por um MMPP**

O MMPP é um processo de Poisson onde a taxa depende do estado de uma cadeia de Markov [1]. Para representar o processo de chegada de pacotes na entrada do



mecanismo de controle usaremos um MMPP com três estados. O estado 0 indica um período de silêncio, o estado 1 indica que a chegada de pacotes se dá com taxa  $\lambda_1$  inferior à taxa de geração de pacotes na fonte ( $\lambda_1 < 1/T$ ) e distribuição exponencial entre chegadas, e o estado 2 indica chegadas com taxa  $\lambda_2$  superior à taxa da fonte ( $\lambda_2 > 1/T$ ). Dizemos que a fonte está inativa (período de silêncio) quando o estado é 0, e que está ativa nos estados 1 e 2.

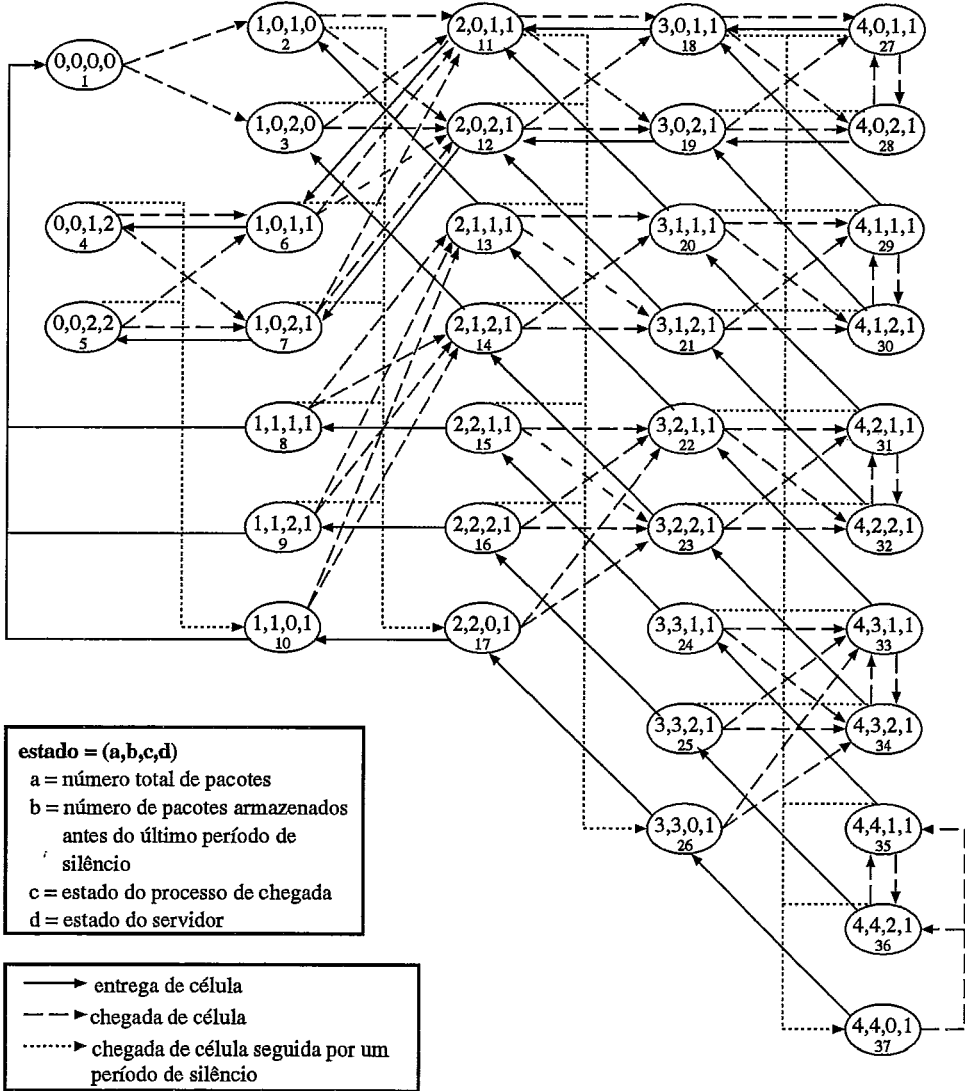


Figura 2.3: Conexão com  $N_{min} = 2$  e  $B = 4$ .

O estado do modelo é portanto caracterizado pelo número total de pacotes na fila, pelo número de pacotes armazenados na fila antes do último período de silêncio, pelo estado do processo de chegadas e pelo estado do servidor. A Figura

2.3 mostra o diagrama de transição de estados para uma conexão com  $N_{min} = 2$  e  $B = 4$ .

Os estados de 1 a 3 na Figura 2.3 indicam que o servidor está esperando por  $N_{min}$  pacotes. Quando o segundo pacote chega, o servidor inicia o processo de entrega. Os estados de 6 a 37 representam o período ocupado do servidor, onde as transições correspondentes à entrega de pacotes são determinísticas (o servidor demora  $T$  unidades de tempo para entregar um pacote). Se a fila esvaziar durante um período ativo da fonte, o modelo transiciona para o estado 4 ou 5, e a entrega de pacotes é reinicializada com a chegada do próximo pacote (a contagem por  $N_{min}$  pacotes só é feita no início de cada período ativo da fonte). Podemos fazer algumas observações em relação a este exemplo:

- um período ativo (a fonte está gerando pacotes) é caracterizado pela geração de pelo menos dois pacotes, onde o primeiro pacote indica o início do período ativo e o último pacote indica o início do período de silêncio;
- como a chegada de novos pacotes é independente da quantidade de pacotes existentes na fila, existe transição do estado da fonte mesmo quando a fila está cheia;
- é possível que um período ativo (estados 1 e 2 do processo de chegada) termine antes que todos os pacotes do período ativo anterior tenham sido entregues. Por exemplo, a transição do estado 14 para o estado 26 indica que um novo período de silêncio foi detectado quando ainda existia um pacote do período ativo anterior para ser entregue.

A seguir vamos estudar o comportamento do tráfego na entrada e na saída do mecanismo de redução do *jitter*. Temos dois objetivos com este estudo. Primeiro, queremos mostrar que podemos obter com o modelo uma variância do *jitter* na entrada do mecanismo maior ou igual a variância de uma distribuição exponencial. E em segundo, queremos calcular a distribuição do *jitter* na saída do mecanismo de controle.

A Figura 2.4 mostra o MMPP com três estados que representa o processo de chegada no modelo que estamos discutindo. A mudança de taxa ocorre imediatamente após a chegada de um pacote. Uma mudança do estado da fonte de 1 para 2 ocorre com probabilidade  $p_{12}$ , e de 2 para 1 com probabilidade  $p_{21}$ . Consequentemente, o intervalo entre chegadas de pacote é hiperexponencial. Com probabilidade  $p_{10}$  o estado da fonte passa de 1 (período ativo) para 0 (período de silêncio), e com probabilidade  $p_{20}$  o estado da fonte passa de 2 (período ativo) para 0 (período de silêncio). Quando o período de silêncio acaba (chega um novo pacote), com probabilidade  $p_{01}$  o estado da fonte vai para 1, e com probabilidade  $1 - p_{01}$  o estado da fonte passa a ser 2. A taxa de chegada de pacotes é  $\lambda_i$  quando o sistema se encontra no estado  $i$ ,  $i = 1, 2$ . Como definido anteriormente, temos  $\lambda_1 < 1/T$  e  $\lambda_2 > 1/T$ .

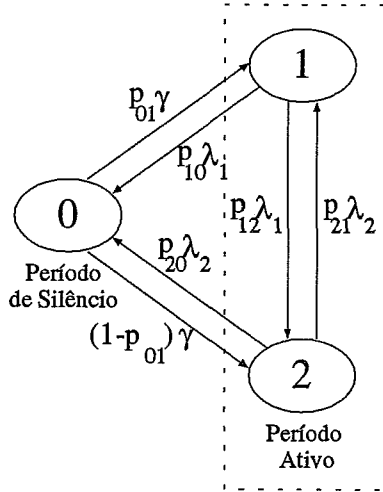


Figura 2.4: Processo de chegada modelado por um MMPP.

Inicialmente, vamos analisar o período de chegada de pacotes, i.e., o período em que a fonte está ativa (estados 1 e 2 da Figura 2.4). Seja  $\lambda$  a taxa média de chegada de pacotes à fila no período ativo, ou seja,

$$\lambda = \lambda_1\pi_1^* + \lambda_2\pi_2^*, \quad (2.1)$$

onde  $\pi_1^*$  é a probabilidade da fonte estar no estado 1 dado que a fonte está ativa,

e  $\pi_2^*$  é a probabilidade da fonte estar no estado 2 dado que a fonte está ativa. Portanto,

$$\pi_1^* = \frac{q\lambda_2}{p\lambda_1 + q\lambda_2} \quad \text{e} \quad \pi_2^* = \frac{p\lambda_1}{p\lambda_1 + q\lambda_2}, \quad (2.2)$$

onde

$$p = p_{12} + p_{10}p_{02} \quad \text{e} \quad q = p_{21} + p_{20}p_{01}.$$

Seja  $X$  a variável aleatória que indica o tempo entre chegadas de pacotes à fila, e  $J_r$  a variável aleatória igual ao *jitter* na saída da rede antes de se aplicar o algoritmo de redução de *jitter*. Então,  $J_r = X - T$ , onde  $T$  é fixo. De 2.1 e 2.2 obtemos o intervalo médio entre chegadas de pacotes no período ativo

$$\begin{aligned} E[X] &= \frac{1}{\lambda} \\ &= \frac{1}{\lambda_1\pi_1^* + \lambda_2\pi_2^*} \\ &= \frac{q\lambda_2 + p\lambda_1}{\lambda_1\lambda_2p + \lambda_1\lambda_2q} \\ &= \frac{q\lambda_2}{\lambda_1\lambda_2p + \lambda_1\lambda_2q} + \frac{p\lambda_1}{\lambda_1\lambda_2p + \lambda_1\lambda_2q} \\ &= \left(\frac{q}{p+q}\right) \frac{1}{\lambda_1} + \left(\frac{p}{p+q}\right) \frac{1}{\lambda_2}, \end{aligned} \quad (2.3)$$

e portanto  $E[J_r] = E[X] - T$ . O tempo entre duas chegadas de pacotes consecutivos é exponencial,  $1/\lambda_i$  corresponde ao intervalo médio entre pacotes quando o sistema se encontra no estado  $i$ ,  $i = 1, 2$ . Além disso,  $q/(p+q)$  e  $p/(p+q)$  correspondem à fração de visitas ao estado 1 e ao estado 2, respectivamente, quando a fonte está ativa. Seja  $Y_i$  uma variável aleatória exponencial com taxa

$\lambda_i$ . Dado que o sistema está ativo e no estado  $i$ , o evento  $J_r > y$  equivale ao evento  $Y_i > y + T$ ,  $y \geq -T$ . Podemos então escrever a função densidade de  $J_r$  como

$$f_{J_r}(y) = \begin{cases} \frac{q}{p+q} \lambda_1 e^{-(y+T)\lambda_1} + \frac{p}{p+q} \lambda_2 e^{-(y+T)\lambda_2} & \text{se } y \geq -T \\ 0 & \text{se } y < -T \end{cases}$$

e portanto, a função distribuição correspondente é

$$F_{J_r}(y) = 1 - \frac{q}{p+q} e^{-(y+T)\lambda_1} - \frac{p}{p+q} e^{-(y+T)\lambda_2}.$$

O primeiro e o segundo momento do *jitter* são dados, respectivamente, por

$$E[Jr] = \frac{q}{p+q} \left( \frac{1}{\lambda_1} - T \right) + \frac{p}{p+q} \left( \frac{1}{\lambda_2} - T \right) \quad (2.4)$$

e

$$E[Jr^2] = \frac{q}{p+q} \left( T^2 - \frac{2T}{\lambda_1} + \frac{2}{\lambda_1^2} \right) + \frac{p}{p+q} \left( T^2 - \frac{2T}{\lambda_2} + \frac{2}{\lambda_2^2} \right). \quad (2.5)$$

Vejamos agora um exemplo de como podemos definir os valores dos parâmetros do modelo de forma a se obter uma distribuição que tem uma determinada variância. Considere uma transmissão de pacotes de voz através de uma rede de dados onde a taxa de perda de pacotes observada é zero (esta suposição é realística para redes pouco congestionadas ou quando existe um algoritmo de recuperação de perdas como em [16]). Neste caso, o *jitter* médio na entrada do mecanismo de controle é zero e a variância é igual ao segundo momento. Podemos então escolher os valores de alguns parâmetros do modelo e usar o primeiro momento para calcular os outros parâmetros. Por exemplo, suponha que  $p/(p+q) = 0.6$  e  $T = 20$ . De (2.4) obtemos  $\lambda_2 = (0.6\lambda_1)/(20\lambda_1 - 0.4)$ . Escolhemos  $1/\lambda_1 < T$  e portanto

precisamos de um valor para  $\lambda_1$  maior que 0.05. Seja  $\lambda_1 = 0.1$ , temos então  $\lambda_2 = 0.0375$ . Podemos agora usar (2.5) e calcular a variância do *jitter* na entrada do mecanismo de controle. Neste exemplo, o desvio padrão é 23.1 e portanto, é maior que o desvio padrão da distribuição exponencial com parâmetro  $\lambda$  igual a  $1/20$ . Observe que precisamos apenas variar  $\lambda_1$  (e naturalmente recalculamos  $\lambda_2$ ) para obtermos valores diferentes para a variância do *jitter*.

No nosso modelo assumimos que o período de silêncio é detectado, e que o intervalo de tempo entre períodos ativos ou de silêncio é exponencialmente distribuído com parâmetro  $\gamma$ . Quando um período de silêncio é detectado, os pacotes que porventura ainda estejam na fila são entregues (e portanto o *jitter* de todos esses pacotes ainda não entregues é zero) e um período de espera por  $N_{min}$  pacotes é iniciado novamente com a chegada do primeiro pacote de um novo período ativo da fonte.

Para calcularmos o *jitter* após a aplicação do algoritmo de redução de *jitter* no destino, precisamos condicionar o valor do *jitter* ao número de pacotes da fila, ao estado do servidor e ao estado da fonte. O *jitter* será zero quando o servidor ( $S$ ) estiver entregando pacotes, independente do estado da fonte. O *jitter* será maior que zero quando o servidor estiver ocioso esperando chegada de pacotes (ver estado 1 do servidor na definição do modelo) em um período ativo (estados 1 e 2 da fonte na definição do modelo). O valor do *jitter*, neste caso, dependerá do estado da fonte ( $F$ ). Note que não entra no cálculo do *jitter* o intervalo de tempo em que o servidor está ocioso devido à espera de  $N_{min}$  pacotes (ver estado 0 do servidor na definição do modelo).

Sejam  $Y$  a variável aleatória que indica o tempo entre a saída do último pacote da fila até a próxima chegada,  $J_s$  a variável aleatória igual ao *jitter* após o mecanismo de controle, e  $N$  a variável aleatória correspondente ao número de pacotes na fila. Além disso, o *jitter*  $J_s$  só está definido imediatamente após o mecanismo de controle começar a entrega de pacotes, portanto para  $S > 0$ . Se  $N$  é maior que zero (fila não vazia), então o espaçamento entre o pacote que

acabou de ser entregue e o próximo pacote é  $T$  e portanto, o *jitter* é zero. Se  $N$  é zero (fila vazia), então o *jitter* é igual a  $Y$ . Seja  $F_{J_s}(y)$  a distribuição do *jitter* após o mecanismo de controle. Então,

$$\begin{aligned}
 F_{J_s}(y) &= P[J_s \leq y] \\
 &= \sum_{i=1}^2 \sum_{j=0}^2 (P[J \leq y | N > 0, S = i, F = j] \times P[N > 0, S = i, F = j] + \\
 &\quad P[J \leq y | N = 0, S = i, F = j] \times P[N = 0, S = i, F = j]) \\
 &= \sum_{i=1}^2 \sum_{j=0}^2 P[J \leq y | N > 0, S = i, F = j] \times P[N > 0, S = i, F = j] + \\
 &\quad \sum_{i=1}^2 \sum_{j=0}^2 P[J \leq y | N = 0, S = i, F = j] \times P[N = 0, S = i, F = j],
 \end{aligned}$$

onde  $y \geq 0$ , uma vez que  $J_s$  é sempre positivo. A probabilidade do servidor ficar ocioso quando existem pacotes na fila e o período de entrega de pacotes já foi iniciado ( $N > 0$  e  $S = 2$ ) é zero, e a probabilidade do servidor entregar pacotes quando a fila está vazia ( $N = 0$  e  $S = 1$ ) também é zero. Além disso, como estamos supondo que o algoritmo é capaz de detectar o início de um período de silêncio, a probabilidade do servidor estar esperando por um pacote ( $S = 2$ ) quando a fila está vazia ( $N = 0$ ) e a fonte está em um período de silêncio ( $F = 0$ ) é zero. Logo,

$$\begin{aligned}
 F_{J_s}(y) &= P[J \leq y | N > 0, S = 1, F = 0] \times P[N > 0, S = 1, F = 0] + \\
 &\quad P[J \leq y | N > 0, S = 1, F = 1] \times P[N > 0, S = 1, F = 1] + \\
 &\quad P[J \leq y | N > 0, S = 1, F = 2] \times P[N > 0, S = 1, F = 2] + \\
 &\quad P[J \leq y | N = 0, S = 2, F = 1] \times P[N = 0, S = 2, F = 1] + \\
 &\quad P[J \leq y | N = 0, S = 2, F = 2] \times P[N = 0, S = 2, F = 2].
 \end{aligned}$$

Como vimos, o *jitter* é zero quando o servidor está entregando pacotes ( $S = 1$ ), independente do estado da fonte. Portanto, para  $y \geq 0$ ,

$$\begin{aligned}
F_{J_s}(y) = & 1 \times P[N > 0, S = 1, F = 0] + \\
& 1 \times P[N > 0, S = 1, F = 1] + \\
& 1 \times P[N > 0, S = 1, F = 2] + \\
& P[J \leq y | N = 0, S = 2, F = 1] \times P[N = 0, S = 2, F = 1] + \\
& P[J \leq y | N = 0, S = 2, F = 2] \times P[N = 0, S = 2, F = 2].
\end{aligned}$$

Se a fila está vazia, o tempo para a chegada do próximo pacote é exponencialmente distribuído com parâmetro  $\lambda_1$  ou com parâmetro  $\lambda_2$  dependendo se o estado da fonte é 1 ou 2. Portanto, a distribuição do *jitter*  $J_s$  na saída do *buffer* é dada por

$$\begin{aligned}
F_{J_s}(y) = & P[N > 0, S = 1, F = 0] + \\
& P[N > 0, S = 1, F = 1] + \\
& P[N > 0, S = 1, F = 2] + \\
& (1 - e^{-\lambda_1 y}) \times P[N = 0, S = 2, F = 1] + \\
& (1 - e^{-\lambda_2 y}) \times P[N = 0, S = 2, F = 2] \\
= & \beta + \beta_1(1 - e^{-\lambda_1 y}) + \beta_2(1 - e^{-\lambda_2 y}),
\end{aligned}$$

onde  $\beta$  é a probabilidade de existir pelo menos um pacote na fila e o processo de entregas de pacotes estar em andamento,  $\beta_1$  é a probabilidade da fila estar vazia, o estado da fonte ser 1 e o servidor estar à espera de novos pacotes para reiniciar as entregas, e  $\beta_2$  é a probabilidade da fila estar vazia, o estado da fonte ser 2 e o servidor estar à espera de novos pacotes para reiniciar as entregas. Note que  $\beta$ ,  $\beta_1$  e  $\beta_2$  estão normalizadas já que estão condicionadas na probabilidade do sistema já ter iniciado a entrega dos pacotes ( $S \neq 0$ ).

### Processo de chegada de pacotes representado por um processo erlangiano

Para obtermos valores menores ou iguais ao desvio padrão de uma distribuição exponencial podemos substituir o MMPP por um processo Erlangiano com  $r$



estágios. Neste caso, o intervalo total entre pacotes consecutivos corresponde à soma de  $r$  variáveis aleatórias identicamente distribuídas, e o desvio padrão equivale a  $1/\sqrt{r}$  do desvio padrão da distribuição exponencial correspondente.

A Figura 2.5 representa o novo processo de chegada dos pacotes na fila. O estado 0 corresponde a um período de silêncio da fonte e os estados de 1 a  $r$  correspondem a um período ativo da fonte. Um pacote só é gerado após o sistema percorrer os estados de 1 a  $r$  do modelo. Quando a fonte se encontra no estado  $r$  e uma transição ocorre, com probabilidade  $p$  é detectado o início de um período de silêncio e com probabilidade  $1 - p$  a fonte continua gerando pacotes. O tempo de duração de um período de silêncio é exponencialmente distribuído com parâmetro  $\gamma$ .

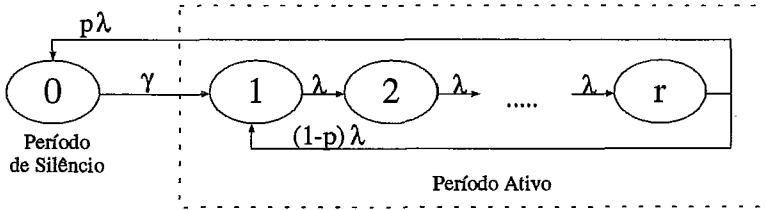


Figura 2.5: Processo de chegada modelado por um processo erlangiano.

A distribuição do *jitter* na entrada do mecanismo de controle corresponde à distribuição erlangiana com  $r$  estágios, ou seja,

$$F_{J_r}(x) = 1 - \sum_{j=0}^{r-1} e^{-\lambda(x+T)} \frac{(\lambda(x+T))^j}{j!},$$

onde  $\lambda = r/T$  (isto nos fornece a média de um pacote a cada  $T$  unidades de tempo).

A distribuição do *jitter* após o mecanismo de controle é obtida de forma semelhante à discutida acima para o processo de chegada representado por um MMPP. A diferença é que agora temos uma fonte com  $r$  estágios no período ativo. Portanto, precisamos condicionar o valor do *jitter* ao número de pacotes da fila, ao estado do servidor e ao número do estágio da fonte erlangiana.

$$\begin{aligned}
F_{J_s}(y) &= P[J \leq y] \\
&= \sum_{i=1}^2 \sum_{j=0}^r (P[J \leq y | N > 0, S = i, F = j] \times P[N > 0, S = i, F = j] + \\
&\quad P[J \leq y | N = 0, S = i, F = j] \times P[N = 0, S = i, F = j]) \\
&= \sum_{i=1}^2 \sum_{j=0}^r P[J \leq y | N > 0, S = i, F = j] \times P[N > 0, S = i, F = j] + \\
&\quad \sum_{i=1}^2 \sum_{j=0}^r P[J \leq y | N = 0, S = i, F = j] \times P[N = 0, S = i, F = j] \\
&= \sum_{j=0}^r P[J \leq y | N > 0, S = 1, F = j] \times P[N > 0, S = 1, F = j] + \\
&\quad \sum_{j=1}^r P[J \leq y | N = 0, S = 2, F = j] \times P[N = 0, S = 2, F = j].
\end{aligned}$$

Se o número de pacotes na fila ( $N$ ) é maior que zero então o espaçamento entre o pacote que acabou de ser entregue e o próximo pacote é  $T$  e portanto, o *jitter* é zero. Se  $N$  é zero, então o *jitter* corresponde ao intervalo até a chegada do próximo pacote. Sabemos que este intervalo tem distribuição erlangiana. Logo, o *jitter* corresponde à soma de  $i$  variáveis aleatórias onde  $i$  é o número do estágio em que a fonte se encontra,  $i = 1, \dots, r$ .

$$\begin{aligned}
F_{J_s}(y) &= \sum_{j=0}^r 1 \times P[N > 0, S = 1, F = j] + \\
&\quad \sum_{j=1}^r \left( 1 - \sum_{i=0}^{r-j} \left( e^{-\lambda y} \frac{(\lambda y)^i}{i!} \right) \right) \times P[N = 0, S = 2, F = j] \\
&= \beta + \sum_{j=1}^r \beta_j \left( 1 - \sum_{i=0}^{r-j} \left( e^{-\lambda y} \frac{(\lambda y)^i}{i!} \right) \right),
\end{aligned}$$

onde  $\beta$  é a probabilidade de existir pelo menos um pacote na fila e o processo de entregas de pacotes estar em andamento,  $\beta_j$  é a probabilidade da fila estar vazia, o estado da fonte ser  $j$  e o servidor estar à espera de novos pacotes para reiniciar as entregas.

A Figura 2.6 mostra o diagrama de transição de estados para um exemplo onde a chegada dos pacotes é representada por um processo erlangiano de 3 estágios

com  $N_{min} = 2$  e  $B = 4$ . Nos estados de 1 a 4 o servidor está esperando por  $N_{min}$  pacotes. Quando o segundo pacote chega (estado 15), o servidor inicia o processo de entrega. Os estados de 8 a 53 representam o período ocupado do servidor. Os estados 5, 6 e 7 representam o período em que a fila está vazia e o servidor está à espera da chegada de novos pacotes para reiniciar as entregas.

### 2.3.2 Modelo 2

No segundo modelo, semelhante ao primeiro, os pacotes que chegam são colocados em uma fila FIFO, mas o processo de entrega de pacotes só é inicializado após um tempo *fixo*  $T_1$  da chegada do primeiro pacote (este tipo de mecanismo é usado, por exemplo, como parte do algoritmo do *Space-Controller* [6]). A escolha de  $T_1$  apresenta problemas semelhantes aos da escolha de  $N_{min}$  no primeiro modelo. Se  $T_1$  é muito pequeno, o número de pacotes armazenados antes de começar a entrega pode ser pequeno, e a fila é esvaziada rapidamente. Se  $T_1$  é muito grande, o número de pacotes armazenados pode ser grande, e uma grande quantidade de *buffers* é necessário. Além disso o retardo para a entrega de pacotes após um período de silêncio aumenta com  $T_1$ . Valores altos de  $T_1$  afetam negativamente a qualidade do serviço oferecida pela rede.

Se a fila esvazia, assumimos que as entregas de pacotes reiniciam com a próxima chegada de um pacote. O tempo de entrega de um pacote é fixo e igual a  $T_2$ . As Figuras 2.7 e 2.8 mostram o diagrama de transição de estados de exemplos com  $B = 4$ , onde o primeiro tem processo de chegada representado por um MMPP, e o segundo exemplo tem processo de chegada representado por um processo erlangiano com  $r = 3$ .

Observe que agora o servidor é caracterizado por quatro estados numerados de 0 a 3. O servidor está no estado 0 quando a fila está vazia e ainda não foi detectado o início de um período ativo da fonte. O servidor está no estado 1 quando o sistema está aguardando o término do tempo  $T_1$  após a chegada do

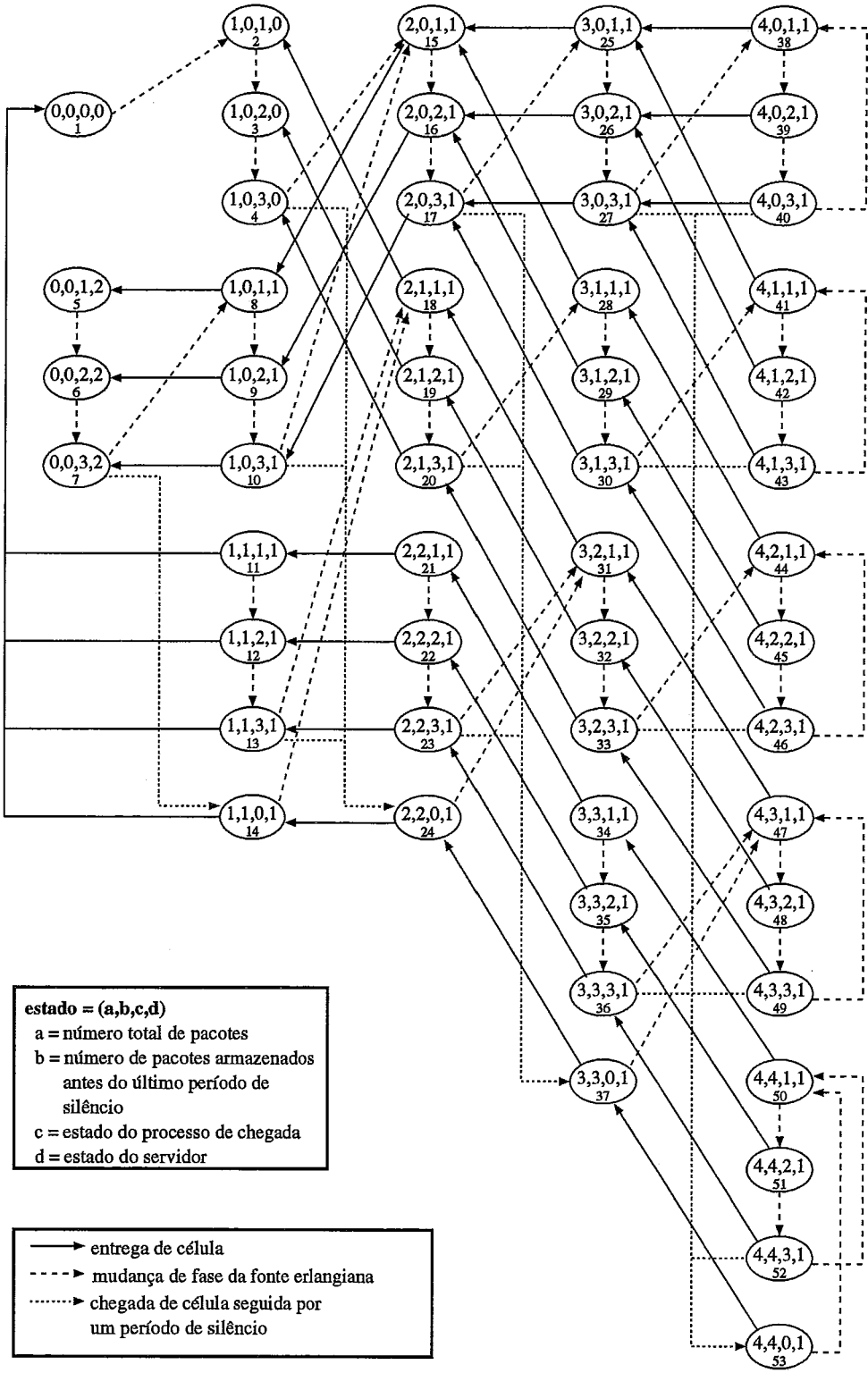


Figura 2.6: Conexão com  $r = 3$ ,  $N_{min} = 2$  e  $B = 4$ .

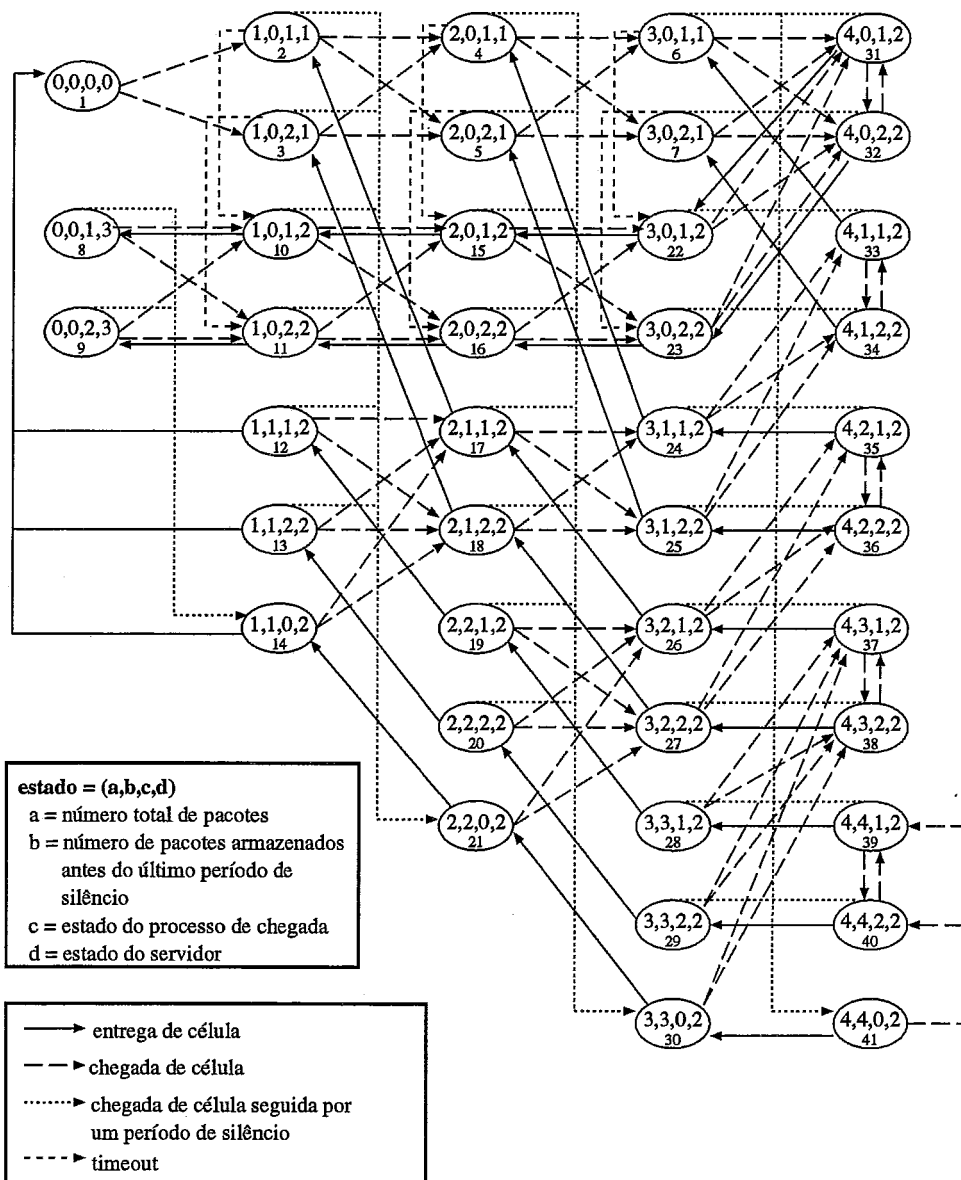


Figura 2.7: Conexão com *timeout* e  $B = 4$ .

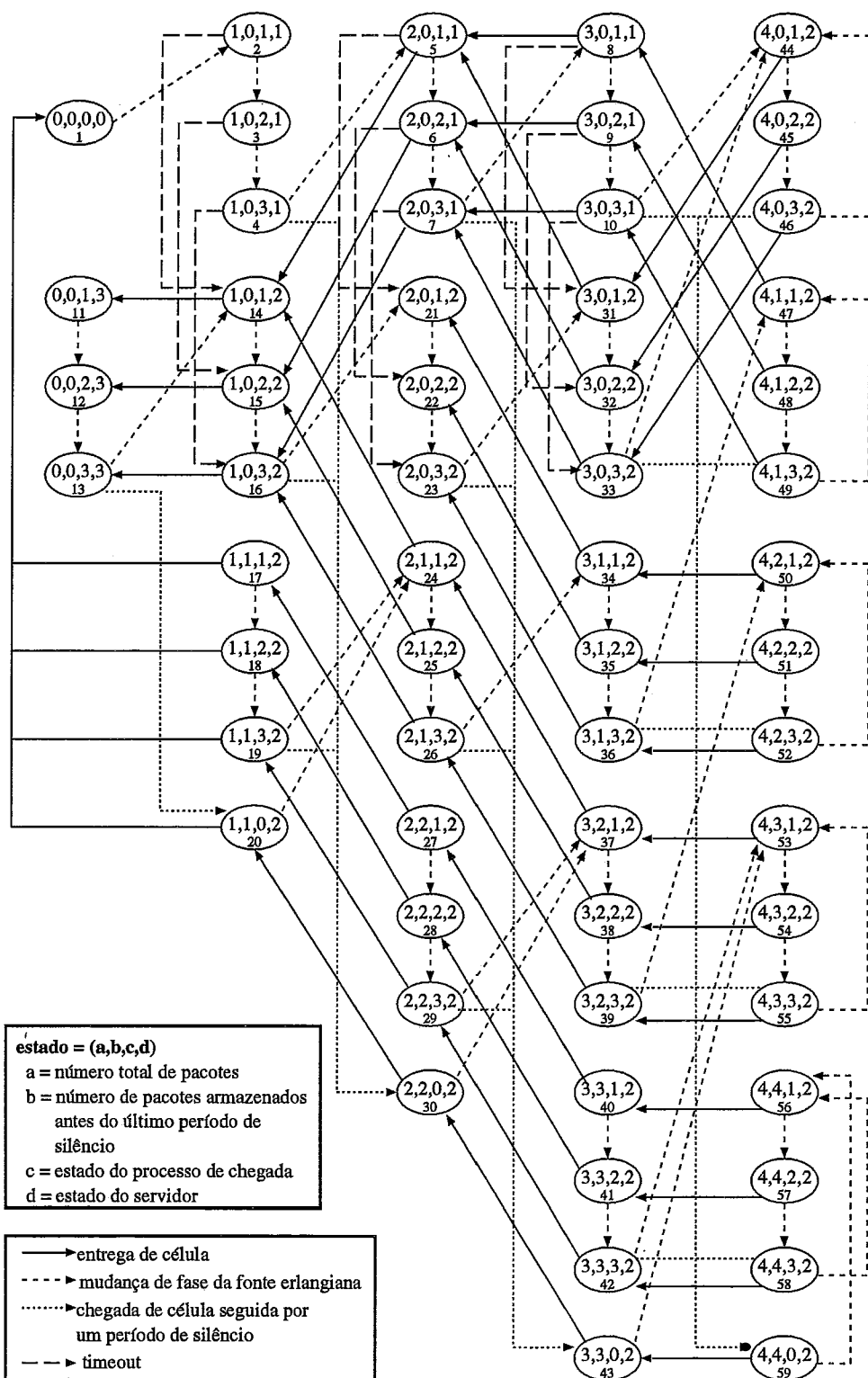


Figura 2.8: Conexão com *timeout*,  $r = 3$  e  $B = 4$ .

primeiro pacote, portanto, a fila não está vazia mas o servidor está esperando para começar a entrega de pacotes. O servidor está no estado 2 quando está entregando pacotes. E por último, o servidor está no estado 3 quando está à espera da chegada de novos pacotes para reiniciar a entrega, pois a fila esvaziou quando a entrega de pacotes já tinha sido iniciada. As suposições feitas no modelo anterior para o processo de chegada e para os períodos de silêncio são também adotadas neste modelo.

## 2.4 Resultados

Para ilustrar a eficácia do algoritmo de armazenamento no destino, vamos inicialmente analisar alguns exemplos de tráfego de voz com diferentes graus de distorção após atravessar um ou mais nós de uma rede. No final, faremos uma comparação desses modelos com os resultados de testes de transmissão de voz feitos entre o Departamento de Ciência da Computação da UCLA e o NCE/UFRJ e que foram apresentados em [16].

É importante notar que os modelos discutidos em 2.3 *não são markovianos*. No primeiro modelo, o tempo de serviço é determinístico e igual a  $T$ . No segundo modelo, tanto o tempo de espera  $T_1$  após a chegada do primeiro pacote de um período ativo quanto o tempo de serviço  $T_2$  são determinísticos. Em [17], é apresentado um algoritmo eficiente para uma classe de modelos não markovianos, onde uma única transição não exponencial pode estar habilitada por vez como no caso dos modelos acima (discutiremos este algoritmo com mais detalhes no Capítulo 3). A seguir, apresentamos alguns resultados obtidos para os modelos acima usando o algoritmo de [17] e implementado de acordo com a Metodologia Orientada a Objetos proposta em [18] como parte de uma ferramenta de modelagem [19] [20] [21].

Considere um aplicativo de voz como mostrado na Figura 2.9. Pacotes de voz

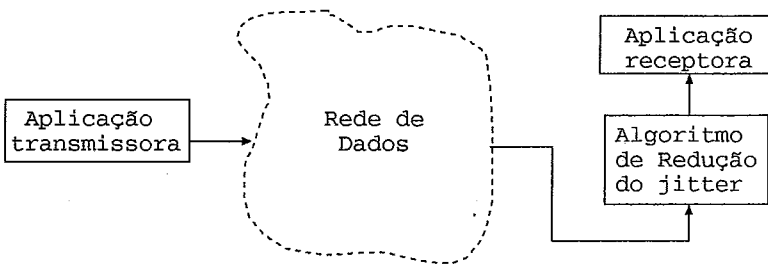


Figura 2.9: Transmissão de pacotes de voz através de uma rede de dados

são gerados pela aplicação transmissora e entregues ao nó mais próximo da rede de dados. Após atravessar a rede, os pacotes são armazenados e tratados por um mecanismo de controle do *jitter* antes de serem entregues ao usuário. Nenhum mecanismo para controle do *jitter* é usado na rede. Suponha que a fonte gera pacotes de voz com taxa constante de  $1/20$  unidades de tempo (este valor é usado em [16] e foi definido através de testes). A Figura 2.10 mostra as distribuições do *jitter* para este tráfego na entrada do algoritmo de redução do *jitter* quando o desvio padrão ( $\sigma$ ) do *jitter* é 14, 21 e 36. O processo de chegada foi representado por um processo erlangiano no primeiro caso, e por um MMPP nos dois outros casos (o desvio padrão para um processo de chegada representado por uma distribuição exponencial é 20). Usamos os seguintes parâmetros para modelar as funções de distribuição da Figura 2.10 de maneira a ter um largo espectro de desvio padrão para o *jitter* na entrada do algoritmo de redução do *jitter*:

- Para  $\sigma = 14$  :  $r = 2$  e  $\lambda = 0.10$ ;
- Para  $\sigma = 21$  :  $\lambda_1 = 0.04$ ,  $\lambda_2 = 0.0667$ ,  $p_{12} = 0.2$ ,  $p_{21} = 0.2$  e  $b = 0.5$ ;
- Para  $\sigma = 36$  :  $\lambda_1 = 0.0222$ ,  $\lambda_2 = 0.5$ ,  $p_{12} = 0.01$ ,  $p_{21} = 0.0002$  e  $b = 0.5$ .

Na análise do modelo usamos os valores apresentados em [22] para a duração do período ativo e do período de silêncio, 400 ms e 600 ms, respectivamente. Portanto, temos  $\gamma = 0.00167$ ,  $p_{10} = p_{20} = 0.005$  para o primeiro modelo, e  $p = 0.005$  para o segundo modelo. A taxa média de chegada de pacotes é  $1/20$ , e o *jitter* médio é zero pois supomos que não há perda de pacotes. A quantidade de



*buffers* alocados para o algoritmo de redução do *jitter* é 50 de maneira a tornar desprezível a probabilidade de perda de pacotes no *buffer* de armazenamento. Note que a curva com desvio padrão 14 cai a zero mais rapidamente do que as outras curvas. Por exemplo, a probabilidade do *jitter* ser maior que 40 unidades de tempo é quase zero para a curva com desvio padrão 14, enquanto ela é em torno de 0.01 para a curva com desvio padrão 21, e em torno de 0.1 para o desvio padrão 36.

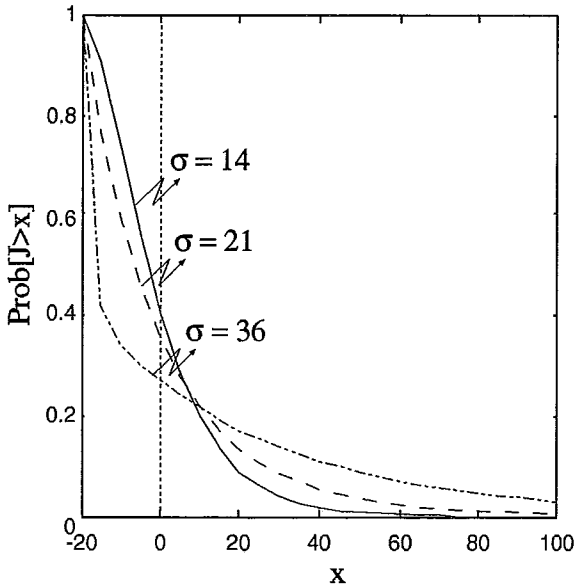


Figura 2.10: Distribuição do *jitter* na entrada do mecanismo de controle.

As Figuras 2.11 e 2.12 mostram os resultados do controle do *jitter* no destino tomando como base as entradas com desvio padrão 14 e 21, respectivamente. Utilizamos  $N_{min} = 5$  no primeiro modelo, e  $T_1 = 100$  no segundo modelo. Note que  $T_1$  foi escolhido de maneira a se obter um número médio de pacotes recebidos da rede igual a  $N_{min}$ . Nestes exemplos, o *jitter* é quase totalmente eliminado nos dois modelos. Por exemplo,  $Prob[J > 0]$  é aproximadamente 0.4 na entrada do algoritmo de redução do *jitter* para os dois exemplos, enquanto é aproximadamente 0.03 após o mecanismo de controle no primeiro exemplo, e aproximadamente 0.05 para o segundo exemplo.

A Figura 2.13 mostra o resultados do controle do *jitter* no destino para a

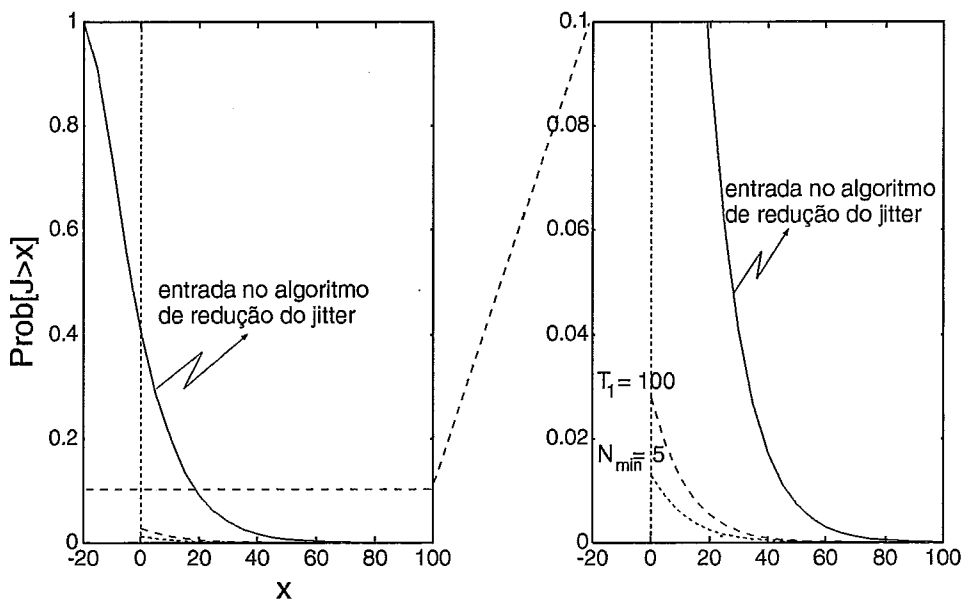


Figura 2.11: Distribuição de *jitter* com  $\sigma = 14$ .

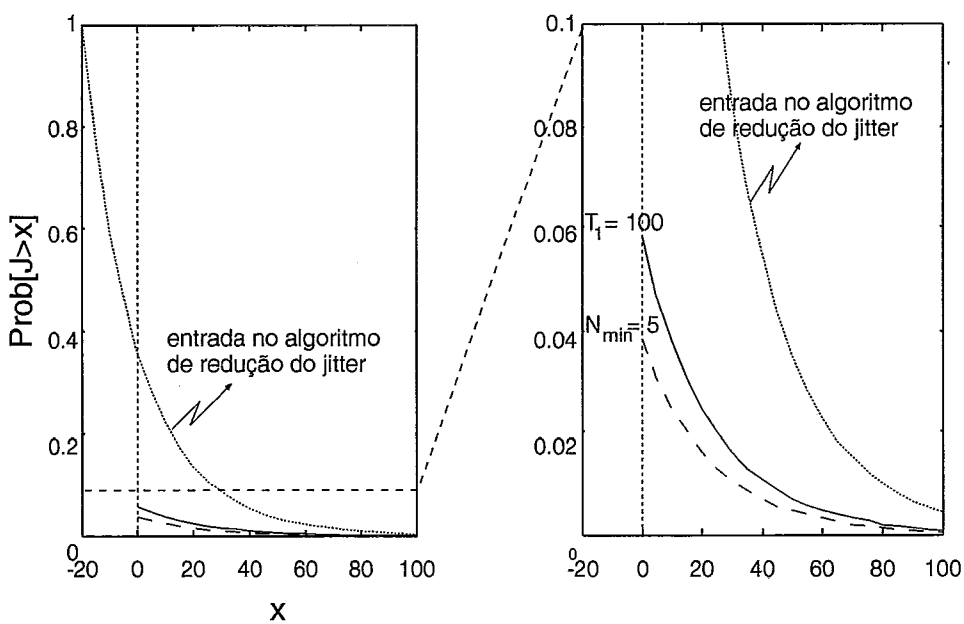


Figura 2.12: Distribuição de *jitter* com  $\sigma = 21$ .

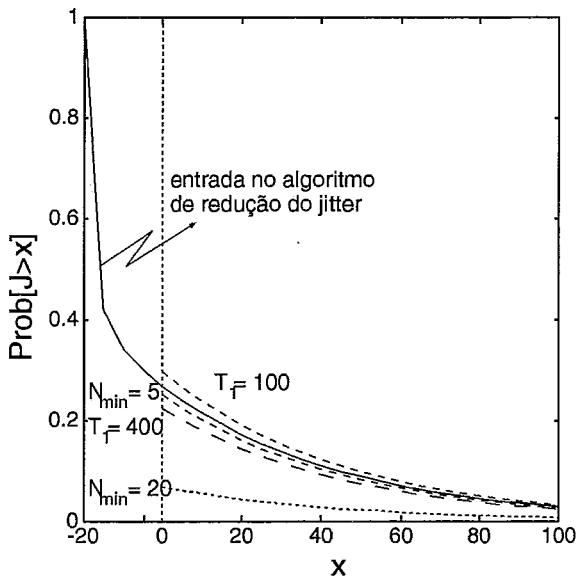


Figura 2.13: Distribuição de *jitter* com  $\sigma = 36$ .

saída com desvio padrão de *jitter* 36. Os valores usados para  $N_{min}$  são 5 e 20, e os valores usados para  $T_1$  são 100 e 400. No primeiro modelo, o *jitter* só tem uma redução significativa quando  $N_{min} = 20$ . No segundo modelo,  $Prob[J > x]$  (no intervalo plotado) é maior que a correspondente probabilidade para o tráfego recebido quando  $T_1 = 100$ . Isto significa que embora o *jitter* negativo tenha sido eliminado, houve um acréscimo na probabilidade de ocorrência de *jitter* positivo. Uma explicação para esse comportamento é a seguinte: o uso do mecanismo para redução do *jitter* apenas eliminou o *jitter* negativo, o algoritmo não conseguiu diminuir o *jitter* positivo, como agora só temos *jitter* com valor maior ou igual a zero, a probabilidade de um *jitter* positivo é maior. Quando  $T_1 = 400$ , o *jitter* positivo é reduzido muito pouco em relação ao *jitter* na entrada do mecanismo de controle. Isto significa que o número de pacotes armazenados durante  $T_1$  não é suficiente para absorver o *jitter* positivo, e portanto, é necessário definir valores maiores para  $T_1$ .

Existem dois parâmetros a serem escolhidos para cada modelo: o tamanho da fila de *buffers* e o valor de  $N_{min}$  para o modelo 1 e de  $T_1$  para o modelo 2. O tamanho da fila é escolhido de maneira a se obter uma probabilidade de bloqueio

dentro de limites aceitáveis. Dizemos que  $N_{min}$  e  $T_1$  são *correspondentes* se eles fornecem uma mesma probabilidade de bloqueio para um mesmo tamanho de *buffer*. Considerando valores de  $N_{min}$  e  $T_1$  correspondentes, o primeiro esquema será melhor se as curvas do modelo 1 permanecerem abaixo das curvas do modelo 2. Isto é verdade pois neste caso a probabilidade do  *jitter* ser maior que um certo valor será sempre inferior para o modelo 1. Para as curvas acima os valores de  $N_{min}$  e  $T_1$  escolhidos fornecem a mesma probabilidade de bloqueio. Podemos então afirmar que, para valores pequenos de desvio padrão, tanto o modelo 1 como o modelo 2 apresentam bom desempenho como mostram as Figuras 2.11 e 2.12. Isso porém não é verdade para valores grandes de desvio padrão, onde o segundo modelo não é eficaz na eliminação do  *jitter* positivo como mostra a Figura 2.13.

Agora iremos comparar os resultados apresentados nesta seção com os resultados obtidos com a ferramenta de comunicação de áudio desenvolvida no projeto ALMADEM (pertencente ao programa PROTEM-III) e apresentados em [16]. Os testes foram feitos entre o Departamento de Ciências da Computação da UCLA e o NCE na UFRJ. Entre as duas instituições existem 16 roteadores e tráfego intenso. As máquinas utilizadas para os testes foram modelos *Sparc* rodando o sistema SunOS 4.1 da Sun Microsystems.

A Figura 2.14 compara os resultados de [16] com a distribuição do  *jitter* com desvio padrão igual a 21. As curvas de entrada no mecanismo de redução do  *jitter* possuem a mesma média e variância para o  *jitter* nos dois casos. As curvas *A* e *a* representam a distribuição do  *jitter* na saída do mecanismo de redução do  *jitter* para  $N_{min} = 5$  no modelo analítico e no teste real, respectivamente. Enquanto que, as curvas *B* e *b* mostram a distribuição do  *jitter* para  $N_{min} = 10$ . Note que as diferenças encontradas entre o modelo analítico e os testes realizados não são significativas, o que valida os modelos discutidos neste capítulo.

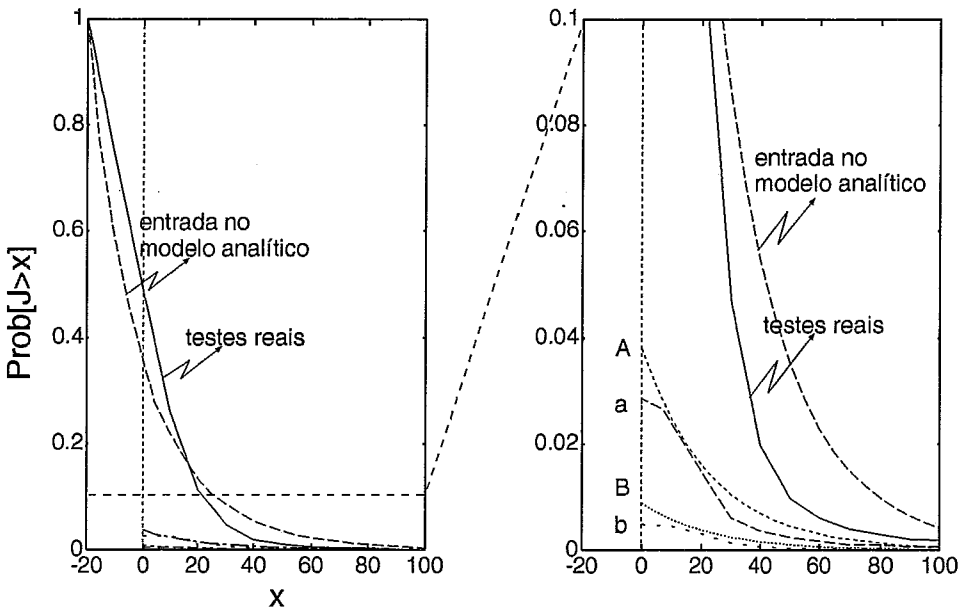


Figura 2.14: Dados reais  $\times$  Distribuição de *jitter* com  $\sigma = 21$ .

## 2.5 Conclusões

Neste capítulo, comentamos sobre alguns mecanismos para controle do *jitter* na rede e no destino. Os mecanismos para controle do *jitter* na rede baseiam-se em introduzir retardos nos pacotes dentro da rede. O que muda de um mecanismo para outro é o tempo que os pacotes são retardados em cada nó da rede. Além disso, alguns mecanismos requerem testes e cálculos adicionais nos comutadores da rede quando comparados ao serviço FIFO. Como podemos ver, o preço a ser pago para a rede limitar o *jitter* de um tráfego é o aumento do retardo fim-a-fim e do nível de complexidade no processamento dos pacotes nos nós da rede. Por outro lado, os controles do *jitter* no destino propostos são de fácil implementação, e não requerem processamento por parte dos nós da rede.

Dois modelos para o estudo do controle do *jitter* no destino são propostos e analisados (veja [23] e [24] para outros exemplos de modelagem e análise do controle do *jitter* no destino). O primeiro modelo armazena  $N_{min}$  pacotes antes de iniciar a entrega dos pacotes recebidos, e o segundo modelo espera por um tempo *fixo*  $T_1$  após a chegada do primeiro pacote antes de iniciar o processo de

entrega. Infelizmente, a escolha de  $N_{min}(T_1)$  não é simples. Se  $N_{min}(T)$  é pequeno, o número de pacotes armazenados antes de iniciar a entrega pode ser pequeno, e a fila é esvaziada rapidamente. Se  $N_{min}(T_1)$  é muito grande o número de pacotes armazenados pode ser grande e uma grande quantidade de *buffers* é necessário. Além disso, o aumento do valor de  $N_{min}(T_1)$  implica no aumento do valor do retardo na entrega dos pacotes.

Os resultados apresentados neste capítulo mostram que é possível reduzir o *jitter* com o armazenamento de poucos pacotes no destino. Os dois modelos têm bom desempenho quando o desvio padrão do *jitter* não é muito grande. Para um tráfego com grande desvio padrão no *jitter*, o primeiro modelo tem bom desempenho mesmo para valores pequenos de  $N_{min}$ , enquanto os resultados do segundo modelo indicam que ele tem um desempenho inferior ao do primeiro esquema.

É interessante observar que os modelos aqui apresentados podem ser usados para analisar outras propostas de mecanismos de controle para redes de comunicação. Por exemplo, em [25] Sen *et al* propõem que servidores *proxy* sejam utilizados para garantir uma melhor QoS nas transmissões de áudio/vídeo. Um servidor *proxy* é um equipamento usado para armazenar as páginas mais acessadas da Internet em um determinado ambiente. Periodicamente, o *proxy* atualiza as páginas com os provedores de origem e quando um usuário solicita acesso a uma dessas páginas ele a entrega de imediato, sem precisar requisitá-la ao provedor. A vantagem do *proxy* é sem dúvida poder oferecer aos usuários maior rapidez de acesso as páginas mais populares. Além disso, a atualização dessas páginas no *proxy* é feita durante períodos de pouca utilização da rede, evitando assim congestionamento.

A proposta de [25] é que o *proxy* passe a armazenar os primeiros pacotes das transmissões de áudio/vídeo mais solicitadas no ambiente sendo monitorado. Quando um usuário solicita uma dessas transmissões, o *proxy* envia uma solicitação ao provedor especificando o ponto em que a transmissão deve ser inici-

ada. Com a chegada dos primeiros pacotes, o *proxy* inicia a entrega. Em [25], são discutidas a implementação dessa proposta e a escolha de alguns parâmetros tais como quantidade de pacotes a ser armazenada no *proxy* e a quantidade de pacotes a ser recebida antes de iniciar a entrega dos pacotes ao usuário. O estudo de medidas de QoS, como por exemplo *jitter*, não foi feito em [25].

Note que podemos usar o primeiro modelo apresentado neste capítulo para estudar este problema. Só que agora estamos interessados na análise transiente do modelo, pois queremos conhecer o comportamento de uma determinada transmissão de áudio/vídeo. Para isso podemos interpretar o período ativo como o tempo de transmissão e o período de silêncio como o final da transmissão. Portanto, a média de duração de um período ativo deve ser igual à média de duração do filme. Precisamos apenas incluir uma alteração no modelo: o início de um período ativo é agora caracterizado pela chegada de  $k$  pacotes ao invés de um único pacote. Estes  $k$  pacotes representam os pacotes armazenados pelo *proxy* mais o primeiro pacote enviado pelo provedor. A entrega dos pacotes só é iniciada quando o *proxy* tem  $N_{min}$  pacotes, onde  $N_{min} \geq k + 1$ .

Os modelos apresentados neste trabalho também podem ser usados para estudar a QoS oferecida pela rede diante de problemas, como por exemplo, retransmissão de pacotes. Em [26], Jean-Marie *et al* estudam como a reordenação de pacotes feita no nó de destino afeta o desempenho de aplicações que exigem o recebimento ordenado dos pacotes. Pacotes chegam fora de ordem devido a erros na transmissão. Supõe-se no modelo que no máximo é pedida a retransmissão de um pacote em um determinado tempo. A modelagem é feita com duas filas. Uma fila armazena os pacotes ordenados e prontos para serem entregues à aplicação. A outra fila armazena os pacotes que chegaram após o pacote que está faltando. Quando este pacote chega, todos os pacotes que estão na segunda fila são transferidos para a primeira fila. Podemos usar o primeiro modelo apresentado neste capítulo para estudar este problema. Por exemplo, a transferência de pacotes da segunda fila para a primeira fila (indicando a chegada do pacote que estava

faltando) é representada pela chegada de um conjunto de pacotes com intervalos de chegada entre eles igual a zero (o *jitter* na entrada do *buffer* é  $-T$ ). Os outros estados do modelo representam a chegada de pacotes quando nenhuma perda é detectada.



# Capítulo 3

## Aproximação para o Algoritmo GTH

### 3.1 Introdução

Uma das medidas mais importantes a ser obtida na análise de um modelo é o conjunto de probabilidades estacionárias dos estados. Esta medida é calculada a partir da matriz de taxas (ou probabilidades) de transição dos estados do modelo e pode ser interpretada como a fração de tempo que o sistema fica em cada estado. A partir da solução do sistema em estado estacionário, podemos calcular várias medidas de interesse para estudar o comportamento do modelo. Por exemplo, considere novamente os modelos de *jitter* estudados no Capítulo 2. A probabilidade da fila de pacotes esvaziar após o início da entrega de pacotes está relacionada à probabilidade do modelo ter *jitter* positivo e portanto, ocorrer uma diminuição na QoS oferecida pela rede. Outras medidas em estado estacionário que podem ser calculadas para os modelos de *jitter* são: número médio de pacotes na fila e taxa de perda de pacotes (para melhor dimensionar o tamanho do *buffer* no modelo real), tempo médio de permanência de um pacote na fila (para calcular o retardo fim-a-fim na entrega dos pacotes), etc.

Os métodos de solução usados para obter as probabilidades em estado estacionário são divididos em diretos e iterativos [28]. Um método é dito direto quando fornece as probabilidades exatas dos estados após um número finito de passos, e um método é dito iterativo quando fornece uma solução que converge para a solução exata a cada iteração do algoritmo. Em geral, métodos diretos são usados quando o número de estados do sistema sendo modelado não é muito grande (da ordem de algumas centenas de estados) e quando a matriz de transição de estados do modelo não é esparsa (poucas transições com valores zero).

Um dos métodos diretos mais importantes na solução de cadeia de Markov é o algoritmo proposto por Grassmann, Taksar e Heyman [29], e conhecido por método GTH. Na realidade, o método GTH é uma variação do método de eliminação de Gauss [28][30] com algumas importantes características a mais. Por exemplo, o algoritmo só possui operações de multiplicação e adição, nenhuma subtração é realizada, evitando assim perda de precisão nos valores calculados. Além disso, existe uma interpretação probabilística para cada passo executado pelo algoritmo GTH.

Um problema do algoritmo GTH é o custo computacional elevado para matrizes que não possuem estruturas especiais. Isto se deve ao fato do GTH introduzir modificações na matriz de transição de estados. A cada passo executado, algumas transições de estado são removidas e novas transições de estado podem ser introduzidas na matriz resultante. Mesmo que a matriz original seja esparsa, o algoritmo pode preencher os espaços vazios da matriz com novas transições, fazendo com que a matriz final não seja esparsa. Por isso, o algoritmo GTH não é usado para resolver modelos muito grandes (da ordem de alguns milhares de estados), a não ser que a matriz de transição de estados do modelo tenha alguma estrutura particular, como por exemplo, matriz de banda.

Neste capítulo propomos uma aproximação para o método GTH de forma que ele possa ser usado na solução de matrizes com milhares de estados impossível de serem resolvidas pelo método GTH. Este algoritmo faz uso de conceitos e

definições de dois outros algoritmos de aproximação apresentados em [31] e [32]. Visando estudar a eficácia do algoritmo proposto, iremos comparar as soluções de alguns modelos markovianos obtidas com o algoritmo GTH e com o novo algoritmo.

Em [17], de Souza e Silva *et al* mostram uma solução para modelos não markovianos que possuem, no máximo, uma única transição não exponencial habilitada em um determinado momento. Este algoritmo usa métodos de solução para modelos markovianos, por exemplo o GTH, como parte da solução. Por isso, também usaremos os algoritmos GTH e de aproximação para resolver alguns modelos não markovianos. Para facilitar a compreensão das definições usadas neste capítulo, faremos uma rápida revisão de métodos de solução de modelos markovianos, em especial do algoritmo GTH. Além disso, discutiremos o algoritmo proposto em [17] para a solução de modelos não markovianos e os algoritmos de aproximação de [31] e [32].

A organização deste capítulo é descrita a seguir. Em 3.2 são apresentados alguns conceitos relacionados a métodos de solução utilizados neste trabalho, além de detalhar o algoritmo GTH e o método de solução para modelos não markovianos introduzido em [17]. Em 3.3 estudamos dois métodos de aproximação para modelos markovianos encontrados na literatura e que vão servir de base para o método proposto. Em 3.4 propomos uma alteração no método GTH com o objetivo de diminuir o custo computacional do algoritmo e calculamos o erro introduzido com esta modificação. Em 3.5 mostramos a solução de alguns modelos markovianos e não markovianos usando o algoritmo GTH e o algoritmo de aproximação aqui proposto. Veremos então que foi possível limitar o erro no segundo algoritmo apenas para o resultado obtido de modelos markovianos. A seção 3.6 conclui o capítulo.

## 3.2 Métodos de Solução

Nesta seção fazemos revisão de alguns conceitos relacionados a métodos de solução que serão úteis nas seções seguintes. Inicialmente, descrevemos brevemente a solução de modelos markovianos e o método de uniformização. Em seguida, detalhamos o algoritmo GTH e o algoritmo proposto em [17] para solução de modelos não markovianos.

### 3.2.1 Solução de Modelos Markovianos

Considere uma cadeia de Markov  $\mathcal{X}$  de tempo contínuo e espaço de estados finito com  $M$  estados. Seja  $\mathbf{Q}$  a matriz de taxas de transição de  $\mathcal{X}$ . Podemos transformar  $\mathcal{X}$  em uma cadeia  $\mathcal{Z}$  de tempo discreto e com matriz  $\mathbf{P}$  de probabilidades de transição usando o método de uniformização [28, 33, 34]. Para isso, basta fazermos

$$\mathbf{P} = \mathbf{I} + \frac{\mathbf{Q}}{\Lambda},$$

onde  $\Lambda$  é a taxa de uniformização e é escolhida tal que seja maior ou igual à maior taxa de saída dentre todos os estados de  $\mathcal{X}$ . O vetor  $\pi$  de probabilidades estacionárias para o modelo markoviano pode ser obtido solucionando

$$\pi \mathbf{Q} = 0 \quad \text{ou} \quad \pi = \pi \mathbf{P},$$

onde  $\pi = \langle \pi_1, \dots, \pi_M \rangle$ .

Para calcular o vetor  $\pi$  podemos usar, por exemplo, o método GTH para solução de cadeias de Markov de tempo discreto. Na realidade, o GTH é uma variação do método direto de Gauss [28]. O GTH será discutido em detalhes a seguir pois o objetivo deste trabalho é apresentar um método de aproximação

para este algoritmo. Podemos também usar métodos iterativos para calcular  $\pi$  como Jacobi, Power, SOR (Successive Over-Relaxation), etc. Veja [28] para uma discussão mais detalhada sobre estes e outros métodos de solução para modelos markovianos.

## Método de Solução GTH

O algoritmo GTH é dividido em duas partes, cada uma com  $M - 1$  passos, onde  $M$  corresponde ao número de estados do modelo. Na primeira parte, a cada passo do algoritmo um estado é *eliminado* e as probabilidades de transição dos demais estados são recalculadas de forma que as probabilidades estacionárias dos estados da nova cadeia de Markov correspondam às probabilidades condicionais dos mesmos estados na cadeia original para o subconjunto restante. Na segunda parte, é feita a operação inversa, onde a cada passo do algoritmo um estado é *adicionado*, e é calculada a probabilidade condicional deste estado. As probabilidades condicionais (normalizadas em 1) encontradas na última parte do algoritmo correspondem às probabilidades estacionárias da cadeia de Markov original.

Em maiores detalhes, seja  $\mathcal{Z} = \{Z_k : k = 0, 1, \dots\}$  uma cadeia de Markov homogênea de tempo discreto com espaço de estados  $\mathcal{S} = \{s_i : i = 1, \dots, M\}$  e matriz de probabilidades de transição  $\mathbf{P}$ . Considere que os estados estão divididos em dois grupos como mostrado na Figura 3.1. O primeiro grupo ( $G_1$ ) é formado pelos estados  $s_1, \dots, s_{M-1}$ , e o segundo grupo ( $G_2$ ) é formado apenas pelo estado  $s_M$ . Note que a soma das probabilidades estacionárias dos estados  $s_1, \dots, s_{M-1}$  corresponde à fração do tempo que o sistema passa em  $G_1$ , enquanto que a probabilidade estacionária de  $s_M$  corresponde ao tempo que o sistema passa em  $G_2$  (como a cadeia é de tempo discreto supomos que cada visita a um estado corresponde a uma unidade de tempo).

Seja  $p_{i,j}^{(M)}$  a probabilidade de transição do estado  $s_i$  para o estado  $s_j$  quando nenhum estado foi ainda eliminado e portanto, a cadeia possui os  $M$  estados

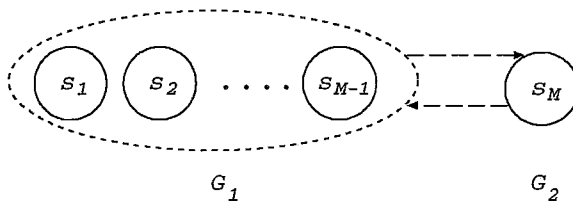


Figura 3.1: Grupos de estados.

originais. Portanto, para  $i = 1, \dots, M - 1$ ,  $p_{i,M}^{(M)}$  é a probabilidade do sistema sair de  $G_1$  para  $G_2$  onde o último estado visitado é  $s_i$ , e  $p_{M,i}^{(M)}$  é a probabilidade do sistema visitar inicialmente o estado  $s_i$  após uma visita a  $G_2$ .

Suponha agora que o tempo de permanência do sistema em  $G_2$  seja nulo. Como a probabilidade de saída de  $G_2$  é  $1 - p_{M,M}^{(M)}$ , podemos afirmar que a probabilidade do sistema reentrar em  $G_1$  através do estado  $s_j$  dado que saiu de  $G_1$  pelo estado  $s_i$  é

$$p_{i,M}^{(M)} \times \frac{p_{M,j}^{(M)}}{1 - p_{M,M}^{(M)}}.$$

Logo, podemos eliminar o estado  $s_M$  e definir uma nova cadeia de Markov com  $M - 1$  estados, onde as probabilidades de transição são definidas por

$$p_{i,j}^{(M-1)} = p_{i,j}^{(M)} + p_{i,M}^{(M)} \times \frac{p_{M,j}^{(M)}}{1 - p_{M,M}^{(M)}}, \quad 1 \leq i, j, \leq M - 1.$$

É importante observar que  $1 - p_{M,M}^{(M)}$  da expressão acima pode ser substituído por  $\sum_{i=1}^{M-1} p_{M,i}^{(M)}$ , e assim podemos evitar operações de subtração no algoritmo GTH. A Figura 3.2 mostra como a probabilidade  $p_{i,j}^{(M-1)}$  da nova matriz é calculada. As probabilidades estacionárias da cadeia de Markov obtida quando eliminamos o estado  $s_M$  correspondem às probabilidades estacionárias dos  $M - 1$  primeiros estados da cadeia de Markov original condicionando que o sistema se encontra em  $G_1$ .

Podemos então repetir as operações discutidas acima para eliminarmos o es-

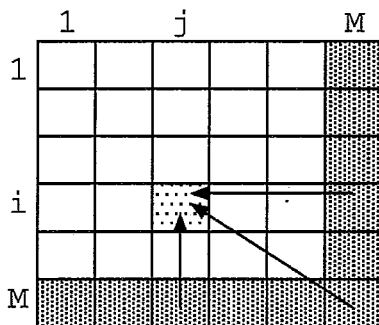


Figura 3.2: Cálculo do elemento  $p_{ij}^{(M-1)}$ .

tado  $s_{M-1}$  e obtermos uma cadeia com  $M - 2$  estados. E assim sucessivamente. Após  $M - 1$  passos, a cadeia de Markov tem apenas um estado e portanto, a probabilidade estacionária desse estado é 1, ou seja,  $\pi_1 = 1$ .

Note que no início do passo  $n$  do algoritmo, o sistema tem  $M - n + 1$  estados e a probabilidade de transição do estado  $i$  para o estado  $j$  é representada por  $p_{ij}^{(M-n+1)}$ , onde  $1 \leq i, j \leq M - n + 1$ . Seja  $\mathbf{A}_{M-n+1}$  a matriz de probabilidades de transição no  $n$ -ésimo passo do algoritmo, onde  $\mathbf{A}_M = \mathbf{P}$ . Vimos na solução de modelos markovianos (início desta seção) que  $\langle \pi_1, \dots, \pi_{M-n+1} \rangle = \langle \pi_1, \dots, \pi_{M-n+1} \rangle \mathbf{A}_{M-n+1}$ . Além disso, sabemos que  $\pi_1 = 1$ . Portanto, para a cadeia com dois estados podemos calcular

$$\pi_2 = \frac{p_{12}^{(2)}}{1 - p_{22}^{(2)}} \pi_1.$$

Como conhecemos  $\pi_1$  e  $\pi_2$  (valores não normalizados), podemos agora calcular

$$\pi_3 = \frac{p_{13}^{(3)}}{1 - p_{33}^{(3)}} \pi_1 + \frac{p_{23}^{(3)}}{1 - p_{33}^{(3)}} \pi_2.$$

E assim sucessivamente, onde a probabilidade do  $i$ -ésimo estado é dada por

$$\pi_i = \sum_{j=1}^{i-1} \frac{p_{ji}^{(i)}}{1 - p_{ii}^{(i)}} \pi_j.$$

Naturalmente, as probabilidades acima calculadas ainda precisam ser normalizadas.

A primeira parte do algoritmo GTH tem um custo da  $O(n^3)$ , enquanto que a segunda parte do algoritmo tem custo da  $O(n^2)$ . Portanto, o custo do algoritmo é da  $O(n^3)$ .

Em [35], Meo *et al* propõem uma modificação no algoritmo de eliminação de Gauss similar ao algoritmo GTH. O método proposto mostra-se mais eficaz que o GTH para uma classe de modelos markovianos bastante utilizada em redes de comunicação de alta velocidade. O problema com este algoritmo, além do uso restrito a uma classe de modelos, é que ele utiliza operações de subtração no cálculo da solução, o que pode levar a problemas de perda de precisão nos valores calculados.

### 3.2.2 Solução de Modelos Não Markovianos

Quando o tempo entre eventos de um modelo não tem distribuição exponencial, o modelo resultante não é markoviano. Por exemplo, os modelos de *jitter* vistos no Capítulo 2 possuem transições determinísticas. A seguir, vamos detalhar o algoritmo introduzido em [17] e implementado na ferramenta TANGRAM-II [20, 21] para a solução de uma classe de modelos não markovianos onde no máximo 1 evento com duração não exponencial (por exemplo, determinístico) pode estar habilitado num instante qualquer.

Seja  $\varphi_j$  um evento determinístico do modelo, i.e., o intervalo de tempo entre a ativação do evento e o disparo é determinístico. Este evento está habilitado durante o mini-intervalo  $\Delta_j$  e poderá executar após transcorrido um intervalo de tempo  $T_j$ . Entretanto,  $\varphi_j$  pode ser desabilitado pela ocorrência de outros eventos. No caso em que o evento  $\varphi_j$  não é desabilitado no intervalo  $\Delta_j$ , o comprimento  $\delta_j$  deste intervalo é igual a  $T_j$ . Caso contrário,  $\delta_j < T_j$ .  $\Delta_j$  corresponde então ao



intervalo entre pontos embutidos onde  $\varphi_j$  está habilitado. Por conveniência de notação, chamamos de  $\Delta_0$  o intervalo onde nenhum evento determinístico está habilitado.

Entre dois pontos embutidos o comportamento dos sistemas descritos pode ser modelado por uma cadeia de Markov. Isto porque, todos os eventos entre pontos embutidos são exponenciais. A matriz de probabilidades de transição da cadeia embutida é então obtida através da análise transiente da cadeia de Markov associada aos intervalos  $\Delta_j$ . Quando o intervalo não tem evento determinístico habilitado (intervalo  $\Delta_0$ ), as probabilidades de transição de estado são facilmente calculadas construindo-se uma cadeia de Markov associada a  $\Delta_0$  e mais um conjunto de estados, um para cada evento determinístico que pode ser habilitado ao término do intervalo.

A partir da obtenção da matriz de probabilidades de transição da cadeia embutida (chamada em [17] de matriz  $\mathcal{H}$ ), o vetor de probabilidades estacionárias pode então ser calculado usando, por exemplo, o algoritmo GTH. Observe que o número de estados da matriz  $\mathcal{H}$  pode ser menor que o número de estados do modelo, isto se deve ao fato que nem toda transição de estado da cadeia original representa um ponto embutido. É também importante notar que o vetor de probabilidades encontrado a partir da matriz  $\mathcal{H}$  representa as probabilidades estacionárias dos pontos embutidos e não dos estados do modelo.

Sejam  $\beta$  e  $\pi$ , respectivamente, o vetor das probabilidades estacionárias da matriz  $\mathcal{H}$  e o vetor das probabilidades estacionárias do modelo original, e seja  $S^{(j)}$  o sub-conjunto dos estados da cadeia de Markov onde o evento  $\varphi_j$  está habilitado. Podemos, então calcular [17]:

$$\pi_i = \frac{\sum_{j=0}^E \sum_{s \in S^{(j)}} E[U_{s,i}^{(j)}] \beta_s}{\sum_{j=0}^E \sum_{s \in S^{(j)}} E[\delta_s^{(j)}] \beta_s}, \quad (3.1)$$

onde  $E$  é o número de eventos determinísticos,  $U_{s,i}^{(j)}$  corresponde ao tempo gasto pelo sistema no estado  $i$  durante o intervalo  $\Delta_j$  que se inicia no estado  $s$ , e  $\delta_s^j$  corresponde ao tamanho do intervalo  $\Delta_j$  cujo estado inicial é  $s$ . Veja [17] para mais detalhes do algoritmo de solução de cadeias embutidas, por exemplo, como calcular  $U_{s,i}^{(j)}$  e  $\delta_s^j$ .

### 3.3 Métodos de Aproximação para a Solução de Modelos Markovianos

Nesta seção discutiremos dois métodos de aproximação encontrados na literatura para a solução de modelos markovianos e que servirão de base para o novo algoritmo de aproximação a ser apresentado na próxima seção.

A partir de uma cadeia de Markov  $\mathcal{X}$  com matriz  $\mathbf{P}$  de probabilidades de transição, os dois algoritmos geram uma nova cadeia de Markov  $\mathcal{X}'$  com matriz de probabilidades de transição  $\mathbf{P}'$ , onde  $\mathbf{P}'$  possui uma estrutura especial.

Os algoritmos dividem os estados de  $\mathcal{X}'$  em dois grupos. Suponha que estes grupos são  $G_1$  e  $G_2$ . As taxas de transição entre os estados de  $G_1$  e as taxas de transição dos estados de  $G_1$  para  $G_2$  são conhecidas. Entretanto, nada sabemos sobre as taxas de transição dos estados de  $G_2$ . As taxas de  $G_2$  são definidas de forma a garantir que a solução de  $\mathcal{X}'$  para os estados de  $G_1$  é um limite inferior para os estados correspondentes em  $\mathcal{X}$ .

O primeiro método a ser estudado utiliza apenas um estado para representar  $G_2$ , enquanto o segundo método faz uso de duplicação e agregação dos estados de  $G_1$  para gerar um conjunto de estados para  $G_2$ . A seguir veremos com mais detalhes estes dois métodos.

### 3.3.1 Cadeias de Markov $\varepsilon$ -quasi-lumpable

Um dos métodos usados para a solução de cadeias de Markov com grande número de estados é a aglutinação (*lumping*). Com este método é possível reduzir o espaço de estados de um modelo markoviano preservando informações suficientes de forma a obter as medidas de interesse desejadas. O problema é que a aglutinação não pode ser aplicada a todos os modelos markovianos. Quando este método é aplicável a um modelo, dizemos que a cadeia de Markov correspondente é aglutinável (*lumpable*). Em [31], Franceschinis e Muntz propõem uma técnica que permite ampliar o conjunto de modelos que podem fazer uso do método de aglutinação, fornecendo valores aproximados para as medidas de interesse desejadas. Este método é aplicável a uma cadeia de Markov não aglutinável se a cadeia puder ser transformada em aglutinável com uma *pequena* perturbação nas taxas de transição.

Dada uma partição  $\mathcal{S} = \{S_1, \dots, S_M\}$  de uma cadeia de Markov  $\mathcal{X}$  de tempo discreto com matriz de probabilidades de transição  $\mathbf{P}$ , dizemos que  $\mathcal{X}$  é aglutinável para a partição  $\mathcal{S}$  se a probabilidade de transição do estado  $s \in S_i$ ,  $i = 1, \dots, M$ , para um determinado estado da partição  $S_j$ ,  $j = 1, \dots, M$  e  $j \neq i$ , sempre tem um mesmo valor, qualquer que seja o estado  $s$ .

Uma cadeia de Markov é dita  $\varepsilon$ -quasi-lumpable se a sua matriz de probabilidade  $\mathbf{P}$  pode ser escrita como  $\mathbf{P} = \mathbf{P}^- + \mathbf{P}^\varepsilon$ , onde  $\mathbf{P}^- + \text{diag}(\mathbf{P}^\varepsilon e^T)$  é aglutinável,  $\mathbf{P}^\varepsilon$  é uma matriz quase nula e  $e^T$  é a matriz coluna com valores 1. A solução de  $\mathbf{P}^- + \text{diag}(\mathbf{P}^\varepsilon e^T)$  fornece uma boa aproximação para o modelo se os valores  $\varepsilon$  forem bem pequenos. O problema é a impossibilidade de se obter limites para o erro da solução quando esta aproximação é usada.

A idéia apresentada em [31] consiste em transformar a matriz de transição  $\varepsilon$ -quasi-lumpable  $\mathbf{P}$  com  $n$  estados em uma matriz de transição aglutinável  $\mathbf{P}_s$  com  $n+1$  estados. Para isso, todas as transições  $\varepsilon$  da matriz  $\mathbf{P}^\varepsilon$  são direcionadas para um novo estado ao invés de serem colocadas na diagonal. A Figura 3.3 mostra

a matriz  $\mathbf{P}_s$  do novo modelo, onde  $y^T = \mathbf{P}^\varepsilon e^T$  e  $x e^T = 1$ . O objetivo é definir a matriz  $\mathbf{P}_s$  de forma que, as probabilidades condicionais dos  $n$  primeiros estados de  $\mathbf{P}_s$  sejam iguais às probabilidades estacionárias dos estados de  $\mathbf{P}$ . Note que temos um problema: nada sabemos sobre o vetor  $x$ . É possível achar  $x$  de forma que as probabilidades condicionais dos estados em  $\mathbf{P}^-$  sejam iguais às probabilidades de  $\mathbf{P}$ , mas isso implica em resolver a matriz original.

$P^-$	$y^T$
$x$	$0$

Figura 3.3: Matriz  $\mathbf{P}_s$ .

Podemos solucionar aproximadamente a matriz  $\mathbf{P}_s$  usando os resultados de Courtois e Semal[36, 37]. Para isto, basta fazermos separadamente cada elemento de  $x$  igual a 1 e encontrar a solução do modelo correspondente. O maior e o menor valor encontrados para uma determinada medida de interesse fornecem, respectivamente, um limite superior e um limite inferior para a medida de interesse desejada. Neste caso, se  $\mathbf{P}_s$  tem  $n + 1$  estados, é preciso resolver o modelo  $n$  vezes. Entretanto, pode-se reduzir o custo do algoritmo aplicando esta regra apenas aos estados cuja coluna correspondente em  $\mathbf{P}^\varepsilon$  tem pelo menos um valor diferente de zero [31].

Os exemplos apresentados em [31] mostram a vantagem do uso do método para modelos com cadeia de Markov  $\varepsilon$ -quasi-lumpable, embora exista a necessidade de solucionar a matriz  $\mathbf{P}_s$  mais de uma vez. Como a matriz  $\mathbf{P}_s$  é aglutinável, o custo para solucionar  $\mathbf{P}_s$ , para qualquer vetor  $x$ , é pequeno em relação ao custo da solução da matriz original  $\mathbf{P}$ . Isto significa que o custo em resolver a matriz  $\mathbf{P}_s$  várias vezes ainda pode ser menor que o custo da solução da matriz  $\mathbf{P}$ .

### 3.3.2 Solução para um Sub-conjunto de Estados de um Modelo Markoviano

Em [32], de Souza e Silva e Ochoa propõem um método de aproximação para calcular as medidas de interesse usando apenas um sub-conjunto de estados da cadeia de Markov correspondente. A idéia é que apenas os estados considerados *importantes* façam parte do grupo de estados selecionados. A importância de um estado é definida pela sua contribuição no cálculo da medida de interesse desejada.

Seja  $\mathcal{X}$  uma cadeia de Markov homogênea de tempo contínuo com espaço de estados  $S = \{s_i : i = 1, \dots, M\}$  e matriz de taxas de transição  $\mathbf{Q}$ . Para  $\{s_i, s_j\} \in S$ , dizemos que o estado  $s_j$  está a  $k$  passos do estado  $s_i$ , se o menor caminho do estado  $s_j$  até o estado  $s_i$  é composto por  $k$  transições.

Seja  $\mathcal{S} = \{S_0, \dots, S_{k-1}, S_k, S_f\}$  uma partição de  $\mathcal{X}$  onde  $S_i$  corresponde ao conjunto de estados que estão a  $i$  passos da partição  $S_0$ ,  $i = 1, \dots, k$ , e  $S_f$  corresponde ao resto dos estados da cadeia de Markov. Precisamos fazer duas observações, uma em relação a  $S_0$  e outra em relação ao parâmetro  $k$ . Em primeiro lugar,  $S_0$  é uma partição com um único estado, onde o estado escolhido é considerado *importante* no cálculo da medida de interesse desejada. Em segundo lugar, o valor de  $k$  é calculado antes de se usar o algoritmo de aproximação ou é calculado iterativamente, durante a execução do algoritmo. Veja [32] para mais detalhes.

Podemos então definir 3 grupos de estados no modelo:  $G_0 = \{S_0\}$ ,  $G_1 = \{S_1, \dots, S_k\}$  e  $G_2 = \{S_f\}$ . Usando a idéia apresentada em [38] podemos construir uma cadeia de Markov  $\mathcal{X}'$  com 4 grupos de estados ( $G'_0, G'_{1a}, G'_{1b}$  e  $G'_2$ ) a partir de  $\mathcal{X}$  da seguinte forma:  $G'_0 = G_0$ ,  $G'_{1a} = G'_{1b} = G_1$  e  $G'_2 = G_2$ . Isto significa que  $G'_{1a}$  e  $G'_{1b}$  são cópias de  $G_1$ , assim como  $G'_0$  é uma cópia de  $G_0$  e  $G'_2$  é uma cópia de  $G_2$ . A Figura 3.4 mostra as transições entre os grupos do novo processo  $\mathcal{X}'$ . As transições entre estados do mesmo grupo são idênticas aos do Processo  $\mathcal{X}$ . Note que não existem transições de  $G'_0$  e  $G'_{1a}$  para  $G'_{1b}$  e nem de  $G'_2$  para  $G'_0$

e  $G'_{1a}$ .

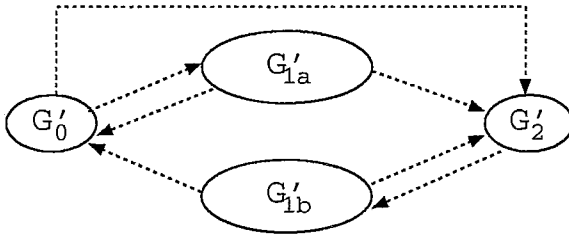


Figura 3.4: Novo Processo  $\mathcal{X}'$ .

Para uma melhor compreensão do processo  $\mathcal{X}'$  mostrado na Figura 3.4, vamos exemplificar usando o modelo de filas discutido em [40]. Este modelo foi escolhido devido à sua simplicidade e por permitir discutir na seção 3.4.2 alguns problemas relacionados à implementação do novo algoritmo.

Considere um modelo markoviano com duas filas, onde cada fila tem 3 servidores e capacidade de armazenamento de 4 pacotes. A taxa de chegada, a taxa de serviço e o número de pacotes da fila  $i$  é, respectivamente,  $\lambda_i$ ,  $\mu_i$ , e  $n_i$ ,  $i = 1, 2$ . Se um pacote chega na fila 1 e encontra a fila cheia, ele é enviado para a fila 2. Pacotes que chegam na fila 2 e encontram a fila cheia, são descartados. A Figura 3.5 mostra a cadeia de Markov para este exemplo, onde o estado do modelo é representado pelo número de pacotes  $n_1$  e  $n_2$  das filas 1 e 2, respectivamente.

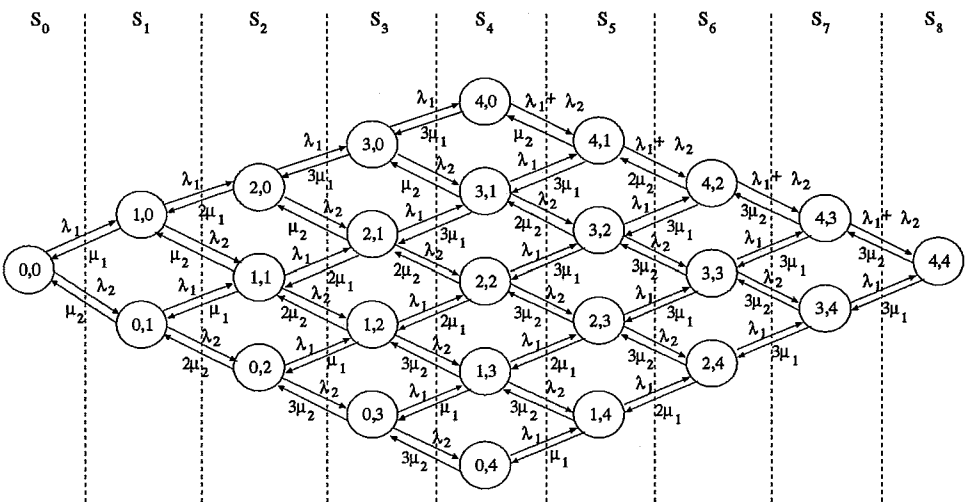


Figura 3.5: Modelo com 2 filas de pacotes.

Seja  $S_0 = \{(0, 0)\}$ , isto é,  $S_0$  representa o sistema com filas vazias. Quando discutimos a partição do processo  $\mathcal{X}$ , vimos que os estados da partição  $S_i$  estão a  $i$  passos do estado de  $S_0$ . Logo, temos  $S_1 = \{(1, 0), (0, 1)\}$ ,  $S_2 = \{(2, 0), (1, 1), (0, 2)\}$ , ...,  $S_8 = \{(4, 4)\}$ . Suponha que o parâmetro  $k$  do processo  $\mathcal{X}$  é igual a 4. Isto significa que  $G_0 = \{S_0\}$ ,  $G_1 = \{S_1, \dots, S_4\}$  e  $G_2 = \{S_5, \dots, S_8\}$ . A partir desses grupos de estados do processo  $\mathcal{X}$  podemos gerar o processo  $\mathcal{X}'$  como mostra a Figura 3.4. Vejamos então o que representa os grupos de estados  $G'_0$ ,  $G'_{1a}$ ,  $G'_{1b}$  e  $G'_2$  para este exemplo.

Suponha que inicialmente as filas estão vazias, isto é, não existem pacotes armazenados. Portanto, o sistema se encontra em  $G'_0$ . Quando um pacote chega, ele é armazenado e o estado do sistema passa a ser  $(0, 1)$  ou  $(1, 0)$ . Consequentemente, o sistema entra em  $G'_{1a}$ . O sistema fica em um dos estados de  $G'_0$  ou de  $G'_{1a}$  enquanto o número total de pacotes nas duas filas for menor ou igual a 4. Quando existem 4 pacotes armazenados no sistema e ocorre a chegada e o armazenamento de um novo pacote em uma das filas, o sistema entra em  $G'_2$ , ficando neste grupo de estados enquanto o número total de pacotes no sistema for maior que 4. Quando o número total de pacotes volta a ser menor ou igual a 4, o sistema passa a ser representado por um dos estados de  $G'_{1b}$ . O sistema fica em  $G'_{1b}$  ou em  $G'_2$  até que as filas esvaziam novamente. Neste caso, o sistema volta a  $G'_0$ .

Note que os estados de  $G'_{1a}$  e  $G'_{1b}$  do processo  $\mathcal{X}'$  e os estados de  $G_1$  do processo  $\mathcal{X}$  representam a situação em que o sistema possui de 0 a 4 pacotes nas filas. Sendo que o sistema está em  $G'_{1a}$  quando o número total de pacotes nunca for maior que 4 desde que as fila esvaziaram na última vez, e o sistema está em  $G'_{1b}$  nos outros casos. Note que a solução de  $\mathcal{X}$  é igual à solução de  $\mathcal{X}'$  [32], ou seja,  $\pi_{G_0} = \pi_{G'_0}$ ,  $\pi_{G_1} = \pi_{G'_{1a}} + \pi_{G'_{1b}}$  e  $\pi_{G_2} = \pi_{G'_2}$ , onde  $\pi_G$  corresponde ao vetor de probabilidades estacionárias dos estados do grupo  $G$ .

Suponha que os estados de cada sub-conjunto  $S_i$  são agregados em um único estado  $f_i$  para  $i = 1, \dots, k$ , e que os estados do subconjunto  $S_f$  são agregados no

estado  $f_{k+1}$ . Como os estados de  $G'_{1b}$  são cópias dos estados de  $G'_{1a}$ , dizemos que  $f_i$  é um *clone* da partição  $S_i$ ,  $i = 1, \dots, k$ . O estado  $f_{k+1}$  é apenas a agregação dos estados de  $S_f$ . Vamos chamar este novo processo de  $\mathcal{X}''$ . Como fizemos apenas a agregação exata de estados no processo  $\mathcal{X}'$ , a solução de  $\mathcal{X}''$  é igual à solução de  $\mathcal{X}$ . O problema é que para fazermos a agregação desses estados precisamos solucionar  $\mathcal{X}'$ .

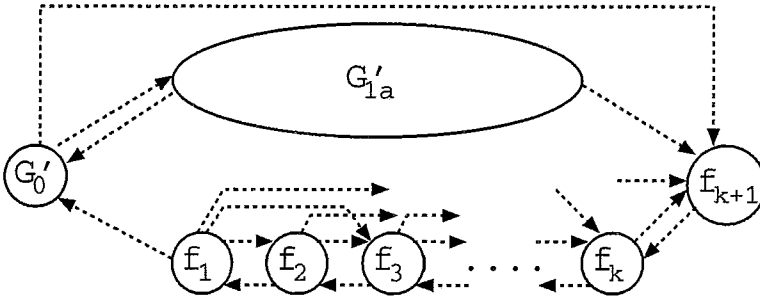


Figura 3.6: Novo processo  $\mathcal{X}''$ .

Em [32], prova-se que é possível encontrar um limite inferior para a solução dos estados de  $G'_0$  e  $G'_{1a}$  definindo as taxas de transição dos estados *clones* da seguinte forma (veja Figura 3.6):

- a transição do estado  $f_i$  para o estado  $f_j$ ,  $i < j$  e  $1 \leq i, j \leq k + 1$ , é igual à maior soma das taxas de transição de um estado em  $S_i$  para os estados em  $S_j$ ;
- a transição do estado  $f_i$  para o estado  $f_{i-1}$  (ou para o estado de  $S_0$  caso  $i = 1$ ),  $1 \leq i \leq k + 1$ , é igual à menor soma das taxas de transição de um estado em  $S_i$  para os estados em  $S_{i-1}$  (ou para o estado de  $S_0$ );
- as transições de  $G'_0$  para  $G'_{1a}$  e  $f_{k+1}$  e de  $G'_{1a}$  para  $f_{k+1}$  são idênticas aos do modelo original.

Note que a submatriz formada pelas transições entre os estados  $f_i$  ( $i = 1, \dots, k + 1$ ) é *upper Hessenberg*. Esta é uma restrição do algoritmo que permite obter um limite inferior para a solução dos estados de  $G'_0$  e de  $G'_{1a}$ .



Em [39], Lui e Muntz apresentam um método iterativo utilizando também o conceito de estados *clonados*. Neste método, uma aproximação para as medidas de interesse e uma parte dos estados da matriz de transição são geradas a cada passo do algoritmo.

### 3.4 Novo Método: uma aproximação para o algoritmo GTH

Vimos em 3.2.1 que em cada passo da primeira parte do algoritmo GTH, um estado é eliminado e novas probabilidades de transição são calculadas para os estados restantes. Por exemplo, considere uma cadeia de Markov de tempo discreto com espaço de estados  $S = \{s_i : i = 1, \dots, M\}$  e matriz de probabilidades de transição  $\mathbf{P}$ . Podemos eliminar o estado  $s_M$  e definir uma nova cadeia de Markov com  $M - 1$  estados, onde as probabilidades estacionárias dos estados da nova cadeia de Markov correpondam às probabilidades condicionais dos mesmos estados na cadeia original para o subconjunto restante. Para isso, precisamos calcular

$$p_{i,j}^{(M-1)} = p_{i,j}^{(M)} + p_{i,M}^{(M)} \times \frac{p_{M,j}^{(M)}}{1 - p_{M,M}^{(M)}}, \quad 1 \leq i, j, \leq M - 1, \quad (3.2)$$

onde  $p_{i,j}^{(M)}$  corresponde à probabilidade de transição do estado  $s_i$  para o estado  $s_j$  quando o modelo tem  $M$  estados. Portanto, para eliminar o estado  $s_M$  no algoritmo GTH precisamos multiplicar a  $M$ -ésima linha de  $\mathbf{P}$  pela  $M$ -ésima coluna de  $\mathbf{P}$ . Esta operação é repetida  $M - 1$  vezes, eliminando assim um estado em cada passo do algoritmo, até que a cadeia de Markov tenha um único estado.

Observe dois casos no cálculo de  $p_{i,j}^{(M-1)}$  da equação 3.2. No primeiro caso, já existe uma transição do estado  $s_i$  para o estado  $s_j$  quando o modelo tem  $M$  estados, ou seja,  $p_{i,j}^{(M)} > 0$ . Portanto, eliminar o estado  $s_M$  significa aumentar

a probabilidade de transição entre estes dois estados se  $p_{i,M}^{(M)} > 0$  e  $p_{M,j}^{(M)} > 0$ . No segundo caso, não existe uma probabilidade de transição do estado  $s_i$  para o estado  $s_j$ , ou seja,  $p_{i,j}^{(M)} = 0$ . Portanto, eliminar o estado  $s_M$  significa criar uma transição entre os estados  $s_i$  e  $s_j$  para a cadeia com  $M - 1$  estados se  $p_{i,M}^{(M)} > 0$  e  $p_{M,j}^{(M)} > 0$ . A partir da observação do segundo caso, podemos ver que o algoritmo GTH pode alterar a estrutura da matriz de probabilidades de transição do modelo original.

A Figura 3.7 mostra um exemplo do que pode acontecer com a estrutura de uma matriz durante a execução da primeira parte do algoritmo GTH. O exemplo mostra uma matriz esparsa de uma cadeia de Markov com nove estados (este tipo de estrutura de matriz é bastante comum em modelos de filas). Suponha que os estados do modelo estão numerados de 1 a 9. No passo 1 temos a estrutura da matriz quando nenhum estado foi ainda eliminado. No final da execução do primeiro passo do GTH, temos uma cadeia com 8 estados pois o estado 9 foi eliminado. A estrutura da nova cadeia é mostrada no passo 2 da Figura 3.7. Os elementos da matriz que estão em branco correspondem às probabilidades de transição dos estados da nova cadeia e os outros elementos correspondem às probabilidades do estado eliminado que precisam ser armazenadas para se obter as probabilidades estacionárias na segunda fase do algoritmo. O passo 3 da Figura 3.7 mostra a estrutura da cadeia com 7 estados que é utilizada no início do terceiro passo do GTH. E assim sucessivamente, até que no oitavo passo do algoritmo GTH temos uma cadeia com apenas dois estados. Note que o algoritmo vai preenchendo os espaços à medida que vai eliminando os estados e portanto, o fato da matriz ser esparsa não é aproveitado pelo GTH para diminuir o custo da solução do modelo.

Sabemos que  $1 - p_{M,M}^{(M)} = \sum_{j=1}^{M-1} p_{M,j}$ , logo temos  $0 \leq (p_{M,j}^{(M)}) / (1 - p_{M,M}^{(M)}) \leq 1$ . Suponha que  $(p_{M,j}^{(M)}) / (1 - p_{M,M}^{(M)}) = \varepsilon$  ou  $p_{i,M}^{(M)} = \varepsilon$  na equação 3.2, onde  $\varepsilon$  é um valor pequeno (em relação ao erro admitido para a solução). Podemos então escrever a equação 3.2 como

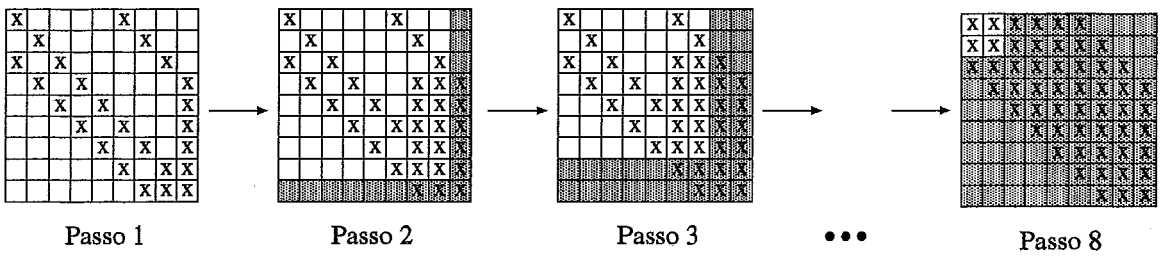


Figura 3.7: Algoritmo GTH.

$$p_{i,j}^{(M-1)} \leq p_{i,j}^{(M)} + \varepsilon. \quad (3.3)$$

A Figura 3.8 mostra o cálculo da  $j$ -ésima coluna quando  $(p_{M,j}^{(M)})/(1-p_{M,M}^{(M)}) = \varepsilon$ ,  $j = 1, \dots, M - 1$ . Note que  $p_{i,j}^{(M-1)} - p_{i,j}^{(M)} \leq \varepsilon$  para qualquer valor de  $p_{i,M}^{(M)}$ ,  $i = 1, \dots, M - 1$ . Portanto, o algoritmo adiciona ao elemento  $(i, j)$  da matriz  $\mathbf{P}$  um valor que é menor ou igual a  $\varepsilon$  se  $p_{i,M}^{(M)} > 0$ . Para o caso em que  $p_{i,M}^{(M)} = \varepsilon$ , o algoritmo adiciona  $\varepsilon$  ao  $j$ -ésimo elemento da linha  $i$  se  $p_{M,j}^{(M)} > 0$ ,  $j = 1, \dots, M - 1$ .

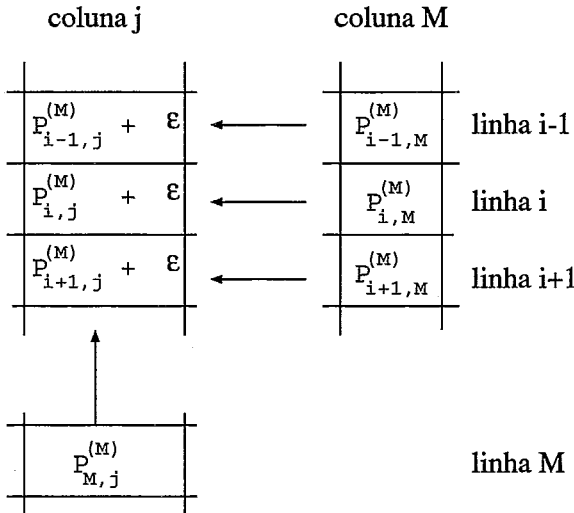


Figura 3.8: Transições com valores  $\varepsilon$ .

Neste ponto é importante fazermos uma observação em relação à Figura 3.7. Como vimos, neste exemplo, a matriz esparsa vai sendo preenchida à medida em que os estados são eliminados na primeira parte do GTH. Se temos transições com valores  $\varepsilon$ , parte da matriz  $\mathbf{P}$  será preenchida com valores menores ou iguais

a  $\varepsilon$ .

A idéia é diminuir o custo computacional do algoritmo GTH eliminando as operações que calculam valores menores ou iguais a  $\varepsilon$ . Uma maneira de fazer isso é redirecionar as transição de saída de um determinado estado que tem valor  $\varepsilon$  para o próprio estado, ou seja, mover estas transições para a diagonal. Este algoritmo utiliza a idéia que algumas transições da matriz  $\mathbf{P}$ , por terem valores muito pequenos, podem ser eliminadas (ou redirecionadas) sem alterar significativamente os valores obtidos com a solução exata do modelo. O problema com esta solução é a impossibilidade de se obter limites para o erro.

A seguir vamos apresentar um algoritmo que calcula o erro introduzido na solução do modelo devido à eliminação de transições com valores  $\varepsilon$ . Antes, porém, vamos estudar o que acontece quando eliminamos uma transição com valor  $\varepsilon$  em alguns casos particulares. Veremos que ao eliminarmos uma transição, podemos alterar significativamente a solução do modelo, mesmo que o valor da transição eliminada seja bem pequeno ( $\varepsilon$ ). Portanto, é preciso cuidado ao se tentar eliminar uma transição  $\varepsilon$ .

### 3.4.1 Transições com valores $\varepsilon$

A Figura 3.9 mostra o estado  $s_i$  de uma cadeia de Markov *ergódica* com matriz de probabilidades de transição  $\mathbf{P}$ ,  $i = 1, \dots, M$ . Neste exemplo, existe uma única transição de entrada para o estado  $s_i$ . Note que o valor da probabilidade de transição desse estado para  $s_i$  é  $\varepsilon$ . Além disso, a probabilidade de saída do estado  $s_i$  para os outros estados da cadeia tem valor  $p$  (probabilidade total de saída do estado  $s_i$ ). Assuma que  $p \gg \varepsilon$ .

Suponha que a transição com valor  $\varepsilon$  na Figura 3.9 é eliminada. Isto implica que  $\pi_i = 0$  pois não mais existe fluxo de entrada no estado  $s_i$ . Logo, o estado  $s_i$  e todas as transições de saída de  $s_i$  podem ser eliminados. Entretanto, é possível

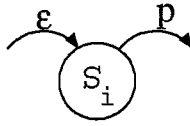


Figura 3.9: O estado  $s_i$  possui uma única transição de entrada.

que com a eliminação da transição  $\epsilon$  tenhamos um problema: a nova cadeia pode não ser ergódica. A Figura 3.10 exemplifica bem esta situação.

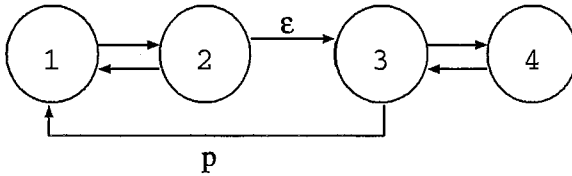


Figura 3.10: Exemplo de cadeia com transição  $\epsilon$ .

No exemplo acima, temos uma cadeia com 4 estados onde a transição do estado 2 para o estado 3 tem valor  $\epsilon$ , enquanto as outras transições da cadeia tem valores bem maiores que  $\epsilon$ . Ao eliminarmos a transição  $\epsilon$ , temos uma cadeia com dois grupos de estados. O primeiro grupo é formado pelos estados 1 e 2 e o segundo grupo é formado pelos estados 3 e 4. Não há transição de estado do primeiro para o segundo grupo na nova cadeia. Isto significa que uma vez que o sistema entra nos estados do primeiro grupo, os estados do segundo grupo não são mais visitados. Portanto, a cadeia não é ergódica e a probabilidade estacionária dos estados do segundo grupo é zero (podemos então eliminar os estados 3 e 4 na nova cadeia). Note que a solução da nova cadeia será tão próxima da solução do modelo original, quanto menor for o valor  $\epsilon$ .

A Figura 3.11 mostra o estado  $s_i$  de uma cadeia de Markov *ergódica* que possui uma única transição de saída. Esta transição tem valor  $\epsilon$ . Além disso, existe pelo menos uma probabilidade  $p$  de entrada no estado  $s_i$  vindo de um outro estado do modelo, onde  $p \gg \epsilon$ .

A eliminação da transição  $\epsilon$  na Figura 3.11 implica que  $\pi_i = 1$  na nova cadeia, pois  $s_i$  é agora um estado absorvente. Note que também podemos ter um estado

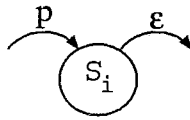


Figura 3.11: O estado  $s_i$  com transição de saída  $\epsilon$ .

absorvente no caso apresentado na Figura 3.9. Entretanto, o estado absorvente não será o estado  $s_i$  e sim, o estado que tem transição de saída  $\epsilon$  para o estado  $s_i$ .

Quando a nova cadeia (gerada com a eliminação de uma transição  $\epsilon$ ) tem estado absorvente, a solução pode ser bem diferente da solução da cadeia original, pois a probabilidade estacionária de um estado absorvente é 1. Portanto, precisamos evitar o aparecimento de estados absorventes quando eliminamos as transições  $\epsilon$ . Para isso, basta verificarmos as transições de saída do estado que dá origem a transição a ser eliminada. Uma transição  $\epsilon$  só será eliminada se o estado de origem possuir outras transições de saída.

**Lema 1.** Seja  $\mathcal{X}$  uma cadeia de Markov de tempo discreto com espaço de estados  $S = \{s_i : i = 1, \dots, M\}$ . Suponha que  $\mathcal{X}$  não possui estados absorventes. Seja  $\mathcal{X}'$  uma cadeia de Markov idêntica à cadeia  $\mathcal{X}$  com a seguinte modificação: a transição  $p_{ij}$  é eliminada, onde  $p_{ij}$  corresponde à probabilidade de transição do estado  $s_i$  para o estado  $s_j$ ,  $\{s_i, s_j\} \in S$  e  $s_i \neq s_j$ . A cadeia  $\mathcal{X}'$  não tem estados absorventes se  $p_{ij}/(1 - p_{ii}) < 1$ .

**Prova.** É trivial. Se  $p_{ij}/(1 - p_{ii}) < 1$  então  $1 - p_{ii} > p_{ij}$ , ou seja, existe uma transição do estado  $s_i$  para o estado  $s_k$ , onde  $k \neq i$  e  $k \neq j$ . Isto significa que após a eliminação da transição  $p_{ij}$ , existirá pelo menos uma transição de saída do estado  $s_i$  e portanto, o estado  $s_i$  da cadeia  $\mathcal{X}'$  não é absorvente.  $\square$

O Lema 1 define a condição necessária e suficiente para que uma transição de saída do estado  $s_i$  seja eliminada sem que  $s_i$  se torne um estado absorvente: é preciso que o estado  $s_i$  possua outras transições de saída. O problema é que

todas as transições de saída de um determinado estado podem ter valor  $\varepsilon$ . Neste caso as transições  $\varepsilon$  podem não ser desprezíveis. A Figura 3.12 exemplifica esta situação.

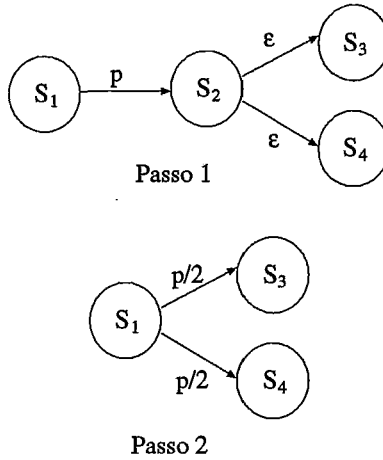


Figura 3.12: Todas as transições de saída de  $s_2$  têm valor  $\varepsilon$ .

O passo 1 da Figura 3.12 mostra 4 estados de uma cadeia de Markov qualquer. O estado  $s_2$  do exemplo no passo 1 tem duas transições de saída, todas com valor  $\varepsilon$ , e apenas uma transição de entrada com valor  $p$ , onde  $p \gg \varepsilon$ . Não podemos eliminar as duas transições  $\varepsilon$  de  $s_2$ , pois o estado  $s_2$  passaria a ser absorvente. E não podemos eliminar apenas uma das transições pois cada transição tem peso  $1/2$  na taxa de saída do estado. Para compreender melhor este problema, veja o passo 2 da Figura 3.12. O passo 2 mostra o que acontece quando eliminamos o estado  $s_2$  usando o GTH. A probabilidade do estado  $s_1$  para o estado  $s_3$  ou para o estado  $s_4$  é  $p/2$ . Portanto, a eliminação de alguma das transições  $\varepsilon$  do estado  $s_2$  pode alterar significativamente a solução da cadeia de Markov original.

A idéia é que a probabilidade de transição  $p_{ij}$  do estado  $s_i$  para o estado  $s_j$  seja considerada *eliminável* se e somente se,

$$\frac{p_{ij}}{1 - p_{ii}} \leq \varepsilon, \quad \text{para } i \neq j.$$

Portanto, a transição do estado  $s_i$  para o estado  $s_j$  só será eliminada quando o

peso da probabilidade  $p_{ij}$  em relação à probabilidade total de saída do estado  $s_i$  for bem pequeno ( $\varepsilon$ ).

Outro problema é a eliminação de transições  $\varepsilon$  em uma cadeia de Markov quase que completamente particionada. A Figura 3.13 mostra um exemplo de uma cadeia quase particionada com 4 estados. Podemos dividir a cadeia em dois grupos, onde os estados 1 e 2 pertencem ao primeiro grupo e os estados 3 e 4 pertencem ao segundo grupo. Ao eliminarmos as transições  $\varepsilon$  temos uma cadeia de Markov particionada e portanto, não ergódica. A solução desse tipo de cadeia pode ser obtida em duas partes. Na primeira parte, é calculada a probabilidade do sistema se encontrar em cada grupo de estados. E na segunda parte, são calculadas as probabilidades condicionais dos estados de cada grupo.

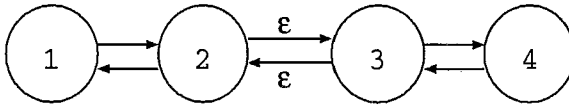


Figura 3.13: Cadeia de Markov quase particionada.

Se aplicarmos as regras acima a cadeia final será particionada em 2. Entretanto, o algoritmo pode facilmente reconhecer a ocorrência desse caso e informar ao usuário da impossibilidade de obter a solução do modelo usando o novo algoritmo.

Precisamos agora definir como as transições  $\varepsilon$  são eliminadas (ou redirecionadas) durante a execução do algoritmo GTH. Considere novamente a cadeia de Markov com espaço de estados  $S = \{s_i : i = 1, \dots, M\}$ . Seja  $p_{ij}$  a probabilidade de transição do estado  $s_i$  para o estado  $s_j$ ,  $\{s_i, s_j\} \in S$ . Queremos eliminar o estado  $S_M$  e calcular as novas probabilidades de transição dos estados restantes. Para  $i = 1, \dots, M$ , de acordo com o que foi discutido acima: a transição  $p_{i,M}$  é eliminada somente se  $p_{i,M}/(1 - p_{i,i}) \leq \varepsilon$ ; e a transição  $p_{M,i}$  é eliminada somente se  $p_{M,i}/(1 - p_{M,M}) \leq \varepsilon$ . Note que só testamos as transições de entrada e de saída do estado  $s_M$ , pois apenas estas transições são usadas pelo GTH no passo que elimina o estado  $S_M$ .



Como observação final, note que as regras para eliminação de transição  $\epsilon$  discutidas acima não garantem que a matriz resultante seja ergódica. Elas apenas garantem que a nova matriz não terá estados absorventes e que a eliminação de uma transição não altera significativamente a solução do modelo. O algoritmo identifica que a matriz não é ergódica quando o estado a ser eliminado tem na matriz de transição a coluna acima da diagonal zerada. Neste caso, o algoritmo informa ao usuário da ocorrência desse caso (a probabilidade estacionária do estado é zero) e continua a execução.

### 3.4.2 Algoritmo Proposto

A Figura 3.14 mostra a estrutura de uma matriz de probabilidades de transição com 12 estados no início e no final da execução da primeira parte do algoritmo GTH. A matriz inicial é esparsa e possui algumas transições com valores  $\epsilon$ , enquanto a matriz final é quase toda preenchida. Note que, se as transições com valores  $\epsilon$  fossem movidas para a diagonal, a nova matriz teria estrutura banda e portanto, o custo da solução seria menor. Como mencionamos, o problema com esta proposta é a impossibilidade de calcular o erro dessa solução em relação a solução exata da cadeia original.

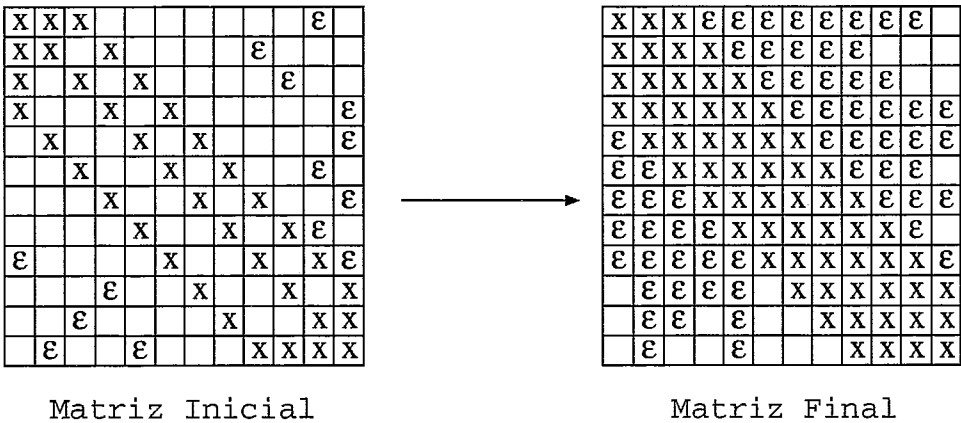


Figura 3.14: Algoritmo GTH.

Seja  $P$  uma matriz de probabilidades de transição com  $M$  estados. De acordo

com o método de [31] podemos transformar a matriz  $\mathbf{P}$  de  $M$  estados em uma matriz de transição  $\mathbf{P}_s$  com  $M + 1$  estados como mostra a Figura 3.15. Suponha que o espaço de estados de  $\mathbf{P}$  é  $S = \{s_1, \dots, s_M\}$  e que o espaço de estados de  $\mathbf{P}_s$  é  $S' = \{s_\varepsilon\} + S$ . Como discutido em 3.3.1, temos  $\mathbf{P}^- = \mathbf{P} - \mathbf{P}^\varepsilon$ ,  $y^T = \mathbf{P}^\varepsilon e^T$  e  $x e^T = 1$ , onde  $\mathbf{P}^\varepsilon$  possui todas as transições  $\varepsilon$ . Isto significa que o vetor  $y^T$  representa as transições  $\varepsilon$  redirecionadas para estado  $s_\varepsilon$ , e o vetor  $x$  representa as probabilidades de transição do estado  $s_\varepsilon$  para os outros estados da cadeia. O vetor  $x$  é definido de forma que o vetor das probabilidades estacionárias de  $\mathbf{P}$  seja igual ao vetor das probabilidades condicionais dos  $M$  últimos estados de  $\mathbf{P}_s$ .

0	$x$
$y^T$	$\mathbf{P}^-$

Figura 3.15: Nova matriz  $\mathbf{P}_s$ .

**Lema 2.** Seja  $\mathbf{P}$  a matriz de probabilidades de transição de uma cadeia de Markov com  $M$  estados. Seja  $\mathbf{P}^-$  um limite inferior para  $\mathbf{P}$ , ou seja,  $\mathbf{P}^- \leq \mathbf{P}$ . Seja  $\mathbf{P}_s$  a matriz estocástica da Figura 3.15, onde  $y^T = \mathbf{P}^\varepsilon e^T$  e  $x e^T = 1$ . Então existe um vetor  $x$  tal que as probabilidades condicionais dos  $M$  últimos estados de  $\mathbf{P}_s$  é igual às probabilidades estacionárias dos estados de  $\mathbf{P}$ .

**Prova.** Veja [31].

A Figura 3.16 mostra o que pode acontecer quando redirecionamos as transições  $\varepsilon$  para um novo estado. Note que usamos o mesmo exemplo da Figura 3.14. Comparando as matrizes finais dos dois exemplos, vemos que os valores  $\varepsilon$  ocupam boa parte da matriz do primeiro exemplo, enquanto ocupam apenas uma coluna da segunda matriz. Para o primeiro exemplo temos um custo de 1012 operações

na primeira parte do GTH, enquanto o segundo exemplo tem um custo de 132 operações. Caso a matriz de probabilidades de transição tenha 1000 estados e mesma estrutura apresentada acima, o custo do primeiro modelo seria de mais de 500 milhões de operações, enquanto o custo (aproximado) do segundo modelo ficaria em torno de 13.000 operações. Este exemplo é apenas para dar idéia ao leitor de como podemos economizar ao redirecionarmos as transições  $\varepsilon$  para um novo estado. Naturalmente, queremos usar o novo método na solução de matrizes com milhares de estados. Além disso, a matriz  $\mathbf{P}^-$  não necessariamente tem uma estrutura banda como mostra este exemplo.

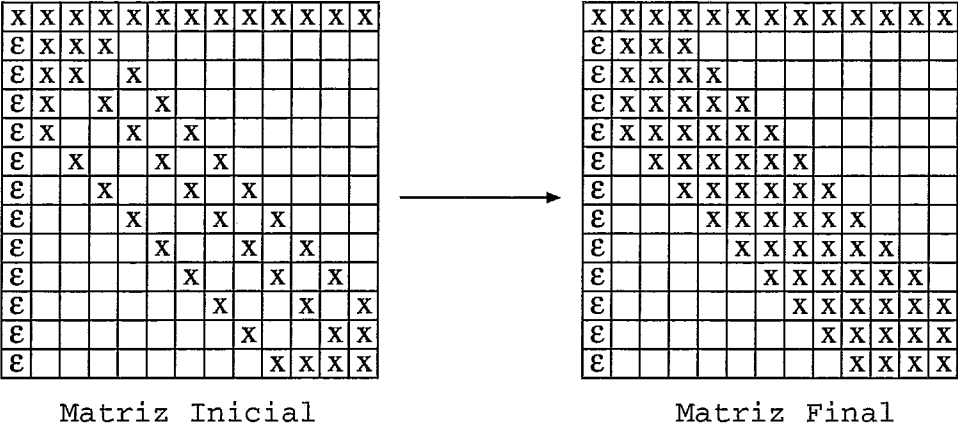


Figura 3.16: Transições  $\varepsilon$  são desviadas para um novo estado.

O problema é que para calcular  $x$  precisamos da solução da matriz  $\mathbf{P}$ . Entretanto, podemos usar os resultados de Courtois e Semal [36, 37] para obter uma aproximação para a solução de  $\mathbf{P}_s$ . Para isto, precisamos fazer cada elemento do vetor  $x$  igual a 1 e solucionar a cadeia correspondente. O menor e o maior valores obtidos para as probabilidades estacionárias dos  $M$  últimos estados de  $\mathbf{P}_s$  fornecem um limite inferior e superior para as probabilidades estacionárias dos estados de  $\mathbf{P}$ . Infelizmente, esta solução pode sair mais cara que a solução do modelo original, pois é necessário solucionar  $\mathbf{P}_s$  várias vezes. Além disso, é possível que novas transições  $\varepsilon$  sejam geradas a cada passo da primeira parte do GTH e portanto, estas transições também devem ser redirecionadas para o estado  $s_\varepsilon$ . Isto significa que pode ser necessário utilizar os resultados de Courtois e Semal [36, 37] em cada passo da primeira parte do GTH.

Resumindo, o redirecionamento das transições  $\varepsilon$  para o estado  $s_\varepsilon$  fornece uma solução exata do modelo se as transições de saída de  $s_\varepsilon$  são conhecidas. Infelizmente, para calcular os valores dessas transições precisamos da solução do modelo original. Portanto, nosso objetivo é tentar obter um limite para o erro da solução do modelo aproximado e ainda conseguir ganhos no cálculo desta solução. A idéia é substituir o estado  $s_\varepsilon$  por um conjunto de estados *clones* dos estados da cadeia como definido no método de [32]. A seguir, discutiremos com mais detalhes o método. Antes porém, vamos introduzir alguns conceitos sobre modelos markovianos com recompensas. Estes conceitos são importantes para podermos apresentar o Lema 3. Discutiremos este tipo de modelagem com mais detalhes no Capítulo 4.

Seja  $\mathcal{R}$  a recompensa média acumulada por  $\mathcal{X}$  durante o intervalo  $(0, t)$ , quando  $t \rightarrow \infty$ . Temos então  $\mathcal{R} = \sum_{i=1}^M \pi_i r_i$ , onde  $\pi_i$  e  $r_i$  correspondem, respectivamente, à probabilidade estacionária do estado  $s_i$  (portanto igual à fração de tempo em  $s_i$ ) e à recompensa associada a  $s_i$  ganha por unidade de tempo passada em  $s_i$ . Suponha que  $r_{lb}$  é a menor recompensa associada a um estado da cadeia  $\mathcal{X}$  e  $r_{ub}$  é a maior recompensa associada a um estado de  $\mathcal{X}$ , isto é,  $r_{lb} \leq r_i \leq r_{ub}$  para todo  $i = 1, \dots, M$ . Além disso, suponha que o espaço de estados é particionado em dois sub-conjuntos:  $\mathcal{G}_1$  e  $\mathcal{G}_2$ . Então, podemos escrever [38]

$$\mathcal{R} = P(\mathcal{G}_1)\mathcal{R}_1 + P(\mathcal{G}_2)\mathcal{R}_2, \quad (3.4)$$

onde  $P(\mathcal{G}_1)$  ( $P(\mathcal{G}_2)$ ) é a probabilidade do sistema se encontrar no subconjunto de estados  $\mathcal{G}_1$  ( $\mathcal{G}_2$ ) e  $\mathcal{R}_1$  ( $\mathcal{R}_2$ ) é o valor de  $\mathcal{R}$  condicionado que o sistema se encontra em um dos estados de  $\mathcal{G}_1$  ( $\mathcal{G}_2$ ). Se as recompensas são limitadas, então  $r_{lb} \leq \mathcal{R}_1 \leq r_{ub}$  e  $r_{lb} \leq \mathcal{R}_2 \leq r_{ub}$  e temos [38]

$$\begin{aligned} \mathcal{P}_{lb}(\mathcal{G}_1)\mathcal{R}_{1,lb} + (1 - \mathcal{P}_{lb}(\mathcal{G}_1))r_{lb} &\leq \mathcal{R} \leq \mathcal{P}_{lb}(\mathcal{G}_1)\mathcal{R}_{1,ub} + (1 - \mathcal{P}_{lb}(\mathcal{G}_1))r_{ub} \\ r_{lb} + \mathcal{P}_{lb}(\mathcal{G}_1)(\mathcal{R}_{1,lb} - r_{lb}) &\leq \mathcal{R} \leq r_{ub} - \mathcal{P}_{lb}(\mathcal{G}_1)(r_{ub} - \mathcal{R}_{1,ub}), \end{aligned} \quad (3.5)$$

onde  $\mathcal{P}_{lb}(\mathcal{G}_1)$  é um limite inferior para a soma das probabilidades dos estados de  $\mathcal{G}_1$  e  $\mathcal{R}_{1,lb}$  ( $\mathcal{R}_{1,ub}$ ) é um limite inferior (limite superior) para o valor de  $\mathcal{R}_1$ . A equação 3.5 indica que podemos obter limites para  $\mathcal{R}$  a partir do limite inferior e do limite superior de  $\mathcal{R}_1$  e de um limite inferior de  $P(\mathcal{G}_1)$ , ou seja, a partir de limites obtidos para os estados do primeiro grupo ( $\mathcal{G}_1$ ).

Seja  $\mathcal{X}$  uma cadeia de Markov de tempo contínuo com espaço de estados  $S = \{s_1, \dots, s_M\}$ . Suponha que existam transições com valores  $\varepsilon$  entre alguns dos estados de  $\mathcal{X}$ . Podemos transformar  $\mathcal{X}$  na cadeia de Markov  $\mathcal{X}'$  de tempo contínuo e espaço de estados  $S' = \{s_1, \dots, s_M, s_\varepsilon\}$ , onde todas as transições  $\varepsilon$  dos  $M$  primeiros estados de  $\mathcal{X}$  são redirecionadas para o estado  $s_\varepsilon$  em  $\mathcal{X}'$  como discutido anteriormente.

Seja  $s_{inicial}$  um estado de  $\mathcal{X}'$  tal que  $s_{inicial} \in \{s_1, \dots, s_M\}$ . Assuma que  $K$  é a maior distância de um estado de  $\mathcal{X}'$  para o estado  $s_{inicial}$ , ou seja, o menor caminho entre um estado de  $\mathcal{X}'$  e o estado  $s_{inicial}$  nunca é maior que  $K$  transições. Considere a partição  $\mathcal{S} = \{S_0, S_1, \dots, S_K\}$  da cadeia  $\mathcal{X}'$ , onde  $S_i$  corresponde ao conjunto de estados de  $\mathcal{X}'$  que estão a  $i$  passos de  $s_{inicial}$ ,  $i = 1, \dots, K$ .

Segundo o método de [32] podemos gerar a cadeia de Markov  $\mathcal{X}''$  de tempo contínuo a partir da cadeia  $\mathcal{X}'$  da seguinte forma:

- é adicionado um estado  $f_i$  (*clone*) para cada partição  $S_i$ ,  $i = 1, \dots, k$ ;
- as taxas de transições entre os estados *clones* e entre o estado  $f_1$  e a partição  $S_0$  são calculadas usando as mesmas regras apresentadas em [32] (veja seção 3.3.2);
- uma recompensa  $r_{lb}$  é associada a cada estado  $f_i$  para todo  $1 \leq i \leq K$ .

É importante observar que o estado  $s_\varepsilon$  de  $\mathcal{X}'$  é substituído pelos estados clones de  $\mathcal{X}''$  e as transições de um estado  $s$  para  $s_\varepsilon$  são agora direcionadas para  $f_k$ . A Figura 3.17 mostra o novo processo markoviano  $\mathcal{X}''$ , onde definimos  $S^* = \{S_1, \dots, S_K\}$ . Note que as transições  $\varepsilon$  de  $S_0$  e de qualquer estado de  $S^*$  são redirecionadas para  $f_K$ .

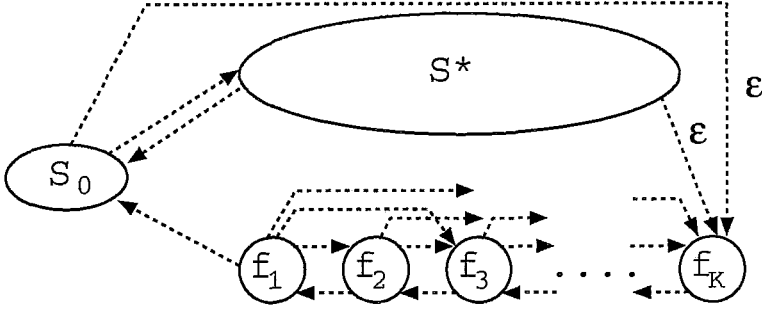


Figura 3.17: Processo markoviano descrito por  $\mathcal{X}''$ .

**Lema 3.** Seja  $\mathcal{R}''$  a recompensa média acumulada por  $\mathcal{X}''$  durante o intervalo  $(0, t)$ , quando  $t \rightarrow \infty$ . Então,  $\mathcal{R}'' \leq \mathcal{R}$ .

**Prova.** Veja [32].

Veja que  $S_0$  e  $S^*$  correspondem ao sub-conjunto  $\mathcal{G}_1$  em 3.4 e que os estados clones correspondem ao sub-conjunto  $\mathcal{G}_2$ . Isto nos permite fazer a seguinte afirmação sobre as soluções de  $\mathcal{X}$  e  $\mathcal{X}''$ : as probabilidades estacionárias dos estados  $\{s_1, \dots, s_M\}$  de  $\mathcal{X}''$  são limites inferiores para as probabilidades estacionárias dos estados  $\{s_1, \dots, s_M\}$  de  $\mathcal{X}$ . Isto é verdade pois, pelas regras acima, as transições entre estados  $f_i$  são escolhidas de forma a aumentar o tempo de permanência nestes estados clones o que consequentemente diminui o tempo relativo nos estados de  $S_0$  e  $S^*$  (ver [32] para maiores detalhes).

Vimos que a partição  $\mathcal{S}$  de  $\mathcal{X}''$  depende da escolha do estado  $s_{inicial}$ . O método de [32] apenas supõe que  $s_{inicial}$  é um estado importante para a medida de interesse a ser calculada para o modelo. Infelizmente, a escolha de  $s_{inicial}$  não é trivial.

Para facilitar a discussão sobre a escolha de  $s_{inicial}$ , vamos novamente usar

o modelo de filas apresentado em [40] e discutido na seção 3.3.2. Este modelo foi escolhido devido à sua simplicidade e por permitir mostrar os problemas que podemos encontrar para aplicar o algoritmo discutido acima. Convém, entretanto, enfatizar que os parâmetros do modelo usados em [40] gera cadeias com milhares de estados. Além disso, não existem taxas  $\varepsilon$  definidas nos exemplos.

Relembrando, temos um modelo markoviano com duas filas, onde cada fila tem 3 servidores e capacidade de armazenamento de 4 pacotes. A taxa de chegada, a taxa de serviço e o número de pacotes da fila  $i$  é, respectivamente,  $\lambda_i$ ,  $\mu_i$ , e  $n_i$ ,  $i = 1, 2$ . Se um pacote chega na fila 1 e encontra a fila cheia, ele é enviado para a fila 2. O estado do modelo é representado por  $n_1$  e  $n_2$  (veja a Figura 3.5).

Similar ao que vimos na seção 3.3.2, suponha que  $s_{inicial}$  corresponde ao estado onde  $n_1 = 0$  e  $n_2 = 0$ , ou seja, as duas filas estão vazias. Então o estado *clone*  $f_i$ , segundo o método de [32], corresponde neste exemplo ao agrupamento dos estados que possuem um total de  $i$  pacotes no sistema.

Para exemplificar o modelo, vamos assumir que  $\lambda_1 = 3$ ,  $\mu_1 = 3$ ,  $\lambda_2 = 1$  e  $\mu_2 = 1$ . Como definida em [32], a taxa de  $f_i$  para  $f_{i-1}$  é igual à menor soma das taxas de saída de um estado de  $S_i$  para os estados de  $S_{i-1}$ ,  $i = 1, \dots, 8$ . Por outro lado, a taxa de  $f_i$  para  $f_{i+1}$  é igual à maior soma das taxas de saída de um estado de  $S_i$  para os estados de  $f_{i+1}$ ,  $i = 1, \dots, 7$ . Não existem taxas de transição de  $f_i$  para  $f_{i+l}$  quando  $l > 1$  para este modelo. Seja  $\lambda$  a taxa de  $f_i$  para  $f_{i+1}$  e seja  $\mu$  a taxa de  $f_i$  para  $f_{i-1}$ . A Figura 3.18 mostra as taxas de transição dos estados *clones* para este exemplo definidas segundo o método de [32] obtidas a partir da Figura 3.5. Note que  $\lambda > \mu$  para os quatro primeiros estados e  $\lambda \leq \mu$  nos outros estados. Isto significa que a probabilidade estacionária dos estados  $f_1$  pode não ser desprezível e portanto, a solução de  $\mathcal{X}''$  pode ser bem diferente da solução de  $\mathcal{X}$ .

Para que o método de [32] forneça uma boa aproximação, a transição de saída de  $f_i$  para  $f_{i-1}$  deve, em geral, ser maior ou igual à transição de  $f_i$  para  $f_{i+1}$ ,

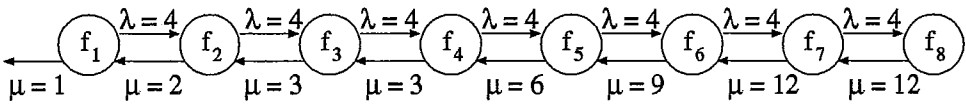


Figura 3.18: Estados clones do modelo com 2 filas de pacotes.

para  $i = 1, \dots, M - 1$  (note também que pela equação 3.5 os limites dependem também das recompensas atribuídas aos estados). Infelizmente, dependendo dos valores de  $\lambda$  e  $\mu$ , pode não ser possível encontrar para este modelo de filas, um único estado  $s_{inicial}$  que atenda a esta condição. Suponha então que a partição  $S_0$  é composta de um conjunto de estados ao invés de um único estado. Para o exemplo do modelo de filas, façamos a partição  $S_0$  igual a todos os estados em que a fila 1 está vazia. Portanto,  $S_i$  contém todos os estados em que a fila 1 tem  $i$  pacotes,  $i = 1, \dots, 4$ . A Figura 3.19 mostra as taxas de transição dos estados para o mesmo exemplo definido acima. Como as transições entre partições correspondem as transições devido a uma chegada ou a uma saída da fila 1, temos  $\lambda = \lambda_1$  e  $\mu = \mu_1$ . Podemos verificar que agora temos  $\lambda \leq \mu$  para todo  $f_i$ ,  $i = 1, 2, 3$ .

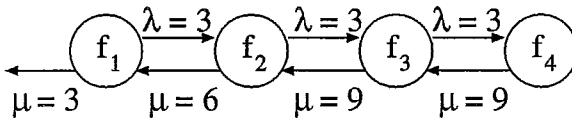


Figura 3.19: Novos estados clones do modelo com 2 filas de pacotes.

Como podemos ver a escolha dos estados de  $S_0$  depende da análise do modelo. Por isso, usamos a ferramenta TANGRAM-II [19] [20] [21] para estudar os modelos apresentados neste capítulo. Os modelos na ferramenta são definidos de acordo com a Metodologia Orientada a Objetos proposta em [18]. Um modelo é visto como um conjunto de objetos, onde cada objeto é especificado através de uma ou mais variáveis de estado. O conjunto de todos os valores possíveis dessas variáveis fornece os estados do sistema. Por exemplo, para o modelo de filas discutido acima temos 2 objetos, *fila 1* e *fila 2*, e cada objeto possui uma única variável de estado (variável  $n_1$  da *fila 1* e variável  $n_2$  da *fila 2*). Os estados de  $S_0$  são definidos como aqueles que possuem  $n_1 = 1$ .



Fazer da partição  $S_0$  um conjunto de estados nos cria um novo problema. Sabemos a taxa de transição de  $f_1$  para  $S_0$ , mas não sabemos como esta taxa é distribuída entre os estados de  $S_0$ .

**Lema 4.** Seja  $q$  a taxa de transição do estado  $f_1$  para a partição  $S_0$  na cadeia  $\mathcal{X}''$ . Faça a transição de  $f_1$  para um dos estados de  $S_0$  igual a  $q$  e encontre a solução do modelo. Repita esta operação para cada estado de  $S_0$ . A solução de  $\mathcal{X}''$  corresponde aos menores valores obtidos com cada solução. As probabilidades estacionárias dos estados  $\{s_1, \dots, s_M\}$  de  $\mathcal{X}''$  é um limite inferior para as probabilidades dos estados  $\{s_1, \dots, s_M\}$  de  $\mathcal{X}$ .

**Prova.** Como definido acima,  $\mathcal{X}$  é uma cadeia de Markov de tempo contínuo com espaço de estados  $S = \{s_1, \dots, s_M\}$  e  $\mathcal{X}'$  é a cadeia de Markov de tempo contínuo e espaço de estados  $S' = \{s_1, \dots, s_M, s_\varepsilon\}$ , onde todas as transições  $\varepsilon$  dos  $M$  primeiros estados de  $\mathcal{X}$  são redirecionadas para o estado  $s_\varepsilon$  em  $\mathcal{X}'$ . Usamos então o método de [32] para gerar, a partir de  $\mathcal{X}'$ , a cadeia de Markov  $\mathcal{X}''$  com espaço de estados  $S'' = \{s_1, \dots, s_M, f_1, \dots, f_k\}$ . Suponha que conhecemos as taxas de transição de  $f_1$  para os estados de  $S_0$ . De [32] sabemos que a solução de  $\mathcal{X}''$  é um limite inferior para a solução de  $\mathcal{X}$ . Infelizmente, não temos os valores das transições de  $f_1$  para  $S_0$ . Entretanto, podemos usar o método de Courtois e Semal [36, 37], para obter um limite: a taxa total de saída de  $f_1$  é direcionada para apenas um dos estados de  $S_0$  e é obtida a solução do modelo correspondente; esta operação é repetida para cada um dos estados de  $S_0$ . A solução de  $\mathcal{X}''$  corresponde ao menor valor obtido com cada solução do modelo e é um limite inferior para a solução de  $\mathcal{X}$ .  $\square$

O Lema 4 mostra que se o número de estados da partição  $S_0$  for  $m$ , teremos que solucionar o modelo  $m$  vezes, onde a taxa de transição de  $f_1$  para um dos estados de  $S_0$  é igual à taxa total de transição de  $f_1$  para  $S_0$ . Infelizmente, usar o método de Courtois e Semal na solução do modelo descrito acima pode ser mais caro que usar o algoritmo GTH na solução do modelo original. Isto pode ser facilmente explicado pelas operações executadas na segunda parte do algoritmo

que tem um custo da  $O(M^2)$  para um modelo com  $M$  estados. A seguir vamos discutir uma alternativa para o uso do Lema 4.

### 3.4.3 Melhorando o Algoritmo Proposto

Seja  $\mathcal{X}$  uma cadeia de Markov de tempo contínuo com espaço de estados  $S = \{s_1, \dots, s_M\}$ . Suponha que existam transições com valores  $\varepsilon$  entre os estados de  $\mathcal{X}$ . Além disso,  $\mathcal{X}'$  é a cadeia de Markov de tempo contínuo e espaço de estados  $S' = \{s_1, \dots, s_M, s_\varepsilon\}$ , onde todas as transições  $\varepsilon$  dos  $M$  primeiros estados de  $\mathcal{X}$  são redirecionadas para o estado  $s_\varepsilon$  em  $\mathcal{X}'$ . Seja  $\mathcal{X}''$  a cadeia de Markov de tempo contínuo com espaço de estados  $S'' = \{f_1, \dots, f_k, s_1, \dots, s_M\}$  gerada a partir da partição  $\mathcal{S} = \{S_0, S_1, \dots, S_K\}$  de  $\mathcal{X}'$  usando o método de [32], onde  $S_i$  corresponde ao conjunto de estados que estão a  $i$  passos de  $S_0$ ,  $i = 1, \dots, K$ .

A Figura 3.20 mostra a estrutura da matriz de transição de  $\mathcal{X}''$ . Note que estamos supondo que a numeração dos estados é feita na seguinte ordem: estados clones, estados de  $S_0$  e o resto dos estados do modelo. Para facilitar a discussão e apresentação do Lema 5 vamos supor que  $S'' = \{e_1, \dots, e_{M+k}\}$ , isto é, o  $i$ -ésimo estado de  $S''$  é denominado  $e_i$ , portanto  $e_1 = f_1, \dots, e_k = f_k, e_{k+1} = s_1, \dots, e_{M+k} = s_M$ . É interessante aqui lembrarmos de dois fatos que irão ajudar a entender o Lema 5. Em primeiro lugar, os estados são eliminados pelo algoritmo na seguinte ordem:  $e_{M+k}, e_{M+k-1}, \dots, e_2$ . Em segundo lugar, não conhecemos os valores das transições de  $e_1$  ( $f_1$ ) para os estados de  $S_0$ .

**Lema 5.** Suponha que  $S_0 = \{s_{k+1}, \dots, s_{m+k}\}$ , isto é,  $S_0$  é formado pelos  $m$  primeiros estados da cadeia após os estados clones. Seja  ${}^l\pi_i$  a probabilidade estacionária (não normalizada) do estado  $e_i \in S''$  quando a transição de  $e_1$  para  $S_0$  é feita somente através do estado  $e_{k+l}$ ,  $1 \leq l \leq m$ . Sejam

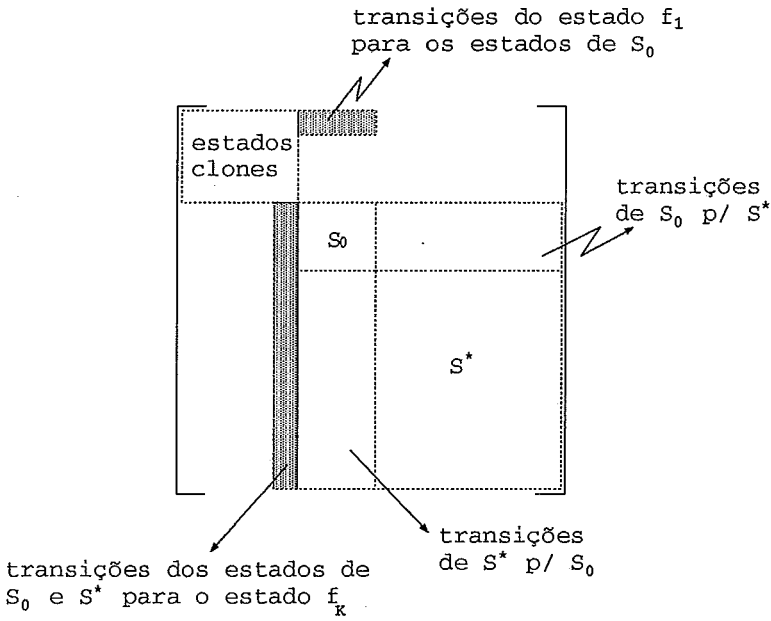


Figura 3.20: Estrutura da matriz de  $\mathcal{X}''$ .

$$\pi_i^{min} = \begin{cases} \min_{1 \leq j \leq m} \{^j \pi_i\} & \text{para } 1 \leq i \leq m+k \\ \sum_{j=1}^{i-1} \frac{p_{ji}^{(i)}}{1 - p_{ii}^{(i)}} \pi_j^{min} & \text{para } m+k+1 \leq i \leq M+k \end{cases} \quad (3.6)$$

e

$$\pi_i^{max} = \begin{cases} \max_{1 \leq j \leq m} \{^j \pi_i\} & \text{para } 1 \leq i \leq m+k \\ \sum_{j=1}^{i-1} \frac{p_{ji}^{(i)}}{1 - p_{ii}^{(i)}} \pi_j^{max} & \text{para } m+k+1 \leq i \leq M+k \end{cases} \quad (3.7)$$

Para  $i = 1, \dots, M$ , temos

$$\frac{\pi_{i+k}^{min}}{\sum_{j=1}^{M+k} \pi_j^{max}} \leq \pi_i \quad \text{e} \quad \text{Erro} = 1 - \frac{\sum_{i=1}^M \pi_{i+k}^{min}}{\sum_{j=1}^{M+k} \pi_j^{max}},$$

onde  $p_{ji}^{(i)}$  é a probabilidade de transição do estado  $s_j$  para o estado  $s_i$  da matriz  $\mathbf{A}_i$  (a cadeia de Markov nesse passo do GTH tem  $i$  estados) e  $\pi_i$  é a probabilidade estacionária (normalizada) do estado  $s_i$  na solução exata do modelo.

**Prova.** Aplicando-se o resultado de Courtois e Semal para encontrar as probabilidades de todos os  $M + k$  estados da cadeia, a probabilidade de  $e_1 (f_1)$  para  $e_{k+1} (s_1)$  é feita igual à probabilidade total de transição do estado  $e_1 (f_1)$  para a partição  $S_0$  (a transição para os outros estados de  $S_0$  é zero) e são encontradas as probabilidades estacionárias (não normalizadas) dos estados  ${}^1\pi_1, \dots, {}^1\pi_{M+k}$ . Em seguida, a probabilidade de  $e_1 (f_1)$  para  $e_{k+2} (s_2)$  é feita igual à probabilidade total de transição do estado  $e_1 (f_1)$  para a partição  $S_0$  e são encontradas as probabilidades (não normalizadas) dos estados  ${}^2\pi_1, \dots, {}^2\pi_{M+k}$ . E assim em diante, até que a transição de  $e_1 (f_1)$  para  $e_{k+m} (s_m)$  é feita igual à probabilidade total de transição do estado  $e_1 (f_1)$  para a partição  $S_0$  e as probabilidades (não normalizadas) dos estados são calculadas  ${}^m\pi_1, \dots, {}^m\pi_{M+k}$ . A probabilidade normalizada do estado  $e_i$  quando o retorno dos estados clones é feito através do estado  $e_{k+l}$  é calculada como  ${}^l\pi_i / (\sum_{j=1}^{M+k} {}^l\pi_j)$ . O menor valor normalizado encontrado para o estado  $e_i$  é um limite inferior para a probabilidade do estado na solução exata do modelo, isto é,

$$\min_{1 \leq l \leq m} \left\{ \frac{{}^l\pi_i}{\sum_{j=1}^{M+k} {}^l\pi_j} \right\} = \pi_i^{\min} \leq \pi_i.$$

Suponha então que ao invés de calcularmos as probabilidades de todos os  $M + k$  estados da cadeia  $\mathcal{X}'$  resolvendo-se  $m$  modelos diferentes conforme indicado acima, calculamos as probabilidades apenas dos estados clones e dos estados de  $S_0$ , ou seja, dos  $k + m$  primeiros estados. O primeiro conjunto de passos do método GTH elimina estados do modelo e, a cada passo obtém-se uma matriz de cardinalidade inferior à anterior. Note que a eliminação dos  $M - m$  últimos estados da Figura 3.20 independe das transições dos estados clones ao conjunto  $S_0$ . Portanto, esses passos podem ser realizados independentemente do modelo. Em seguida resolvemos  $k$  modelos com  $k + m$  estados aplicando-se Courtois e Semal na matriz obtida após a eliminação de  $M - m$  estados.

Considere então  $\pi_i^{\min}$  e  $\pi_i^{\max}$  como, respectivamente, a menor e a maior probabilidade não normalizada do estado  $e_i$  encontradas para  $i = 1, \dots, k + m$ , isto

é,

$$\pi_i^{min} = \min\{\pi_i^1, \dots, \pi_i^m\} \quad (3.8)$$

e

$$\pi_i^{max} = \max\{\pi_i^1, \dots, \pi_i^m\}. \quad (3.9)$$

Note que temos um valor mínimo e um valor máximo para as probabilidades estacionárias (não normalizadas) dos estados  $e_1, \dots, e_{k+m}$ . Precisamos agora calcular as probabilidades estacionárias dos estados  $e_{k+m+1}, \dots, e_{M+k}$ . Sabemos que

$${}^l\pi_i = \sum_{j=1}^{i-1} \frac{p_{ji}^{(i)}}{1 - p_{ii}^{(i)}} {}^l\pi_j, \quad \text{para } i = 1, \dots, M+k \text{ e } l = 1, \dots, m. \quad (3.10)$$

Portanto, podemos afirmar a partir de 3.8, 3.9 e 3.10 que

$$\sum_{j=1}^{k+m} \frac{p_{j,k+m+1}^{(k+m+1)}}{1 - p_{k+m+1,k+m+1}^{(k+m+1)}} \pi_j^{min} \leq \min_{1 \leq j \leq m} \{\pi_{k+m+1}^j\} \leq \sum_{j=1}^{k+m} \frac{p_{j,k+m+1}^{(k+m+1)}}{1 - p_{k+m+1,k+m+1}^{(k+m+1)}} \pi_j^{max}. \quad (3.11)$$

A equação 3.11 usa os valores mínimos (máximos) das probabilidades estacionárias dos estados  $e_1, \dots, e_{k+m}$  para calcular um limite inferior (superior) para a probabilidade estacionária (não normalizada) do estado  $e_{k+m+1}$ . Podemos então definir

$$\pi_{k+m+1}^{min} = \sum_{j=1}^{k+m} \frac{p_{j,k+m+1}^{(k+m+1)}}{1 - p_{k+m+1,k+m+1}^{(k+m+1)}} \pi_j^{min}$$

e

$$\pi_{k+m+1}^{max} = \sum_{j=1}^{k+m} \frac{p_{j,k+m+1}^{(k+m+1)}}{1 - p_{k+m+1,k+m+1}^{(k+m+1)}} \pi_j^{max}.$$

Similarmente podemos calcular  $\pi_{k+m+2}^{min}$  e  $\pi_{k+m+2}^{max}$ . E assim sucessivamente, até  $\pi_{M+k}^{min}$  e  $\pi_{M+k}^{max}$ . No final temos dois vetores com as probabilidades não normalizadas dos estados da cadeia:  $\pi^{min} = \{\pi_1^{min}, \dots, \pi_{M+k}^{min}\}$  e  $\pi^{max} = \{\pi_1^{max}, \dots, \pi_{M+k}^{max}\}$ . Note que  $\pi^{min}$  e  $\pi^{max}$  podem possuir valores menores ou maiores que 1, já que os vetores não estão normalizados. Note que não podemos simplesmente normalizar o vetor  $\pi^{min}$  pois os seus componentes não seriam um limite para os componentes de  $\pi$ .

Suponha que o menor valor normalizado obtido para a probabilidade estacionária do estado  $e_{i+k}$  ocorre quando a transição de  $e_1$  ( $f_1$ ) para  $S_0$  é feita somente através do estado  $e_{k+l}$ ,  $i = 1, \dots, M$  e  $l = 1, \dots, m$ . Isto significa que

$$\frac{l\pi_{i+k}}{\sum_{j=1}^{M+k} l\pi_j} \leq \pi_i.$$

Como  $\pi_{i+k}^{min} \leq l\pi_i \leq \pi_{i+k}^{max}$ , temos

$$\frac{\pi_{i+k}^{min}}{\sum_{j=1}^{M+k} \pi_j^{max}} \leq \frac{l\pi_{i+k}}{\sum_{j=1}^{M+k} l\pi_j}$$

e portanto,

$$\frac{\pi_{i+k}^{min}}{\sum_{j=1}^{M+k} \pi_j^{max}} \leq \pi_i.$$

□

### 3.5 Exemplos

Nesta seção são mostrados resultados numéricos de três modelos usando os algoritmos discutidos neste capítulo. O primeiro é o modelo de *jitter* apresentado no Capítulo 2. O segundo modelo é o sistema de *polling* com serviço *gated* descrito em [43, 44]. E o terceiro modelo é um servidor de disco modelado em [45].

### 3.5.1 Modelo 1: Jitter

Considere o exemplo mostrado no Capítulo 2 onde o desvio padrão para o *jitter* na saída da rede é 36 (escolhemos este exemplo devido às probabilidades de transição entre estados terem valores bem distintos, a menor probabilidade de transição é  $1.8e-04$  e a maior é  $9.1e-01$ , isto dá origem ao aparecimento de valores  $\varepsilon$  na matriz, de forma a exemplificar o método aqui proposto). Neste modelo o processo de entrega de pacotes só é inicializado após um tempo  $T_1$  da chegada do primeiro pacote e o tempo de entrega de um pacote é fixo e igual a  $T_2$ . A capacidade de armazenamento do equipamento é de  $B$  pacotes. Pacotes que chegam e encontram a fila cheia são descartados. A medida de interesse desejada para este modelo é a probabilidade do *jitter* ser maior que um determinado valor após o início da entrega de pacotes.

O modelo é definido usando 3 objetos: *fonte* que é responsável pela geração dos pacotes; *fila* (ou *buffer*) que armazena os pacotes; e *servidor* que entrega os pacotes ao destino. Cada objeto é caracterizado por uma ou mais variáveis de estado (veja Figura 2.7). O objeto *fila* possui duas variáveis de estado: número total de pacotes na fila ( $n_1$ ) e número de pacotes armazenados na fila antes do último período de silêncio ( $n_2$ ). Os objetos *fonte* e *servidor* possuem uma variável de estado cada um que são, respectivamente, o estado da fonte ( $f$ ) e o estado do servidor ( $s$ ).

Visando exemplificar o uso dos algoritmos discutidos neste capítulo, inicialmente iremos calcular a medida de interesse assumindo que o tempo de permanência em qualquer estado do modelo é exponencialmente distribuído. Por exemplo, o intervalo entre duas entregas de pacote é exponencialmente distribuído com média  $1/T_2$ . Em seguida, vamos calcular a medida de interesse para o modelo com transições determinísticas.

A Figura 3.22 mostra a estrutura da matriz de probabilidades de transição do segundo modelo de *jitter* apresentado no Capítulo 2 quando  $B = 50$ . Esta matriz

possui 2801 estados e é esparsa, sendo que as probabilidades com valores maiores que zero se concentram em torno da diagonal. Note que usamos a ferramenta TANGRAM-II [20, 21] para visualizar a estrutura da matriz. Esta ferramenta apresenta os elementos da matriz com cores que variam de branco (probabilidade zero) a preto (probabilidade 1). Probabilidades pequenas (próximas de zero) são representadas por cores claras (por exemplo, amarelo) e as probabilidades grandes (próximas de 1) são representadas por cores escuras (por exemplo, vermelho escuro).

Uma característica importante da ferramenta TANGRAM-II e que será utilizada na solução dos modelos é a possibilidade de reordenar os estados e portanto, alterar a estrutura da matriz. Como vimos, cada estado do modelo é representado por um conjunto de variáveis de estado. O usuário então especifica a ordem em que as variáveis de estado devem aparecer na representação dos estados do modelo e a ferramenta organiza os estados em ordem crescente dos valores das variáveis. A estrutura da matriz do modelo de *jitter* mostrada na Figura 3.22 usa a seguinte lista de variáveis:  $n_1$ ,  $n_2$ ,  $f$  e  $s$ . Isto significa que o estado 1 é  $(0,0,0,0)$ , o estado 2 é  $(0,0, 1, 3)$ , o estado 3 é  $(0,0,2,3)$ , o estado 4 é  $(1,0,1,1)$ , e assim por diante. Vamos chamar a esta matriz de  $\mathbf{P}$ .

Precisamos agora definir os estados de  $S_0$  do novo algoritmo de aproximação. Como vimos, uma maneira é agrupar os estados da cadeia de acordo com uma ou mais variáveis de estado do modelo. Para este exemplo escolhemos a variável de estado  $n_2$ . Lembramos que  $n_2$  corresponde ao número de pacotes armazenados na fila antes do último período de silêncio. A Figura 3.21 mostra o agrupamento de estados usando  $n_2$ . O estado *clone*  $f_i$  representa os estados que tem armazenado  $i$  pacotes dos períodos ativos anteriores,  $i = 1, \dots, 50$ . Apenas dois tipos de eventos ocorrem nos estados *clones*: detecção de início de um período de silêncio e entrega de pacote. No primeiro caso, o valor de  $n_2$  é incrementado com o número de pacotes do período ativo atual e portanto,  $n_2$  passa a ser igual ao número total de pacotes na fila ( $n_1$ ). A transição de  $f_i$  para  $f_{i+j}$  significa que o período ativo



atual tinha  $j$  pacotes armazenados quando ocorreu um novo período de silêncio. Note que existem transições do estado  $f_i$  para os estados  $f_{i+1}, f_{i+2}, \dots, f_{50}$ ,  $i = 1, \dots, 49$ . No segundo caso, o valor de  $n_2$  é decrementado em 1 pois um pacote é entregue (os pacotes mais antigos são entregues primeiro). A taxa do primeiro evento corresponde à taxa de ocorrência de silêncio ( $\gamma$ ), enquanto a taxa do segundo evento corresponde à taxa de entrega de pacotes ( $1/T_2$ ). No exemplo escolhido para esta seção temos  $\gamma = 0.00167$  e  $1/T_2 = 0.05$ . Portanto, a transição de  $f_i$  para  $f_{i-1}$  corresponde à maior taxa de saída do estado  $f_i$ ,  $i = 1, \dots, 50$ .

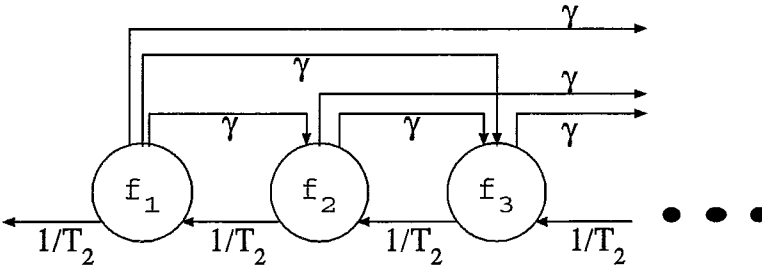


Figura 3.21: Estados clones do modelo de *jitter*.

Para usar a definição de  $S_0$  discutida acima precisamos reordenar os estados do modelo de acordo com o valor de  $n_2$ . A Figura 3.23 mostra a estrutura da matriz de transição que reordena os estados usando a seguinte lista de variáveis de estado:  $n_2, n_1, f, s$ . Vamos chamar a esta matriz reordenada de  $\mathbf{P}'$ . Os estados de  $S_0$  possuem  $n_2 = 0$  (são 201 estados neste exemplo). Note que existem probabilidades com valores pequenos ocupando algumas posições na parte superior da matriz. Estas transições correspondem à ocorrência de períodos de silêncio.

A Tabela 3.1 apresenta três soluções para o exemplo de *jitter* com desvio padrão 36 na saída da rede, visto no Capítulo 2, quando  $\varepsilon = 1.0e-04$  e  $T_1 = 800$ . A primeira solução corresponde à solução exata do modelo onde a matriz de probabilidades de transição é  $\mathbf{P}$  (veja Figura 3.22). A segunda solução corresponde à solução da matriz  $\mathbf{P}^- + \text{diag}(\mathbf{P}^\varepsilon e^T)$ , onde  $\mathbf{P}' = \mathbf{P}^- + \mathbf{P}^\varepsilon$ . E a terceira solução corresponde à solução encontrada pelo algoritmo proposto em 3.4 usando a matriz de probabilidades de transição  $\mathbf{P}'$  (veja Figura 3.23). Convém aqui observar que o motivo de usarmos a matriz  $\mathbf{P}$  no primeiro caso se deve ao fato que a solução

desta matriz tem um menor custo para o algoritmo do GTH, enquanto que o motivo de usarmos a matriz  $\mathbf{P}'$  nas duas outras soluções se deve ao fato do novo algoritmo exigir que os estados de  $S_0$  (grupo inicial) sejam os primeiros da matriz.

A Figura 3.26 mostra o número de operações requeridas para as três soluções quando o valor de  $T_1$  varia de 100 a 800. Podemos fazer dois comentários a respeito da Figura: 1) o número de operações requeridas pelos algoritmos praticamente se mantém constante quando variamos  $T_1$ ; 2) existe uma diferença de pelo menos uma ordem de magnitude entre o número de operações da solução exata e o número de operações das outras duas soluções (lembramos que usamos a matriz  $\mathbf{P}$  na solução exata e a matriz  $\mathbf{P}'$  nas outras soluções). É interessante notar em relação à segunda observação que os algoritmos de aproximação são mais baratos que o GTH, apesar da matriz  $\mathbf{P}$  possuir uma estrutura mais compacta que a matriz  $\mathbf{P}'$ . A explicação para isto é simples. As transições que estão distantes da diagonal na matriz  $\mathbf{P}'$  possuem valores pequenos e são, portanto, eliminadas (ou redirecionadas). Logo, a matriz usada pelos algoritmos de aproximação é bastante similar à matriz  $\mathbf{P}$ .

A Figura 3.27 mostra o número de operações requeridas para as três soluções quando o valor de  $B$  varia de 50 a 150. O modelo tem 2801 estados quando  $B = 50$  e 23401 estados quando  $B = 150$ . Por problemas de memória só foi possível usar o algoritmo GTH para  $B$  variando de 50 a 100. O número de operações do GTH para os outros valores de  $B$  foi calculado de acordo com a estrutura da matriz de transição. Para os outros dois algoritmos obtivemos as soluções do modelo facilmente.

Vejamos agora o modelo de *jitter* com as transições determinísticas. Neste exemplo, temos dois tipos de evento determinístico:  $\varphi_1$  que corresponde ao início da entrega dos pacotes ( $T_1$  unidades de tempo após a chegada do primeiro pacote) e  $\varphi_2$  que corresponde à entrega de um pacote ao destino. Portanto, os pontos embutidos correspondem ao início e ao término do tempo  $T_1$ , e aos instantes de início e/ou término de uma entrega de um pacote. Existem então três tipos

$x$	$\mathbf{P}$	$\mathbf{P}^- + \text{diag}(\mathbf{P}^\varepsilon e^T)$	Limite Inferior (Alg. Proposto)
0	2.5092073088e-01	2.6131190857e-01	2.4587567667e-01
10	2.0095640781e-01	2.0927845747e-01	1.9691549975e-01
20	1.6094900813e-01	1.6761426285e-01	1.5771258112e-01
30	1.2890653304e-01	1.3424483793e-01	1.2631442891e-01
40	1.0324322287e-01	1.0751875328e-01	1.0116716685e-01
50	8.2689083462e-02	8.6113421453e-02	8.1026338296e-02
60	6.6226957407e-02	6.8969562317e-02	6.4895239257e-02
70	5.3042187721e-02	5.5238782131e-02	5.1975594193e-02
80	4.2482303103e-02	4.4241589316e-02	4.1628051960e-02
90	3.4024729268e-02	3.5433768626e-02	3.3340546402e-02
100	2.7250928438e-02	2.8379449709e-02	2.6702955869e-02

Tabela 3.1:  $\Pr[\text{Jitter} > x]$  para  $T_1 = 800$  e  $B = 50$ .

de intervalos:  $\Delta_0$  igual ao intervalo onde não existem eventos determinísticos habilitados (antes da chegada da primeira pacote e durante o período em que a fila fica vazia após o início da entrega dos pacotes),  $\Delta_1$  igual ao intervalo onde  $\varphi_1$  está ativo (tempo de espera após chegada do primeiro pacote), e  $\Delta_2$  igual ao intervalo onde  $\varphi_2$  está ativo (entrega de pacotes). Note que o evento  $\varphi_1$  é desabilitado se a fila enche antes do período de espera  $T_1$  acabar, enquanto que o evento  $\varphi_2$  nunca é desabilitado antes do tempo  $T_2$ , uma vez ativado.

A Figura 3.24 mostra a estrutura da matriz  $\mathcal{H}$  gerada a partir da matriz  $\mathbf{P}$  discutida acima. A matriz  $\mathcal{H}$  possui 2701 estados. Observe que a matriz  $\mathcal{H}$  possui uma estrutura muito diferente da matriz  $\mathbf{P}$  apresentada na Figura 3.22. A Figura 3.25 mostra a matriz  $\mathcal{H}'$  gerada a partir da reordenação dos estados de  $\mathcal{H}$  usando o mesmo critério de ordenação de  $\mathbf{P}'$ :  $n_2, n_1, f, s$ . Novamente, os estados de  $S_0$  correspondem aos estados onde  $n_2 = 0$ . Embora as estruturas de  $\mathcal{H}$  e  $\mathcal{H}'$  sejam diferentes, o GTH preenche a parte superior das duas matrizes.

Quando estudamos uma solução para modelos não markovianos em 3.2.2, vimos que a solução da Matriz  $\mathcal{H}$  nos fornece o vetor  $\beta$  das probabilidades estacionárias dos pontos embutidos e que precisamos usar a equação 3.1 para calcular o vetor  $\pi$  das probabilidades estacionárias dos estados do modelo. Para isto é preciso calcular a duração dos intervalos entre dois pontos embutidos e o tempo gasto em cada estado durante cada intervalo. O vetor  $\beta$  fornece a probabilidade de cada estado no início do intervalo. A probabilidade de um estado corresponde então ao percentual de tempo gasto no estado em relação ao tempo gasto nos intervalos dos pontos embutidos. Note que  $\pi$  é um vetor normalizado. Como o vetor  $\beta$  é uma aproximação para a solução de  $\mathcal{H}$ , o vetor  $\pi$  não é a solução exata da matriz  $\mathbf{P}$ , mas uma aproximação para a solução de  $\mathbf{P}$  onde, infelizmente, não conhecemos o erro.

A Tabela 3.2 apresenta três soluções para a matriz  $\mathcal{H}$  quando  $\varepsilon = 1.0e - 04$  e  $T_1 = 800$ . Note que a aproximação da medida de interesse obtida usando o algoritmo proposto é melhor do que a aproximação obtida pela solução de  $\mathcal{H}^{-1} + \text{diag}(\mathcal{H}^\varepsilon e^T)$ .

A Figura 3.28 mostra o número de operações requeridas para as três soluções quando o valor de  $T_1$  varia de 100 a 800. Novamente temos uma diferença de pelo menos 1 ordem de magnitude entre a solução do algoritmo GTH e os outros dois algoritmos. Observando as Figuras 3.26 e 3.28 podemos notar que o número de operações requeridas na solução da matriz exponencial ( $\mathbf{P}$  e  $\mathbf{P}'$ ) é menor que o número de operações requeridas na solução da matriz de pontos embutidos ( $\mathcal{H}$  e  $\mathcal{H}'$ ), apesar da primeira matriz ter um maior número de estados. Isto se deve à estrutura das matrizes  $\mathcal{H}$  e  $\mathcal{H}'$  que são parcialmente cheias.

### 3.5.2 Modelo 2: Sistema de Polling

O segundo exemplo é um modelo de um multiplexador de dados e voz  $T_1 - T_2$  [41], conforme indicado na Figura 3.29. O objetivo deste mecanismo é garantir uma

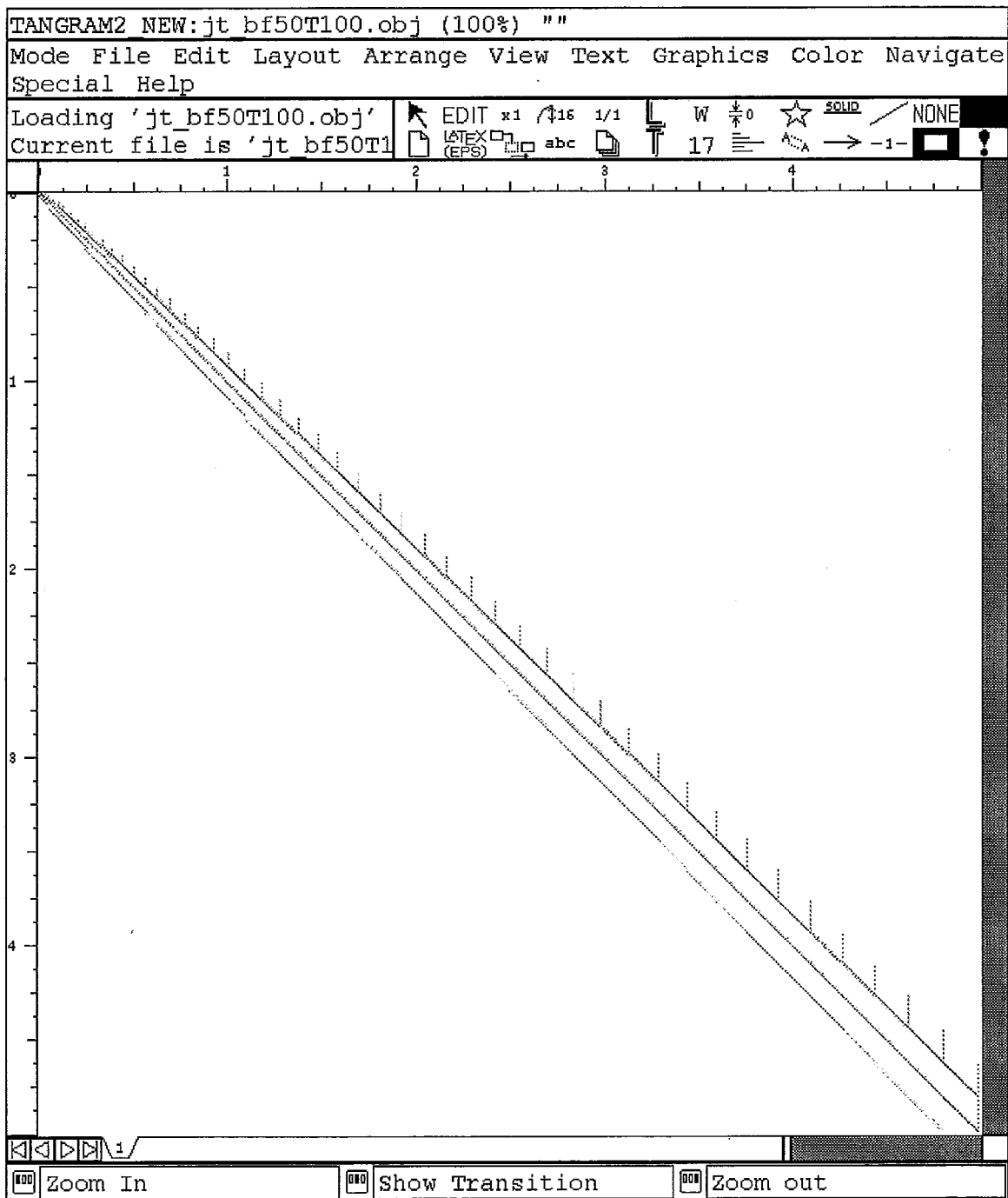


Figura 3.22: Estrutura da Matriz  $P$  do modelo de *jitter*.

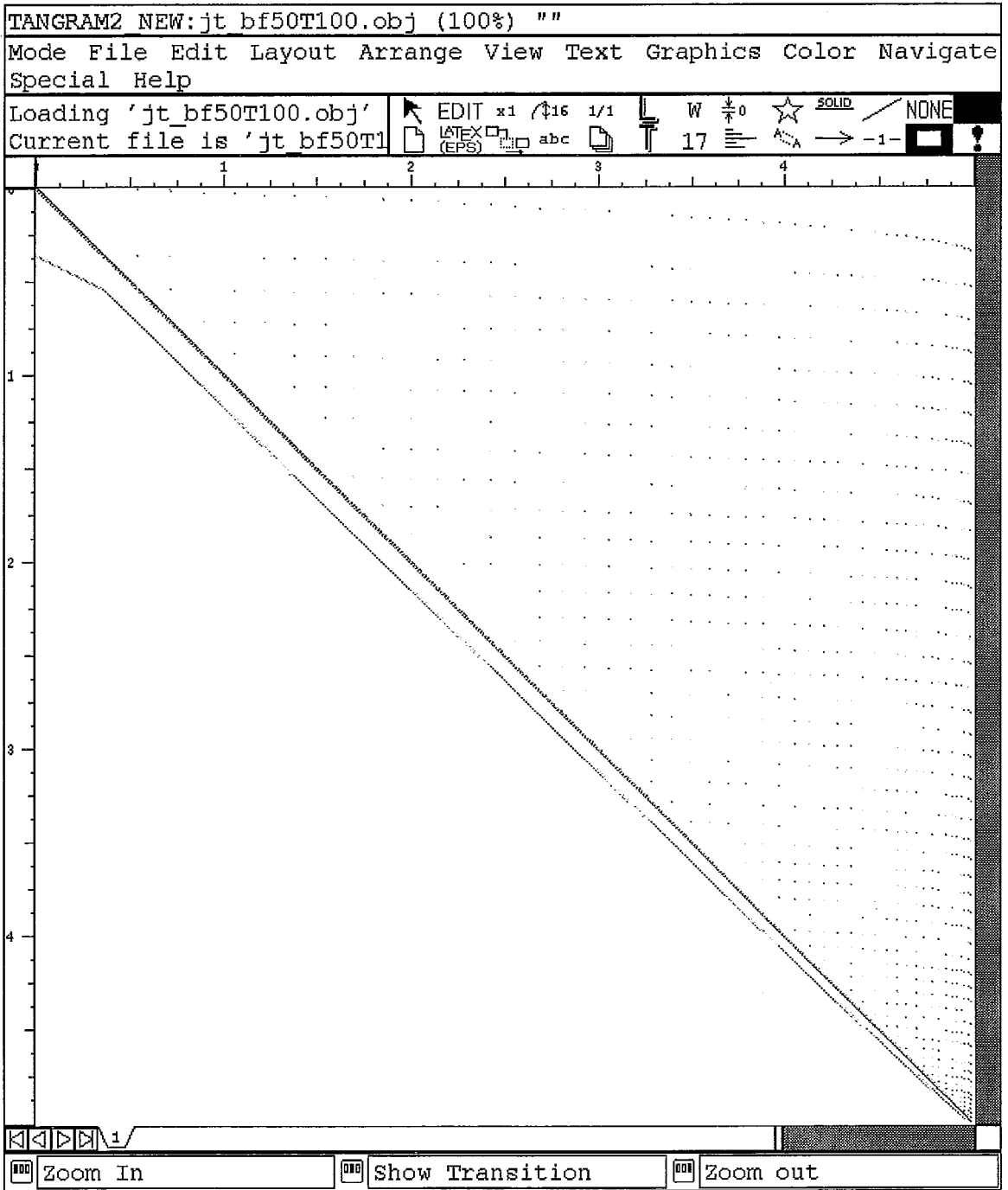


Figura 3.23: Estrutura da Matriz  $P'$  do modelo de *jitter*.

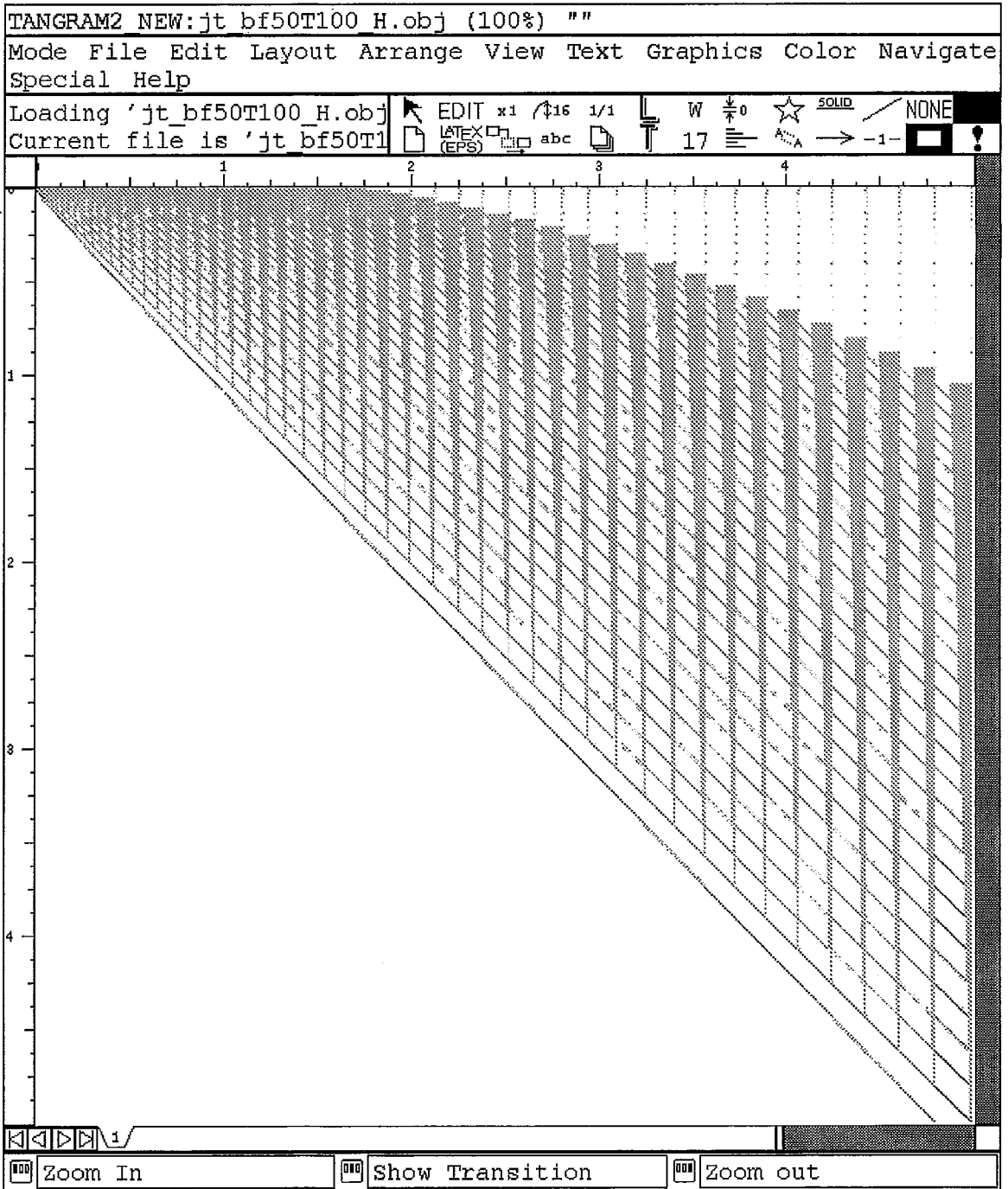


Figura 3.24: Estrutura da Matriz  $\mathcal{H}$  do modelo de *jitter*.

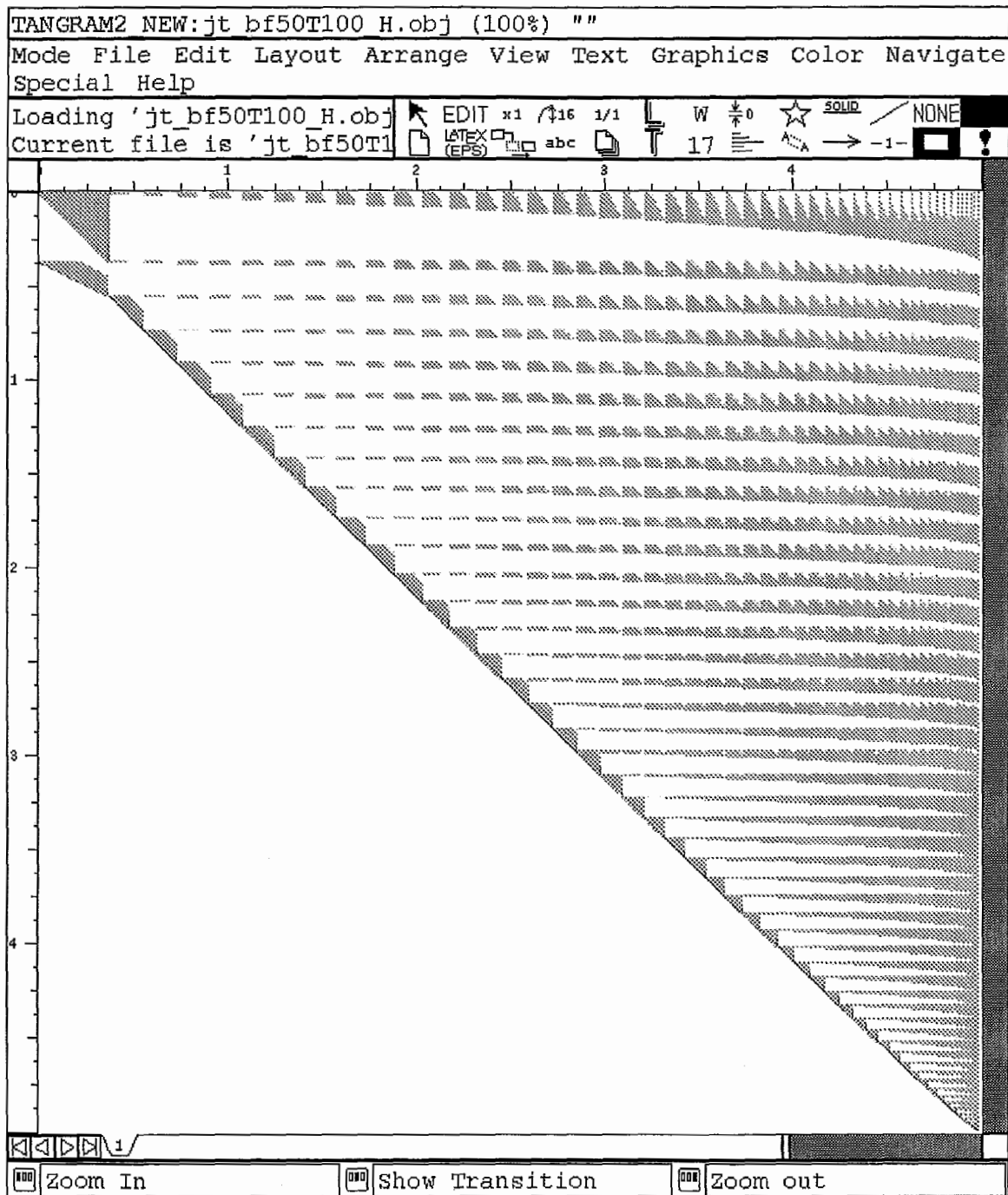


Figura 3.25: Estrutura da Matriz  $\mathcal{H}'$  do modelo de *jitter*.



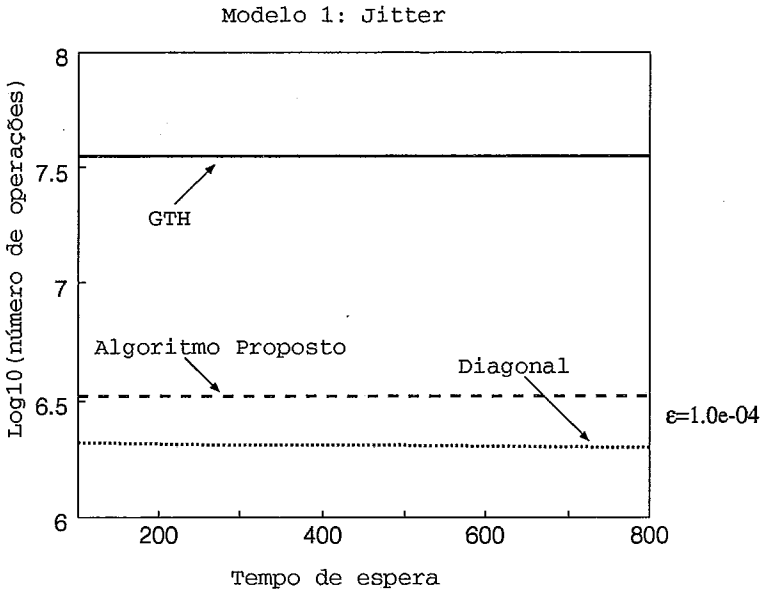


Figura 3.26: Número de operações requeridas na solução da matriz exponencial para  $B = 50$ .

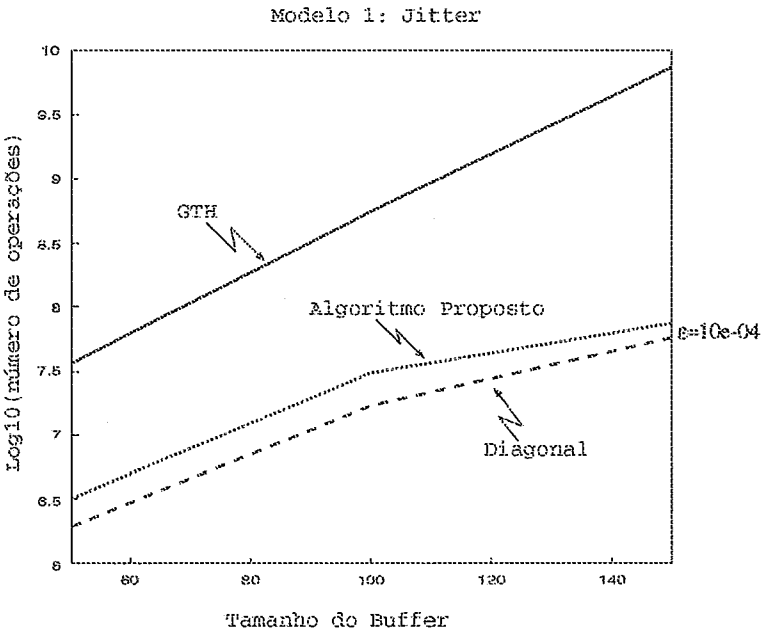


Figura 3.27: Número de operações requeridas na solução da matriz exponencial para diferentes valores de  $B$ .

$x$	$\mathcal{H}$	$\mathcal{H}^- + \text{diag}(\mathcal{H}^\varepsilon e^T)$	Aproximação Alg. Proposto
0	2.5961166006e-01	2.6173145370e-01	2.5871515108e-01
10	2.0792696730e-01	2.0962474260e-01	2.0720893513e-01
20	1.6653190277e-01	1.6789167709e-01	1.6595681975e-01
30	1.3337795958e-01	1.3446702373e-01	1.3291736675e-01
40	1.0682445709e-01	1.0769670530e-01	1.0645556122e-01
50	8.5557348972e-02	8.6255945961e-02	8.5261894598e-02
60	6.8524195323e-02	6.9083712386e-02	6.8287561376e-02
70	5.4882080863e-02	5.5330206675e-02	5.4692557099e-02
80	4.4314812638e-02	4.3955901790e-02	4.3804109295e-02
90	3.5204957097e-02	3.5492414309e-02	3.5083384155e-02
100	2.8196191039e-02	2.8426419938e-02	2.8098821403e-02

Tabela 3.2:  $\Pr[\text{Jitter} > x]$  para  $T_1 = 800$  e  $B = 50$ .

fração mínima de uso do canal para cada um dos tráfegos (dados e voz). Evitando assim, que os pacotes de um determinado tipo monopolizem o canal e aumentem o retardo fim-a-fim dos pacotes do outro tráfego. Naturalmente, os valores  $T_1$  e  $T_2$  dependem da QoS definida pela rede para estes dois tipos de tráfego.

O comportamento deste multiplexador é descrito a seguir. Pacotes de dados são servidos até que a fila reservada a esses pacotes se esvazie ou um temporizador ativado no início do serviço da fila de dados seja atingido. O limite deste temporizador é  $T_1$ . Após o período de serviço da fila de pacotes de dados, o canal multiplexado comuta para a outra fila e é alocado aos pacotes de voz. Os pacotes de voz são então servidos até que a fila se esgote ou até que o limite de um temporizador de valor  $T_2$  seja atingido. Uma vez que todos os pacotes de voz são servidos ou o temporizador  $T_2$  dispara (o que ocorrer primeiro), o canal comuta para a fila de dados. O tempo de comutação do canal é um parâmetro do modelo, e neste trabalho é suposto constante.

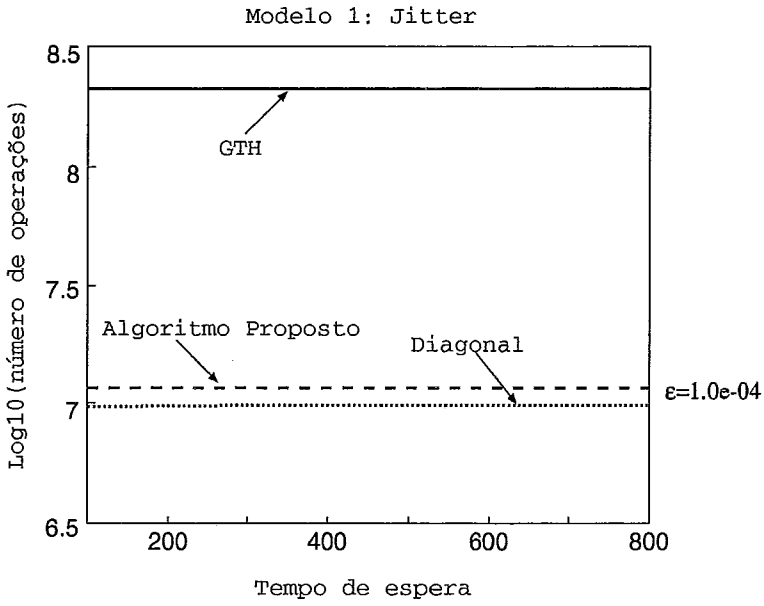


Figura 3.28: Número de operações requeridas na solução da matriz de pontos embutidos.

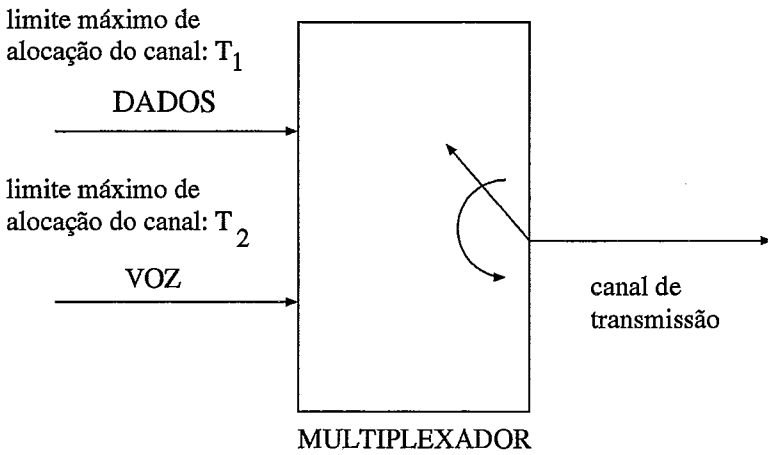


Figura 3.29: Multiplexador  $T_1 - T_2$ .

Note que este multiplexador limita o tempo em que o canal de transmissão fica alocado a cada um dos dois tráfegos. Em consequência é garantido a cada um dos tráfegos uma fração do canal. Entretanto, a capacidade do canal é dinamicamente alocada a um dos tráfegos, caso o outro não esteja usando a fração máxima a ele alocada.

Para este modelo, supomos que o processo de chegada de pacotes de voz e dados tem distribuição Poisson, com taxas diferentes. O tamanho dos pacotes de voz e de dados tem distribuição exponencial, com médias diferentes. Seja  $B$  o número máximo de pacotes que cada fila pode armazenar. Um pacote é descartado quando chega ao multiplexador e já existem outros  $B$  pacotes na fila. Os limites  $T_1$  e  $T_2$  e os tempos de comutação são constantes, e portanto o modelo não é markoviano.

Existem várias políticas para servir a fila dedicada a cada um dos tráfegos sendo multiplexado. Por exemplo, cada fila pode ser servida de modo exaustivo ou *gated* [42]. Quando o serviço é exaustivo, o servidor serve a fila enquanto esta tiver pacotes ou até acabar o tempo alocado para a fila. O serviço *gated* é semelhante ao serviço exaustivo, a diferença consiste em que o servidor só serve os pacotes que já estavam na fila quando o serviço foi iniciado, ou seja, os pacotes que chegam depois do início do serviço aguardam pelo menos até a próxima visita do servidor para serem transmitidos. Os detalhes da solução destes modelos e de outros mais complexos podem ser encontrados em [43, 44].

O modelo é definido usando 5 objetos: *fonte de pacotes de dados*, *fonte de pacotes de voz*, *fila de pacotes de dados*, *fila de pacotes de voz* e *controle*. Quando um pacote é gerado pela *fonte de pacotes de dados* (*fonte de pacotes de voz*), ele é enviado para a *fila de pacotes de dados* (*fila de pacotes de voz*). Se já existem  $B$  pacotes na fila, o pacote é descartado. O *controle* é responsável por definir quem pode utilizar o canal de transmissão e por quanto tempo cada fila pode utilizar o canal.

Os objetos *fila de pacotes de dados* e *fila de pacotes de voz* possuem sempre o mesmo valor. O objeto *fila de pacotes de dados* é caracterizado pelas variáveis  $n_d$  e  $g_d$  para o modelo *gated*. A variável  $n_d$  representa o número total de pacotes na fila e a variável  $g_d$  representa o número de pacotes que já estavam na fila quando o serviço foi iniciado. Naturalmente,  $n_d$  e  $g_d$  podem variar de zero a  $B$ . Similarmente, o objeto *fila de pacotes de voz* é caracterizado pelas variáveis  $n_v$  e  $g_v$ . O objeto *controle* é caracterizado pela variável de estado  $c$ . Esta variável possui valores de 1 a 4 onde cada valor representa como o canal está sendo utilizado naquele determinado momento. O valores 1 e 3 significam, respectivamente, o uso do canal pela *fila de pacotes de dados* e pela *fila de pacotes de voz*. Os valores 2 e 4 representam a comutação da fila de pacotes de dados para a fila de pacotes de voz e a comutação da fila de pacotes de voz para a fila de pacotes de dados, respectivamente.

Os parâmetros para este exemplo são os mesmos utilizados em [44] e são dados a seguir. O tamanho médio dos pacotes de dados e voz é de 400 bits e 600 bits, respectivamente. A capacidade do canal é de 1.5 Mbps e o intervalo de comutação do multiplexador é de 0.1 mseg. O valor dos temporizadores para dados e voz é de 2 e 8 mseg, respectivamente. Entre os vários exemplos apresentados em [44] escolhemos o exemplo em que a carga média dos pacotes de dados corresponde a 10% da capacidade do canal e a política de serviço das filas é *gated*. A carga média dos pacotes de voz nos exemplos é sempre igual a 60% da capacidade do canal. Este exemplo foi escolhido por apresentar uma maior diferença entre os valores das taxas do modelo o que facilita o aparecimento de transições  $\varepsilon$ . Similarmente ao que fizemos com o modelo de *jitter*, vamos inicialmente estudar o modelo do multiplexador  $T_1 - T_2$  assumindo que todas as transições são exponenciais. E em seguida, faremos o estudo do modelo com as transições determinísticas. A medida de interesse desejada é o retardo médio de um pacote no multiplexador  $T_1 - T_2$ .

A Figura 3.32 mostra a estrutura da matriz que representa o modelo onde

todas as transições são exponenciais,  $B = 25$  e o estado do modelo é caracterizado por  $n_d, g_d, n_v, g_v$  e  $c$ . O número de estados da matriz cresce à medida que aumentamos o valor de  $B$  (tamanho de cada fila). Por exemplo, temos 1452 estados para  $B = 10$  e 30.752 estados para  $B = 30$ . A Figura 3.33 mostra a matriz de probabilidades reordenada com a seguinte lista de variáveis de estado:  $n_d, n_v, g_d, g_v$  e  $c$ . Comparando as Figuras 3.32 e 3.33 podemos notar que os elementos da segunda matriz estão mais concentrados na diagonal e portanto, esta matriz tem um custo menor no algoritmo GTH. Vamos então chamar a esta segunda matriz de  $\mathbf{P}$  e usá-la na solução dos três algoritmos.

Para usarmos o novo algoritmo precisamos definir os estados de  $S_0$  na matriz  $\mathbf{P}$ . Podemos, por exemplo, definir os estados de  $S_0$  como aqueles estados que possuem  $n_d = 0$ . A Figura 3.30 mostra os estados clones do novo modelo, onde  $\lambda$  e  $\mu$  representam, respectivamente, a maior taxa de chegada e a menor taxa de serviço do modelo. Iremos usar este agrupamento para solucionar o modelo.

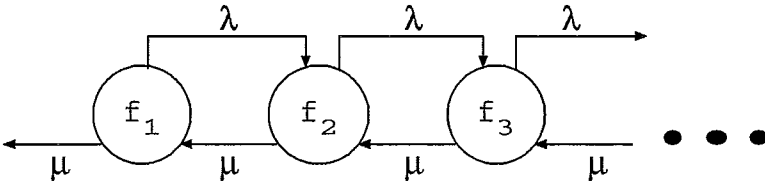


Figura 3.30: Estados clones do modelo de *polling*.

As Tabelas 3.3 e 3.4 mostram, respectivamente, o retardo médio dos pacotes de dados e o retardo médio dos pacotes de voz quando  $B$  varia de 10 a 30. Assumimos que  $\varepsilon = 10e - 04$ . Podemos fazer algumas observações em relação a estes resultados. Em primeiro lugar, os valores obtidos na solução de  $\mathbf{P}^- + \text{diag}(\mathbf{P}^\varepsilon e^T)$  são menores que os valores obtidos na solução de  $\mathbf{P}$  quando  $B = 10$  e são maiores que os valores da solução de  $\mathbf{P}$  quando  $15 \leq B \leq 30$ . Isto ilustra o que foi comentado no início deste capítulo sobre a impossibilidade de prever se a solução obtida é maior ou menor que a solução exata do modelo. Em segundo lugar, a solução de  $\mathbf{P}^- + \text{diag}(\mathbf{P}^\varepsilon e^T)$  pode não ser uma boa aproximação para este modelo. Por exemplo, o retardo médio de um pacote de voz para  $B = 30$  possui

quase o dobro do valor calculado na solução da matriz  $\mathbf{P}$  (respectivamente, 0.995 e 0.577). Por último, é importante enfatizar que o valor máximo de 30 definido para  $B$  nestes exemplos, se deve ao fato de que não foi possível executar o algoritmo GTH para  $B$  maior que 30. Entretanto, não tivemos problemas com os outros dois algoritmos para  $B = 10, \dots, 100$ . Como o objetivo é comparar as soluções dos três algoritmos, mostramos aqui apenas as soluções do modelo quando  $B$  varia de 10 a 30.

Tam. do Buffer	$\mathbf{P}$	$\mathbf{P}^- + \text{diag}(\mathbf{P}^\varepsilon e^T)$	Limite Inferior Alg. Proposto
10	6.5156002274	6.4640521573	6.4189730994
15	7.0157544826	7.0493442013	6.7494928734
20	7.1434242845	7.4766293169	6.7893996517
25	7.1719634654	7.7188956812	6.7938905115
30	7.1778739878	8.6174844658	6.7942799104

Tabela 3.3: Retardo médio de um pacote de dados.

Tam. do Buffer	$\mathbf{P}$	$\mathbf{P}^- + \text{diag}(\mathbf{P}^\varepsilon e^T)$	Limite Inferior Alg. Proposto
10	0.5340364619	0.5278693040	0.5190574229
15	0.5672516184	0.6031076469	0.5369935383
20	0.5752625547	0.6955525990	0.5383218542
25	0.5769817501	0.7390575300	0.5384539511
30	0.5773262580	0.9952789714	0.5384644558

Tabela 3.4: Retardo médio de um pacote de voz.

A Figura 3.36 mostra o número de operações requeridas nas três soluções quando o valor de  $B$  varia de 10 a 30. Observe que o número de operações da solução de  $\mathbf{P}$  cresce mais rapidamente que as outras duas soluções quando

incrementamos o valor de  $B$  (a diferença é de uma ordem de magnitude quando  $B = 10$  e é de duas ordens de magnitude quando  $B = 30$ ).

Vejam agora o modelo do multiplexador  $T_1-T_2$  com transições não exponenciais. Neste exemplo, temos quatro eventos determinísticos:  $\varphi_1$  que corresponde ao evento de disparo do temporizador de dados  $T_1$ ,  $\varphi_2$  que corresponde ao evento de disparo do temporizador de voz  $T_2$ , e  $\varphi_3$  e  $\varphi_4$  que correspondem aos eventos de término de comutação entre voz e dados e dados e voz, respectivamente. Os pontos embutidos equivalem aos instantes onde o canal multiplexado inicia e termina o serviço de uma das filas de entrada. Existem então quatro tipos de intervalos:  $\Delta_i$ ,  $i = 1, \dots, 4$ , onde  $\Delta_i$  corresponde ao intervalo onde  $\varphi_i$  está ativo. Note que não existem intervalos onde apenas eventos exponenciais estão ativos.

A Figura 3.34 mostra a estrutura da matriz  $\mathcal{H}$  com 2652 estados quando  $B = 25$ . Esta matriz possui 462 estados quando  $B = 10$  e 3782 estados quando  $B = 30$ . Novamente, a estrutura da matriz  $\mathcal{H}$  é bem diferente da estrutura da matriz  $\mathcal{P}$ . A Figura 3.35 mostra a estrutura da matriz reordenada com a seguinte lista de variáveis de estado  $c$ ,  $n_d$ ,  $g_d$ ,  $n_v$  e  $g_v$ . Vamos chamar a esta nova matriz de  $\mathcal{H}'$ . As soluções apresentadas a seguir para os algoritmos serão todas calculadas usando  $\mathcal{H}'$  por ser esta uma matriz esparsa.

Os estados da partição  $S_0$  usado no algoritmo proposto correspondem na Figura 3.35 aos estados que possuem transição para os últimos estados do modelo (bloco inferior à esquerda com 121 estados quando  $B = 10$  e 958 estados quando  $B = 30$ ). Estes estados possuem como característica comum a variável de estado  $c$  com valor 1, ou seja, todos os estados que representam a alocação do canal para a fila de pacotes de dados. A Figura 3.31 mostra os estados clones gerados a partir desse agrupamento. Temos apenas 3 estados: os estados  $f_1$  e  $f_3$  representam, respectivamente, a comutação da fila de voz para a fila de dados e a comutação da fila de dados para a fila de voz; e o estado  $f_2$  representa os estados onde o canal está alocado para os pacotes de voz. Note que existe uma transição de  $f_1$  para  $f_3$ . Esta transição corresponde à seguinte situação: foi feita a comutação para a fila



de pacotes de dados, só que esta fila está vazia e portanto, é iniciada de imediato a comutação para a fila de pacotes de voz. A situação contrária (fila de pacotes de voz vazia) também existe, mas não é representada nos estados *clones* pois não existe transição, segundo a definição do algoritmo, do estado  $f_3$  para o estado  $f_1$ .

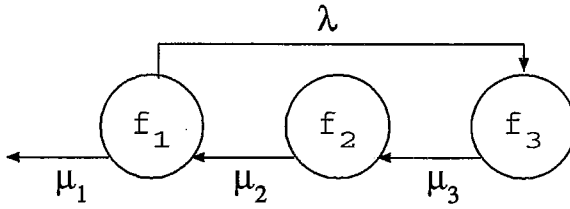


Figura 3.31: Estados clones do modelo de *polling*.

As Tabelas 3.5 e 3.6 mostram o retardo médio dos pacotes quando  $B$  varia de 10 a 30 e  $\varepsilon = 1.0e - 04$ . Tanto a solução de  $\mathcal{H}^- + \text{diag}(\mathcal{H}^\varepsilon e^T)$  quanto a solução do algoritmo proposto podem ser consideradas boas aproximações para a solução da matriz  $\mathbf{P}$ , sendo que a segunda solução é uma aproximação melhor que a primeira.

Tam. do Buffer	$\mathcal{H}$	$\mathcal{H}^- + \text{diag}(\mathcal{H}^\varepsilon e^T)$	Aproximação Alg. Proposto
10	1.2164281377	1.1621512948	1.1828155051
15	1.2716405857	1.1838712744	1.2195546321
20	1.2835511747	1.1803411264	1.2236903011
25	1.2856808486	1.1667880034	1.2237582555
30	1.2860365698	1.1628050758	1.2236980653

Tabela 3.5: Retardo médio de um pacote de dados.

A Figura 3.37 mostra o número de operações requeridas pelas três soluções. Os mesmos comentários feitos para a matriz  $\mathbf{P}$  podem aqui serem repetidos para a matriz  $\mathcal{H}'$  (lembramos que esta matriz foi usada pelos três algoritmos). Entretanto, é interessante observar que o número de operações requeridas na solução da matriz  $\mathcal{H}'$  é só ligeiramente inferior ao número de operações requeridas na solução da matriz  $\mathbf{P}$ , apesar da matriz  $\mathbf{P}$  possuir de 3 ( $B = 10$ ) a 8 ( $B = 30$ )

Tam. do Buffer	$\mathcal{H}$	$\mathcal{H}^- + \text{diag}(\mathcal{H}^s e^T)$	Aproximação Alg. Proposto
10	1.7112352226	1.6470255333	1.6769106644
15	1.8279547058	1.7113883950	1.7624126662
20	1.8552616197	1.7125063435	1.7742562995
25	1.8610265552	1.6949020046	1.7753226679
30	1.8622094739	1.6891209550	1.7752823517

Tabela 3.6: Retardo médio de um pacote de voz.

vezes mais estados que a matriz  $\mathcal{H}'$ . Mais uma vez, a explicação para esse fato se encontra nas diferentes estruturas das matrizes  $\mathbf{P}$  e  $\mathcal{H}'$ .

### 3.5.3 Modelo 3: Servidores de Vídeo com Carga Mista

Na seção 3.5.2 estudamos um mecanismo que garante a cada um dos tráfegos (dados e voz) uma fração de uso do canal de transmissão. Nesta seção, iremos estudar um modelo de um servidor com serviço *G-Gated*. Temos um único recurso, neste caso um disco, e vários pedidos de acesso às informações armazenadas no disco. Imagine que o disco armazena arquivos com características bem distintas como, por exemplo, filmes e dados. Precisamos, portanto, definir como estes pedidos vão ser atendidos. A seguir vamos discutir o modelo de um servidor de disco com política de serviço *g-gated* que foi apresentado em [45].

O servidor de disco [45] atende a duas classes de pedidos:  $C$  e  $NC$ . Quando um pedido chega ao servidor, ele é armazenado na fila correspondente ao seu tipo e aguarda a sua vez. O tempo é dividido em ciclos de tamanho  $T$ . Em cada período  $T$  o servidor deve atender exatamente  $N_c$  pedidos  $C$ . O valor de  $T$  e  $N_c$  dependem do nível da QoS que o sistema fornece aos seus usuários. Assume-se que em cada ciclo chegam  $N_c$  pedidos  $C$  para serem servidos no próximo ciclo.

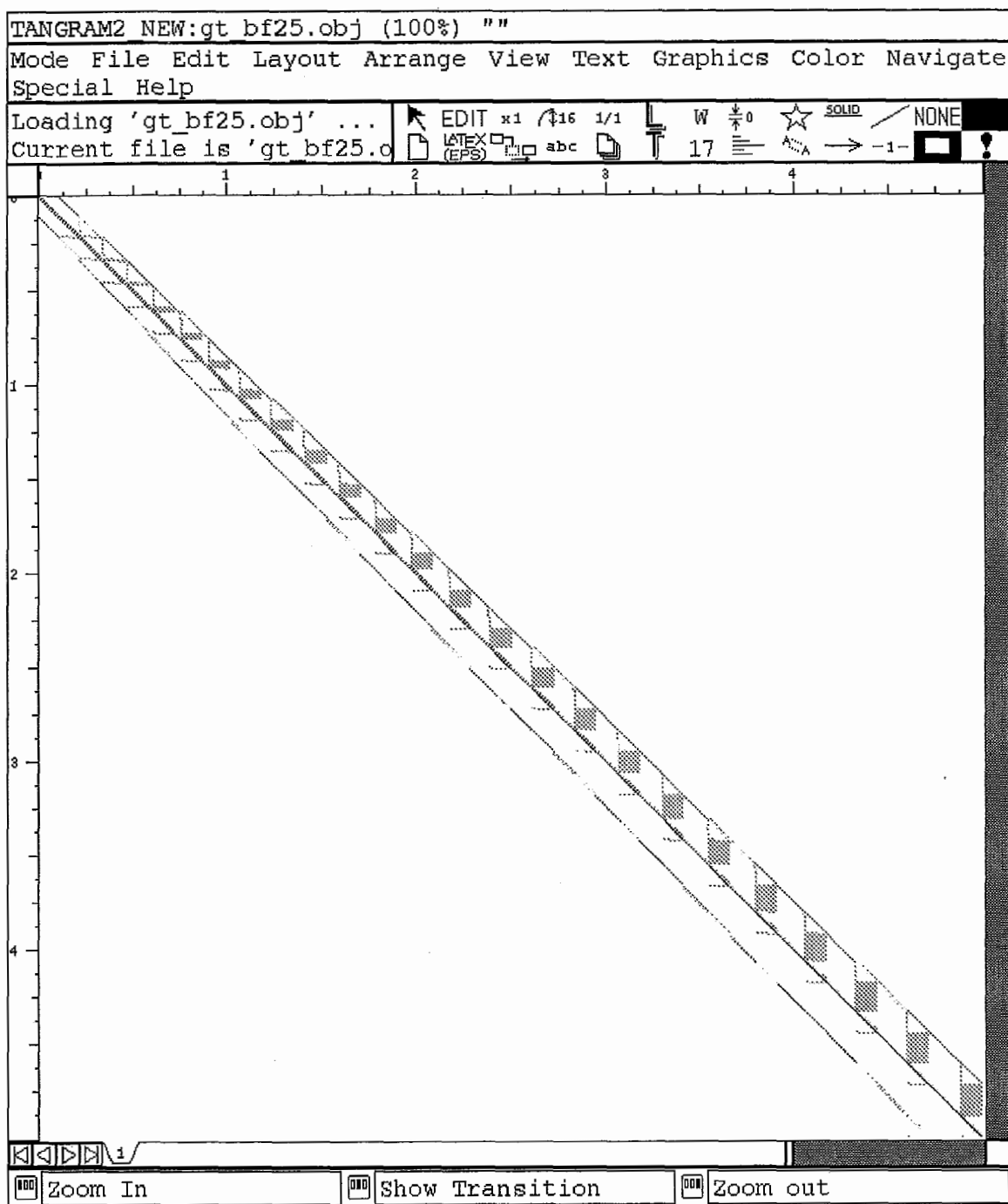


Figura 3.32: Estrutura da Matriz do modelo do Multiplexador  $T_1 - T_2$ .

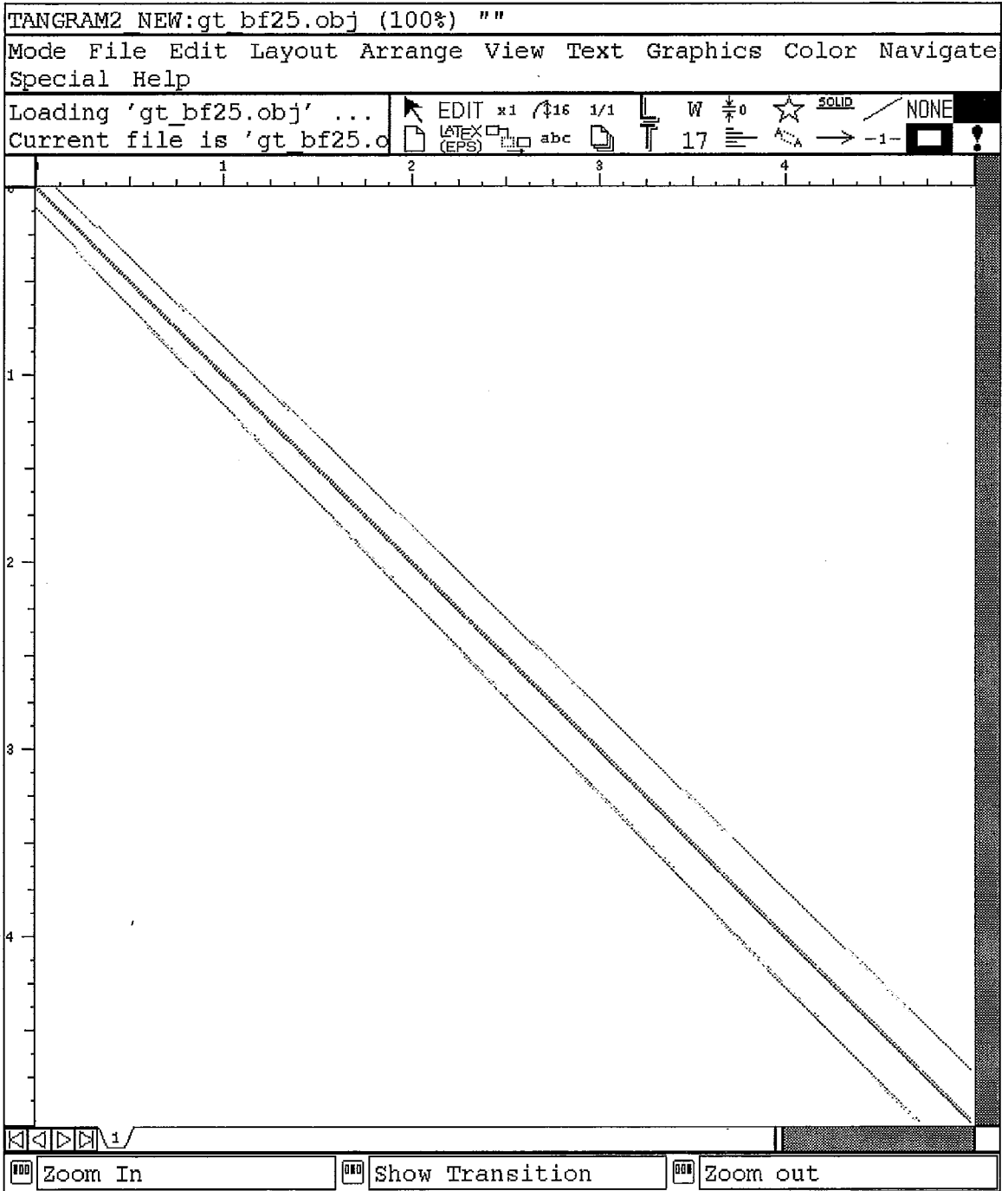


Figura 3.33: Estrutura da Matriz  $P$  do modelo do Multiplexador  $T_1 - T_2$ .

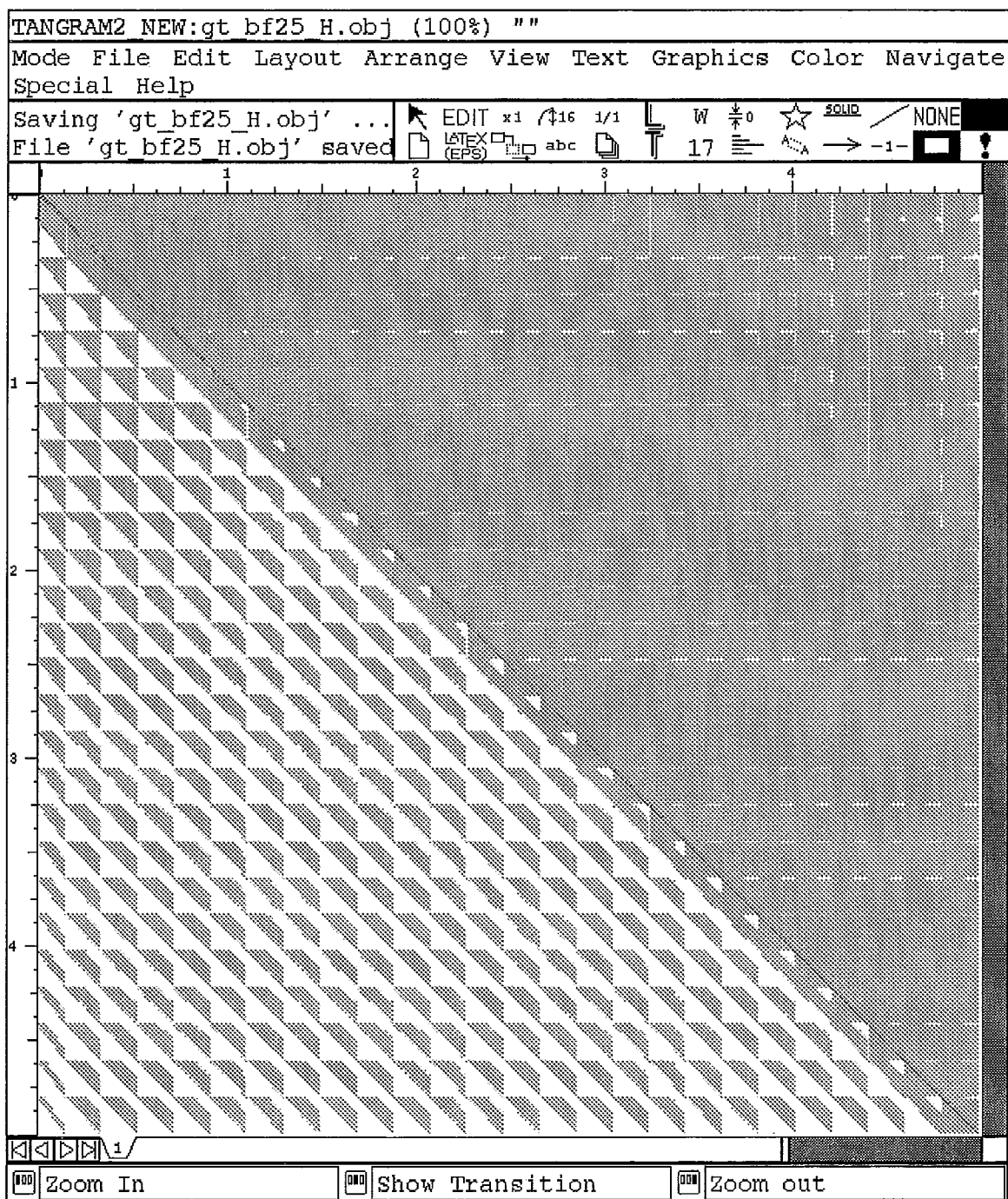


Figura 3.34: Estrutura da Matriz  $\mathcal{H}$  do modelo do Multiplexador  $T_1 - T_2$ .

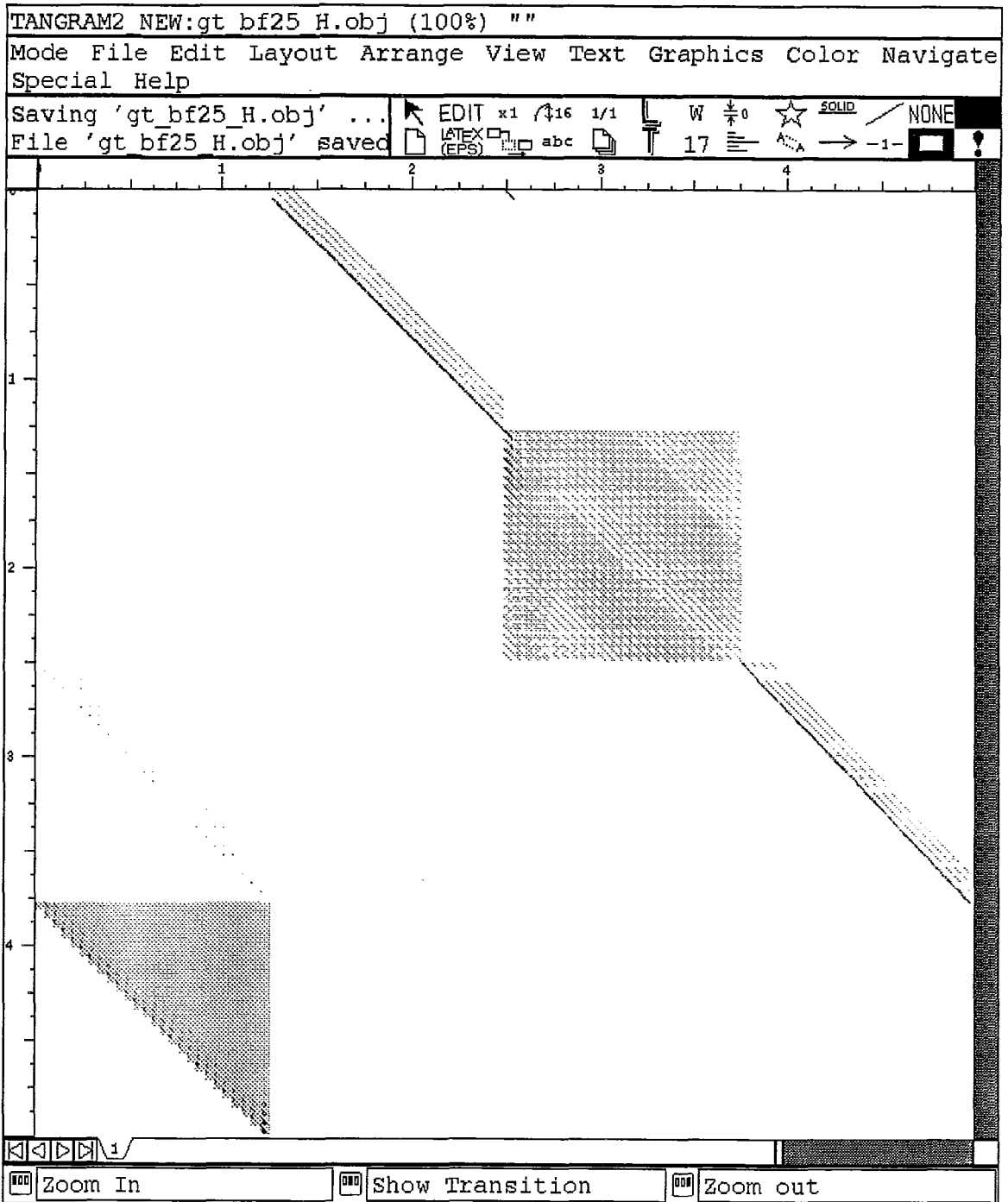


Figura 3.35: Estrutura da Matriz  $\mathcal{H}'$  do modelo do Multiplexador  $T_1 - T_2$ .

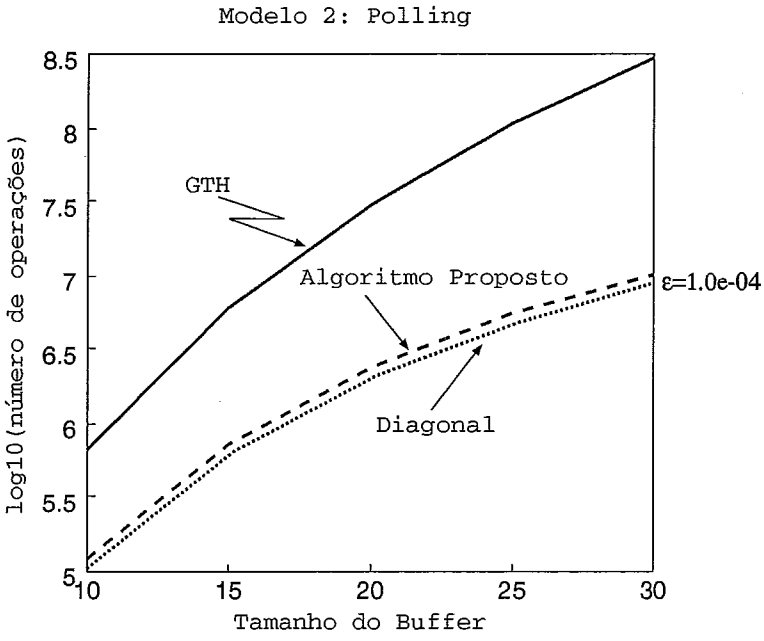


Figura 3.36: Número de operações requeridas na solução da matriz exponencial.

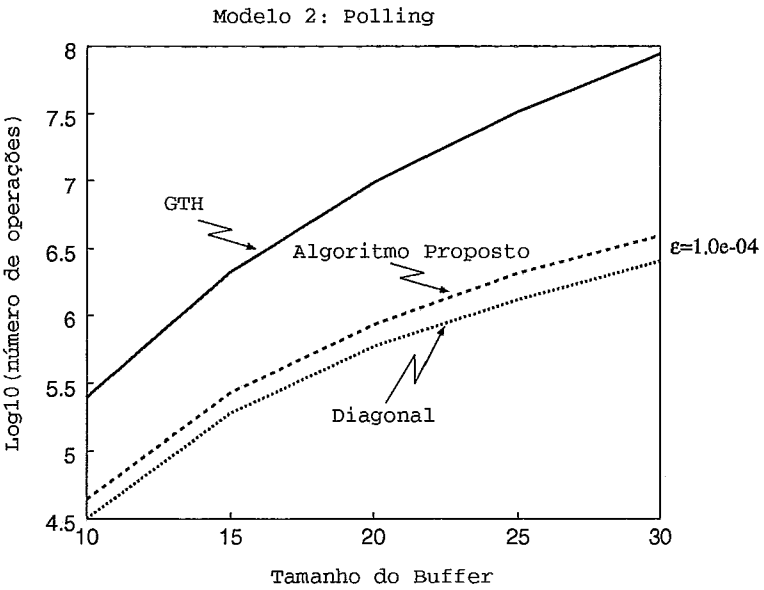


Figura 3.37: Número de operações requeridas na solução da matriz de pontos embutidos.

O servidor utiliza o algoritmo *g-gated* para escalonar os pedidos a serem atendidos. A descrição do algoritmo é dada a seguir. Cada ciclo  $T$  é dividido em  $N_{mc}$  mini-ciclos de tamanho  $T/N_{mc}$  cada um. No primeiro mini-ciclo, o sistema serve  $N_c/N_{mc}$  pedidos da classe  $C$ . Ao fim do serviço, o servidor verifica se existe algum pedido  $NC$  na fila. Se existem pedidos  $NC$  armazenados, os pedidos  $NC$  que já estão na fila são servidos até o fim do mini-ciclo ou até que todos os pedidos  $NC$  (os que já estavam na fila) sejam atendidos. Caso não existam pedidos  $NC$  para serem servidos no mini-ciclo o servidor volta a atender pedidos  $C$ . Esta política de atendimento é mantida nos primeiros  $N_{mc}$  mini-ciclos. No último mini-ciclo o servidor só atende pedidos  $NC$  após servir os últimos  $N_c/N_{mc}$  pedidos  $C$  (se ainda existirem pedidos  $C$  a serem atendidos no ciclo). Note que o servidor nunca fica ocioso nos primeiros  $N_{mc} - 1$  mini-ciclos, por isso o algoritmo é dito guloso (*greedy*).

Os parâmetros para o exemplo são os mesmos utilizados em [45] para um disco com 2.25 GBytes e 5288 cilindros, uma taxa de transferência de 75 Mbps e tempo de latência máximo de 8.33 segundos:  $T = 1.48754$  segundos;  $N_c = 24$ ;  $N_{mc} = 6$ ; o tempo de serviço de 24 pedidos  $C$  tem uma distribuição erlangiana com 6 estágios (o número de estágios depende da QoS escolhida, veja [45] para mais detalhes); e o tempo de serviço de um pedido  $NC$  é exponencialmente distribuído com média de 0.018407 segundos.

A Figura 3.38 mostra a estrutura da matriz de transição do modelo assumindo que todas as transições são exponenciais. O limite  $T$  é constante, e portanto o modelo não é markoviano. Os pontos embutidos deste exemplo correspondem ao início e ao fim de cada mini-ciclo. A Figura 3.39 mostra a estrutura da matriz  $\mathcal{H}$  dos pontos embutidos. Para este exemplo a matriz de transição do modelo tem 4941 estados e a matriz  $\mathcal{H}$  tem 3996 estados. A Figura 3.40 mostra a matriz  $\mathcal{H}'$  obtida pela ordenação dos estados conforme definido em [45]. Os estados do grupo inicial usado no algoritmo proposto correspondem na Figura 3.40 aos estados que possuem transição apenas para os últimos estados do modelo (bloco inferior à



esquerda com 259 estados). Este estados representam o sistema no  $N_{mc}$ -ésimo mini-ciclo.

As Tabelas 3.7 e 3.8 mostram o tempo médio de resposta para um pedido  $NC$  para  $\varepsilon = 1.0e - 05$  e  $\varepsilon = 1.0e - 03$ , respectivamente, quando a taxa de chegada de pedidos  $NC$  varia de 1 a 10. Note que os valores obtidos na solução de  $\mathcal{H}^- + \text{diag}(\mathcal{H}^\varepsilon e^T)$  e usando o algoritmo proposto são menores que os valores obtidos na solução de  $\mathbf{P}$ . Embora o algoritmo proposto apresente novamente uma melhor aproximação em comparação à solução exata.

Taxa de Chegada	$\mathcal{H}$	$\mathcal{H}^- + \text{diag}(\mathcal{H}^\varepsilon e^T)$	Aproximação Alg. Proposto
1	8.5309255855e-02	8.5211824070e-02	8.5249021836e-02
2	8.8155120799e-02	8.8042486353e-02	8.8091853646e-02
4	9.4480540868e-02	9.4364984173e-02	9.4430481616e-02
6	1.0130366183e-01	1.0119635330e-01	1.0127005822e-01
8	1.0827843193e-01	1.0817052606e-01	1.0825465280e-01
10	1.1508465436e-01	1.1497640075e-01	1.1506956013e-01

Tabela 3.7: Tempo médio de resposta de um pedido  $NC$  quando  $\varepsilon = 10e - 05$ .

Taxa de Chegada	$\mathcal{H}$	$\mathcal{H}^- + \text{diag}(\mathcal{H}^\varepsilon e^T)$	Aproximação Alg. Proposto
1	8.5309255855e-02	8.2554738838e-02	8.4659286935e-02
2	8.8155120799e-02	8.4853090716e-02	8.7640481682e-02
4	9.4480540868e-02	8.9013311956e-02	9.3782893023e-02
6	1.0130366183e-01	9.3946296685e-02	1.0057588108e-01
8	1.0827843193e-01	9.8744536955e-02	1.0758563017e-01
10	1.1508465436e-01	1.0330094600e-01	1.1452575879e-01

Tabela 3.8: Tempo médio de resposta de um pedido  $NC$  quando  $\varepsilon = 10e - 03$ .

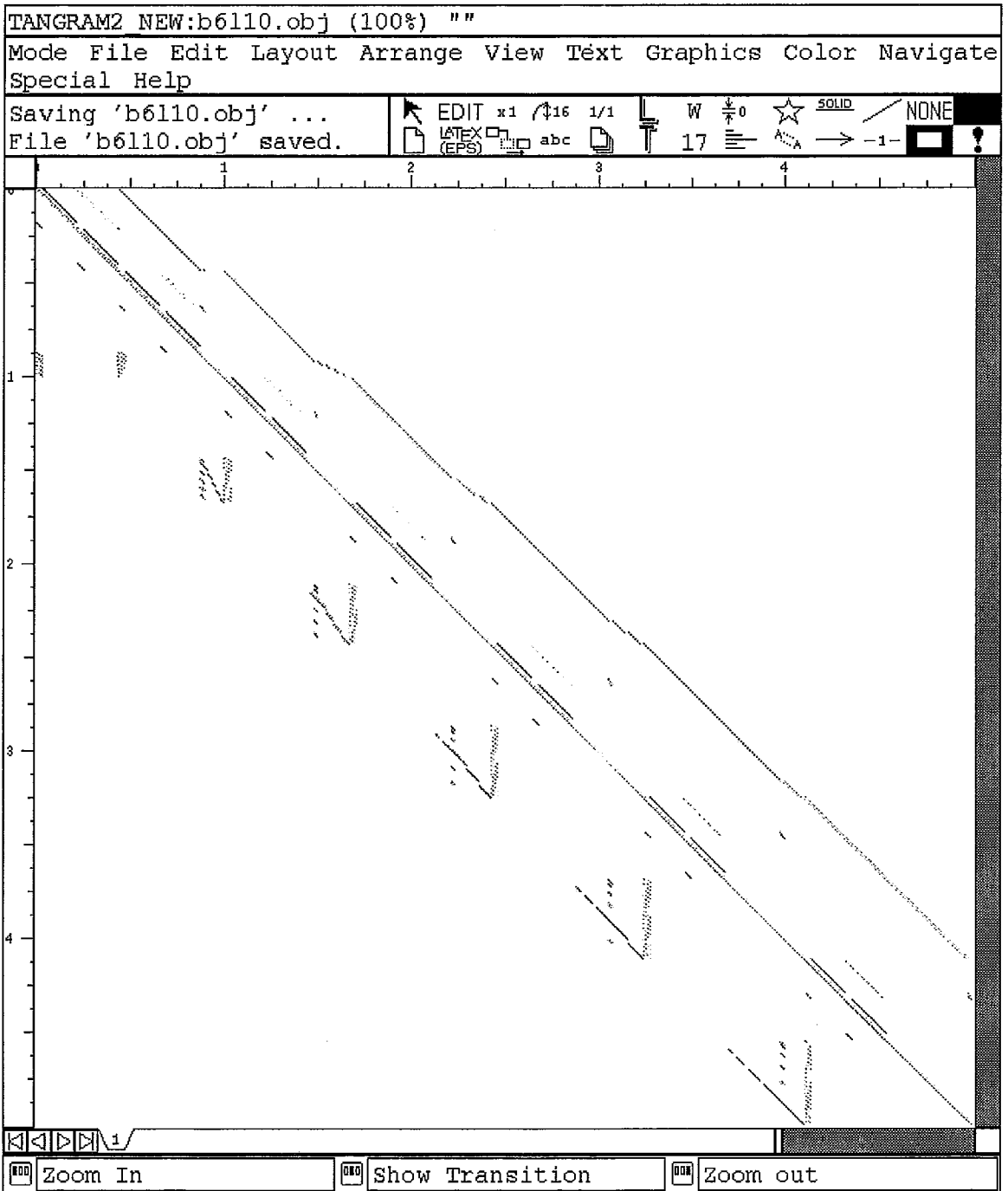


Figura 3.38: Matriz  $P$  do Modelo do Servidor de Disco.

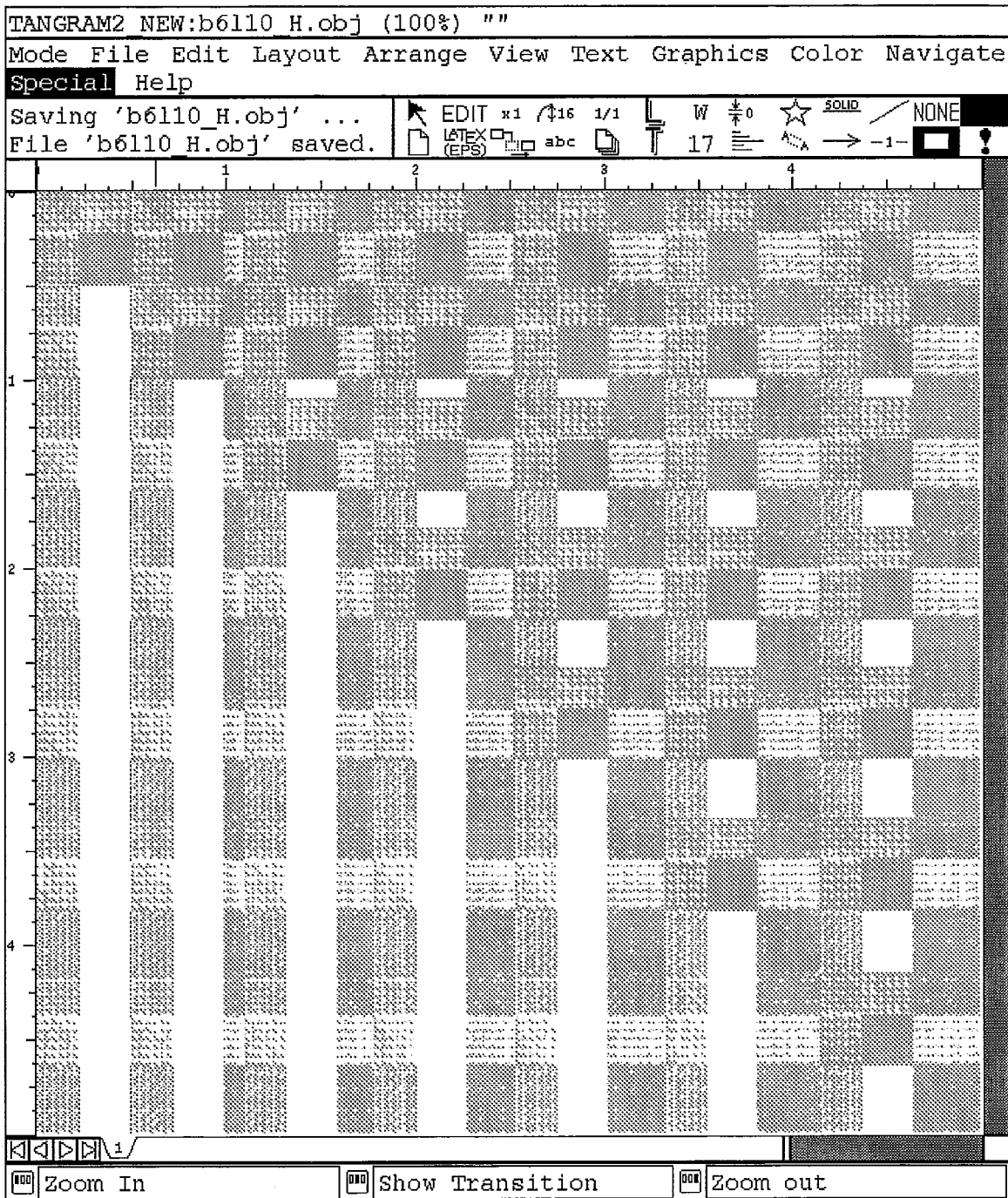


Figura 3.39: Matriz  $\mathcal{H}$  do Modelo do Servidor de Disco.

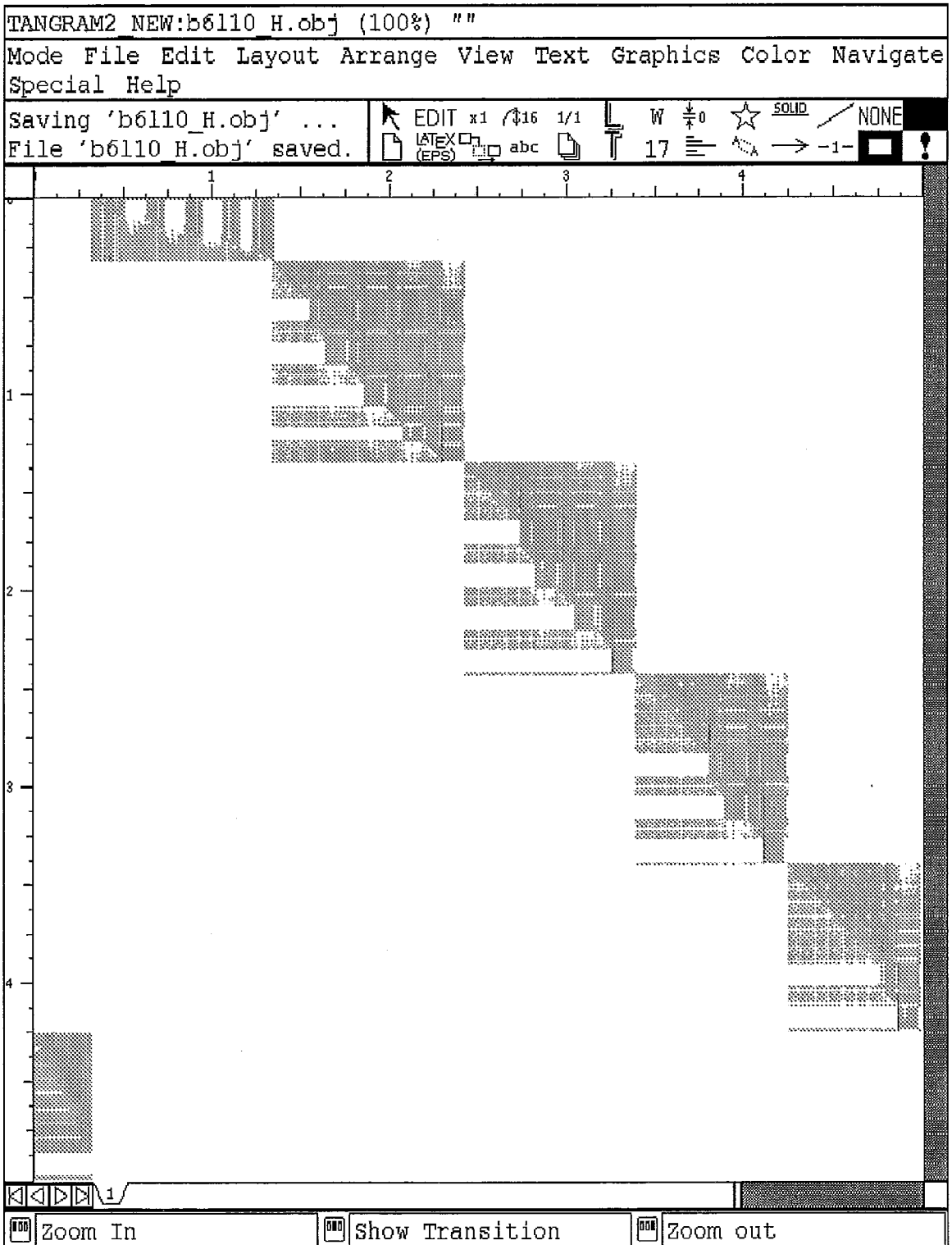


Figura 3.40: Matriz  $\mathcal{H}'$  do Modelo do Servidor de Disco.

A Figura 3.41 mostra o número de operações requeridas para os algoritmos quando a taxa de chegada de pedidos  $NC$  varia de 1 a 10 e  $\varepsilon$  é igual a  $1.0e-05$  e a  $1.0e-03$ . Note que o número de operações é praticamente constante para os três algoritmos quando variamos a taxa de chegada, e que existe uma diferença de pelo menos uma ordem de magnitude entre o número de operações do algoritmo GTH e o número de operações dos outros dois algoritmos.

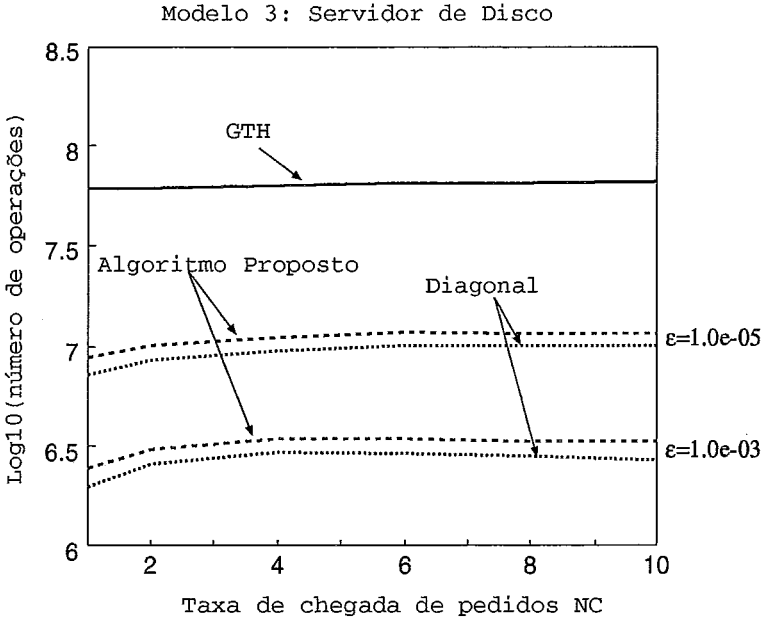


Figura 3.41: Número de operações requeridas na solução da matriz  $\mathcal{H}$ .

### 3.6 Conclusões

Um dos métodos de solução mais importantes na solução de cadeias de Markov é o método GTH [29]. Este método tem duas importantes características: em primeiro lugar, o algoritmo só faz uso de operações de multiplicação e adição, nenhuma subtração é realizada, evitando assim perda de precisão nos valores calculados; em segundo lugar, é possível fornecer uma explicação probabilística para cada passo executado pelo algoritmo. Entretanto, o uso do GTH tem algumas restrições, a principal delas é o custo computacional da  $O(N^3)$  se a matriz de transição de estados não tiver uma estrutura especial, onde  $N$  corresponde ao

número de estados do sistema sendo modelado. O uso deste método é apropriado quando  $N$  não é muito grande (da ordem de algumas centenas de estados) e quando a matriz de transição de estados do modelo não é esparsa.

Neste capítulo, propusemos uma aproximação para o método GTH de forma que ele possa ser usado na solução de matrizes com milhares de estados. O método proposto tem como base dois métodos de aproximação existentes na literatura ([31] e [32]). O novo método utiliza a idéia que probabilidades de transição com valores muito pequenos, podem ser eliminadas (ou redirecionadas) e mesmo assim termos uma boa aproximação para a solução exata do modelo. O objetivo em introduzir uma perturbação na matriz de probabilidades é diminuir o custo computacional da solução do modelo. Os resultados apresentados mostram que o novo algoritmo fornece uma boa aproximação para a solução de modelos markovianos e de modelos não markovianos. Sendo que apenas para os modelos markovianos foi obtido um limite para o erro cometido pelo novo algoritmo.

# Capítulo 4

## Distribuição da Recompensa Acumulada em um Tempo Finito

### 4.1 Introdução

No Capítulo 3 estudamos métodos de solução e de aproximação para modelos markovianos em estado estacionário. Como vimos, o conjunto das probabilidades estacionárias nos fornece o comportamento do sistema durante um intervalo  $t$ , onde  $t \rightarrow \infty$ , isto é, para um intervalo *suficientemente grande* para que as medidas de interesse cheguem a um valor estável.

Infelizmente, a análise em estado estacionário pode não fornecer todas as respostas sobre o comportamento do modelo a ser estudado. Para fim de exemplificação, considere novamente o primeiro modelo de *jitter* apresentado no Capítulo 2. Neste modelo, temos uma fonte que alterna períodos de geração de pacotes com longos períodos de silêncio. Durante um período ativo da fonte, pacotes são gerados com taxa constante e entregues ao nó da rede de dados mais próximo. Quando o equipamento no destino recebe o primeiro pacote após um período de silêncio, ele armazena o pacote e aguarda a chegada de mais  $N_{min} - 1$  pacotes

antes de iniciar a entrega ao usuário. Se um novo período de silêncio é detectado antes que existam  $N_{min}$  pacotes armazenados, a entrega dos pacotes recebidos é iniciada imediatamente. Se a fila for esvaziada após o início da entrega de pacotes, a entrega é reinicializada com a chegada do próximo pacote.

Na análise do modelo de *jitter* poderíamos questionar: qual o tempo de espera pelos  $N_{min} - 1$  pacotes após a chegada do primeiro pacote de um período ativo ? qual o tempo de espera por pacotes quando a fila esvazia após o início da entrega de pacotes ? qual a taxa de perda de pacotes durante um determinado intervalo de tempo  $t$  após o início de um período ativo da fonte ? etc. Note que estas questões possuem uma característica em comum: supõe-se que o modelo está em um determinado estado e deseja-se conhecer o comportamento do sistema após um tempo de observação finito. Portanto, para obter as respostas a essas perguntas precisamos fazer a análise *transiente* do modelo.

Um aspecto importante dos modelos que iremos estudar é a possibilidade de associar recompensas aos estados do modelo. A recompensa ganha corresponde a uma abstração do rendimento obtido pelo sistema durante o intervalo de observação. Pode-se atribuir dois diferentes tipos de recompensas a uma cadeia de Markov. O primeiro tipo de recompensa consiste em associar uma taxa de recompensa a cada estado da cadeia. Isto quer dizer que, por cada unidade de tempo em que o sistema permanece em um determinado estado, ele ganha a recompensa associada a esse estado. O segundo tipo consiste em atribuir uma recompensa ao sistema cada vez que este efetua uma certa transição. Este tipo de recompensa é chamado de recompensa por impulso, pois o valor da recompensa é somado uma única vez quando a transição ocorre.

Uma importante medida de interesse na análise transiente é a distribuição da recompensa acumulada. Por exemplo, suponha que o modelo de *jitter* possui dois tipos de recompensa : 0 e 1. A recompensa 1 está associada aos estados onde a entrega de pacotes está habilitada e a recompensa zero está associada aos outros estados do modelo. Neste caso, a distribuição da recompensa acumulada durante



um intervalo de tempo  $(0, t)$  corresponde ao tempo gasto pelo sistema entregando pacotes.

Formalmente, para uma cadeia de Markov homogênea de tempo contínuo  $\mathcal{X} = \{X(t) : t \geq 0\}$  com espaço de estados  $\mathcal{S} = \{s_i : i = 1, \dots, M\}$  e taxa de recompensa  $r_i$  associada ao estado  $s_i$ , podemos definir a recompensa acumulada durante o intervalo  $(0, t)$  como

$$CR(t) = \int_0^t IR(x)dx,$$

onde  $IR(t) = r_i$  se  $X(t) = s_i$ . A taxa de recompensa média acumulada no mesmo período de tempo é definida por

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

A distribuição da recompensa acumulada em um tempo finito já foi estudada em vários artigos. Em [47], E. de Souza e Silva e R. Gail usando a técnica de uniformização apresentaram um algoritmo para modelos com duas taxas de recompensa. Em [48], Smith *et al* utilizaram transformada de Laplace e inversão numérica para modelos com várias taxas de recompensa. Em [49], E. Souza e Silva e R. Gail utilizaram técnica de uniformização, e desenvolveram uma metodologia para o cálculo da recompensa acumulada em modelos gerais. A complexidade do algoritmo entretanto é exponencial em relação ao número de estados da cadeia de Markov. Em [50], Donatiello e Grassi fizeram uso da técnica de uniformização e de transformada de Laplace para obter um algoritmo com complexidade polinomial em relação ao número de estados e de recompensas. Em [52], de Souza e Silva *et al* propuseram um novo algoritmo que também faz uso da técnica de uniformização e que possui complexidade polinomial em relação a um parâmetro que é menor que o número de recompensas do modelo sendo estudado (portanto menor custo que o algoritmo de [50]). Uma grande vantagem de [52] é a interpretação probabilística das equações.

O principal problema dos algoritmos apresentados em [50] e [52] é a possibilidade de apresentarem problemas numéricos pois trabalham com valores positivos

e negativos. Além disso, não existe um limite de valor para os termos calculados pelos algoritmos, podendo ocorrer problemas de *overflow/underflow*. Mais recentemente, Nabli e Sericola [54] propuseram um algoritmo para cadeias de Markov que também possui uma complexidade polinomial, tendo como grande vantagem só trabalhar com números positivos e não maiores que 1.

Convém observar que os algoritmos citados acima trabalham apenas com taxas de recompensa. Um algoritmo com complexidade polinomial para cálculo da distribuição da recompensa acumulada para cadeias de Markov com taxas de recompensa e/ou recompensas por impulso é apresentado em [51, 53]. Outra observação importante diz respeito ao algoritmo proposto em [54]. Não há uma interpretação probabilística para este algoritmo. Isto significa que os termos apresentados na solução não possuem um significado associado ao comportamento do sistema sendo modelado.

Neste capítulo vamos estudar os algoritmos propostos em [53] e [54] para o cálculo da distribuição da recompensa acumulada em um tempo finito. O objetivo é fornecer uma interpretação probabilística para o algoritmo de [54] usando a mesma metodologia de [49]. Em seguida, vamos mostrar um algoritmo para a distribuição de recompensa acumulada com limites que foi apresentado em [61] e que se tornou possível a partir da interpretação probabilística aqui mostrada. Também apresentaremos neste capítulo dois novos algoritmos de combinação linear de estatísticas de ordem que foram obtidos durante o estudo do algoritmo de [54]. A importância desses algoritmos será discutida no decorrer do trabalho.

A apresentação deste capítulo é descrita a seguir. Na seção 4.2, discutimos os algoritmos propostos em [52] (e a metodologia de [49]) e [54], além de revermos alguns conceitos utilizados pelos algoritmos a serem estudados. Na seção 4.3 apresentamos dois novos algoritmos para combinação linear de estatísticas de ordem. Na seção 4.4 apresentamos uma interpretação probabilística para o algoritmo de [54] e exemplificamos o seu uso com um algoritmo para a distribuição de recompensa acumulada com limites. Na seção 4.5 apresentamos as nossas

conclusões.

## 4.2 Algoritmos

Nesta seção são discutidos os algoritmos para o cálculo da distribuição da recompensa acumulada apresentados em [52] e [54]. Inicialmente vamos descrever brevemente a técnica de uniformização (seção 4.2.1) e combinação linear de estatísticas de ordem (seção 4.2.2). Os conceitos apresentados nestas duas seções são utilizados em 4.2.3 quando discutimos o algoritmo proposto em [52]. Em seguida, na seção 4.2.4 fazemos uma breve revisão de processo de renovação que é utilizado na definição do algoritmo proposto em [54]. Este segundo algoritmo é estudado na seção 4.2.5.

### 4.2.1 Técnica de Uniformização

A técnica de uniformização foi proposta em 1953 por Jensen [33] e consiste em transformar uma cadeia de Markov de tempo contínuo em uma cadeia de Markov equivalente de tempo discreto. Dessa forma, soluções transientes podem ser obtidas trabalhando com o problema em tempo discreto, o que pode facilitar a obtenção da solução desejada. A seguir fazemos uma breve apresentação desta técnica. Veja [34] para uma discussão mais detalhada sobre este método e sua utilização na área de análise de desempenho de sistemas de comunicação/computação.

Seja  $\mathcal{X} = \{X(t) : t \geq 0\}$  uma cadeia de Markov homogênea de tempo contínuo com espaço de estados  $\mathcal{S} = \{s_i : i = 1, \dots, M\}$  e matriz de taxas de transição  $\mathbf{Q}$ . Seja  $\mathcal{Z} = \{Z_n : n = 0, 1, \dots\}$  uma cadeia de Markov de tempo discreto com o mesmo espaço de estados  $\mathcal{S}$  e matriz de probabilidades de transição  $\mathbf{P}$ . Suponha que  $\mathbf{P} = \mathbf{I} + \mathbf{Q}/\Lambda$ , onde  $\mathbf{I}$  é a matriz identidade  $M \times M$  e  $\Lambda$  tem um valor

maior ou igual à maior taxa de saída dos estados de  $\mathbf{Q}$ . Pode ser mostrado que a taxa de transição do estado  $i$  para o estado  $j$ , onde  $i \neq j$ , é a mesma em  $\mathcal{X}$  e em  $\mathcal{Z}$ . Portanto, temos  $X(t) = Z_{N(t)}$  para  $t \geq 0$ , supondo que os tempos de transição são dados por um processo de Poisson.  $\mathcal{N} = \{N(t) : t \geq 0\}$  com taxa  $\Lambda$  e independente de  $\mathcal{Z}$ .

Seja  $\pi(t) = \langle \pi_1(t), \pi_2(t), \dots, \pi_M(t) \rangle$  o vetor de probabilidades transientes, onde  $\pi_i(t)$  é a probabilidade do processo se encontrar no estado  $i$  no tempo  $t$ ,  $i = 1, \dots, M$ . Descondicionando no número de transições no período  $(0, t)$ , temos

$$\pi(t) = \sum_{n=0}^{\infty} e^{-\Lambda t} \frac{(\Lambda t)^n}{n!} \pi(0) \mathbf{P}^n, \quad (4.1)$$

onde  $\pi(0)$  é o vetor de probabilidades iniciais e  $\mathbf{P}^n$  corresponde ao  $n$ -ésimo passo da matriz de probabilidades de transição de  $\mathcal{Z}$ .

Para avaliar numericamente a equação 4.1, podemos truncar a série infinita para um valor  $N$ , e o erro resultante do truncamento pode ser estimado a partir dos termos restantes da Poisson. Portanto, temos

$$\pi(t) = \sum_{n=0}^N e^{-\Lambda t} \frac{(\Lambda t)^n}{n!} \pi(0) \mathbf{P}^n + \epsilon(N), \quad (4.2)$$

onde

$$\epsilon(N) \leq 1 - \sum_{n=0}^N e^{-\Lambda t} \frac{(\Lambda t)^n}{n!}. \quad (4.3)$$

## 4.2.2 Combinação Linear de Estatísticas de Ordem

Seja  $U_1, U_2, \dots, U_n$  um conjunto de variáveis aleatórias (v.a.'s) contínuas, uniformes, independentes e identicamente distribuídas no intervalo  $(0, 1)$ . Seja  $U_{(i)}$

o  $i$ -ésimo menor valor entre estas v.a.'s, portanto  $U_{(1)} < U_{(2)} < \dots < U_{(n)}$ . Dizemos que  $U_{(1)}, U_{(2)}, \dots, U_{(n)}$  são as estatísticas de ordem das v.a.'s  $U_1, U_2, \dots, U_n$ . A estatística de ordem  $U_{(i)}$  no intervalo  $(0, 1)$  tem a mesma distribuição de  $tU_{(i)}$  no intervalo  $(0, t)$  para  $i = 1, \dots, n$ . A Figura 4.1 mostra um exemplo com  $n = 5$  e intervalo  $(0, t)$ . Neste exemplo a v.a. com menor valor é  $U_2$  e a com maior valor é  $U_4$ . Definimos  $U_{(0)} = 0$  e  $U_{(n+1)} = 1$ .

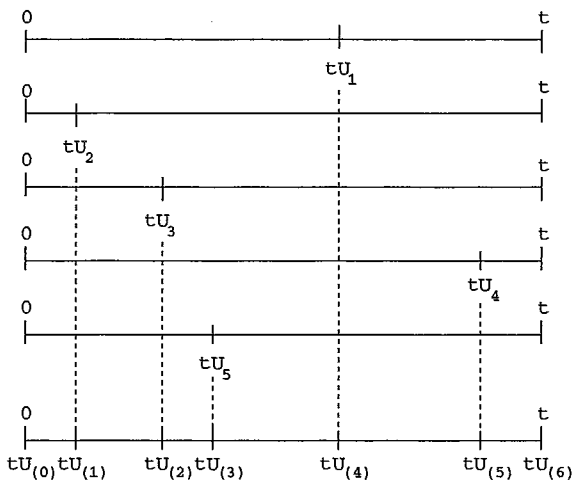


Figura 4.1: Exemplo de estatísticas de ordem com 5 variáveis.

Considere agora a combinação linear de estatísticas de ordem  $U_{(k)}$  definida pela seguinte equação:

$$c_{n+1} = \sum_{j=1}^{n+1} a_j U_{(j)},$$

onde  $a_k$  é um número real positivo,  $1 \leq k \leq n + 1$ . A função distribuição de  $c_{n+1}$  é dada por

$$P[c_{n+1} \leq r] = P \left[ \sum_{k=1}^{n+1} a_k U_{(k)} \leq r \right].$$

Suponha  $Y_k = t(U_{(k)} - U_{(k-1)})$  e  $d_k = \sum_{i=k}^{n+1} a_i$ ,  $k = 1, \dots, n + 1$  (veja Figura 4.2). Claramente temos que [56]

$$\sum_{k=1}^{n+1} a_k U_{(k)} = \sum_{k=1}^{n+1} d_k Y_k$$

e portanto,

$$P[c_{n+1} \leq r] = P \left[ \sum_{k=1}^{n+1} d_k Y_k \leq r \right]. \quad (4.4)$$

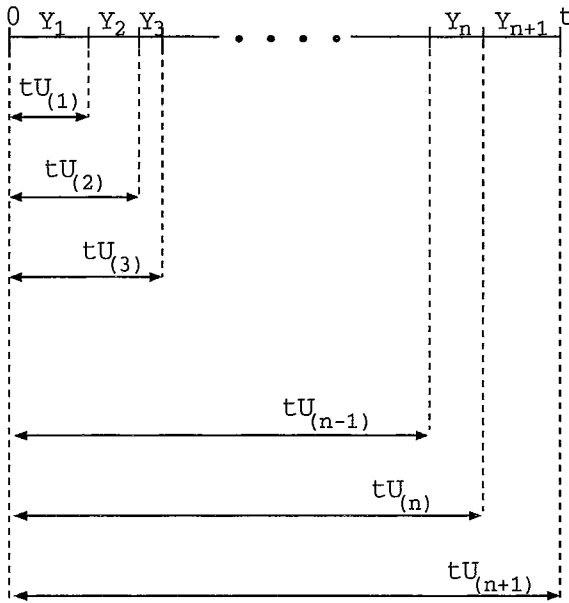


Figura 4.2: Combinação linear de estatísticas de ordem  $U_{(k)}$ .

Em [49] foi demonstrado que a recompensa total acumulada por uma cadeia de Markov no intervalo  $(0, t)$  dado que ocorreram  $n$  transições nesse intervalo pode ser obtida através de combinação linear de estatísticas de ordem (eq. 4.4). A seguir discutiremos esta interpretação.

Seja  $\mathcal{X} = \{X(t) : t \geq 0\}$  uma cadeia de Markov homogênea de tempo contínuo com espaço de estados  $\mathcal{S} = \{s_i : i = 1, \dots, M\}$  e matriz de taxas de transição  $\mathbf{Q}$ . Seja  $\mathcal{Z} = \{Z_n : n = 0, 1, \dots\}$  uma cadeia de Markov de tempo discreto com o mesmo espaço de estados  $\mathcal{S}$  e matriz de probabilidades de transição  $\mathbf{P} = \mathbf{I} + \mathbf{Q}/\Lambda$ , onde  $\Lambda$  tem um valor maior ou igual à maior taxa de saída dos estados de  $\mathbf{Q}$ ,

portanto  $\mathcal{Z}$  é o processo obtido de  $\mathcal{X}$  após uniformização com taxa  $\Lambda$ . Conforme indicado em 4.1, o número de transições em um estado  $s_i \in \mathcal{S}$  antes de sair para um estado  $s_j$  ( $j \neq i$ ) é dado por um processo Poisson  $\mathcal{N} = \{N(t) : t \geq 0\}$  com taxa  $\Lambda$ .

Suponha que  $n$  transições ocorrem em  $\mathcal{X}$  durante o intervalo  $(0, t)$  dividindo  $(0, t)$  em  $n+1$  subintervalos. Seja  $\mathbf{k}$  um vetor tal que o  $l$ -ésimo componente indica o número de vezes que a recompensa  $r_l$ , associada a um conjunto de estados de  $\mathcal{Z}$ , apareceu em um *caminho amostral*  $v_n$  de  $\mathcal{Z}$ .

Note que mais de um subintervalo pode estar associado a uma mesma taxa de recompensa. Seja  $\mathcal{R} = \{r_1, \dots, r_{L+1}\}$  o conjunto de taxas de recompensas de  $\mathcal{X}$ , onde  $r_1 > r_2 > \dots > r_L > r_{L+1} = 0$  e seja  $G_{\mathbf{k}}$  o conjunto de todos os vetores de estados de  $\mathcal{Z}$  de comprimento  $n+1$  tal que o vetor  $\mathbf{k}$  seja o mesmo. Para  $v \in G_{\mathbf{k}}$ ,

$$CR(t)|n, \mathbf{k}, v = \sum_{j=1}^{n+1} d_{j,v} Y_j, \quad (4.5)$$

onde  $d_{j,v}$  é a recompensa associada ao  $j$ -ésimo subintervalo de  $(0, t)$ .

É possível demonstrar (ver [55]) que a distribuição conjunta das v.a.'s  $Y_k$ ,  $k = 1, \dots, n+1$ , é invariante sob qualquer permutação de  $1, \dots, n+1$ . Isto significa que podemos alterar a sequência das v.a.'s  $Y_k$  sem modificar a distribuição conjunta dos  $Y_k$ . Como consequência desta propriedade, e pelo fato do processo de Poisson  $\mathcal{N}$  e a cadeia discreta  $\mathcal{Z}$  serem independentes, é possível agrupar os intervalos com a mesma recompensa em sequência. Assim considera-se que os  $k_1$  primeiros subintervalos possuem recompensa  $r_1$ , os seguintes  $k_2$  subintervalos possuem recompensa  $r_2$ , e assim por diante. Portanto,  $CR(t)|n, \mathbf{k}, v$  é independente do vetor  $v$  que produziu  $\mathbf{k}$ , e logo depende apenas de  $\mathbf{k}$ .

Seja  $\xi(i)$  o índice da recompensa associada ao subintervalo  $i$  e  $n_j = \sum_{i=1}^j k_{\xi(i)}$ ,  $j = 1, \dots, L$ . Podemos então escrever

$$\begin{aligned}
ACR(t) | n, \mathbf{k} &= \frac{CRT(t) | n, \mathbf{k}}{t} \\
&= \sum_{j=1}^{L+1} r_{\xi(j)} (U_{(n_j)} - U_{(n_{j-1})}) \\
&= \sum_{j=1}^L (r_{\xi(j)} - r_{\xi(j+1)}) U_{(n_j)} + r_{\xi(L+1)}
\end{aligned}$$

Portanto,

$$P[ACR(t) > r | n, \mathbf{k}] = P \left[ \sum_{j=1}^L (r_{\xi(j)} - r_{\xi(j+1)}) U_{(n_j)} > r - r_{\xi(L+1)} \right]. \quad (4.6)$$

A equação 4.6 nos mostra que para calcularmos  $P[ACR(t) \leq r | n, \mathbf{k}]$ , precisamos obter a distribuição da combinação linear das estatísticas de ordem no intervalo  $(0, 1)$ . Baseado em resultado de Weisberg [57] pode ser mostrado que (ver Lema 1 de [52]):

$$P[ACR(t) > r | n, \mathbf{k}] = \sum_{i: r_i > r} \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} f_i(r, \mathbf{k}) \quad (4.7)$$

e

$$P[ACR(t) \geq r | n, \mathbf{k}] = \sum_{i: r_i \geq r} \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} f_i(r, \mathbf{k}), \quad (4.8)$$

onde

$$f_i(r, \mathbf{k}) = \frac{(r_i - r)^n}{\prod_{\substack{j=1 \\ j \neq i}}^{L+1} (r_i - r_j)^{k_j}} \quad (4.9)$$

e (por definição)

$$\frac{d^{(k_i-1)}}{dr_i} f_i(r, \mathbf{k}) = \begin{cases} f_i(r, \mathbf{k}) & \text{se } k_i = 1 \\ 0 & \text{se } k_i < 1 \end{cases} \quad (4.10)$$

Note que, para  $r_l \in \mathcal{R}$  e  $k_l > 0$ , podemos escrever a equação 4.9 da seguinte



forma:

$$f_i(r, \mathbf{k}) = \begin{cases} \left( \frac{r_i - r}{r_i - r_l} \right) f_i(r, \mathbf{k} - \mathbf{1}_l) & \text{se } l \neq i \\ (r_i - r) f_i(r, \mathbf{k} - \mathbf{1}_l) & \text{se } l = i \end{cases} \quad (4.11)$$

### 4.2.3 Algoritmo 1

Utilizando os conceitos discutidos nas seções 4.2.1 e 4.2.2, podemos escrever a distribuição da recompensa média acumulada em um tempo  $t$  como [52]

$$P[ACR(t) > r] = \sum_{n=0}^{\infty} e^{-\Lambda t} \frac{(\Lambda t)^n}{n!} \sum_{\mathbf{k} \in \mathcal{K}_n} \Gamma[n, \mathbf{k}] P[ACR(t) > r | n, \mathbf{k}], \quad (4.12)$$

onde  $\mathcal{K}_n = \{\mathbf{k} : \|\mathbf{k}\| = \sum_{j=1}^{L+1} k_j = n + 1\}$  e  $\Gamma[n, \mathbf{k}]$  é a probabilidade de se obter um vetor  $\mathbf{k}$  dadas  $n$  transições em  $(0, t)$ . (Em [49] identifica-se a associação de recompensas a intervalos como *colorir* um intervalo. Desta forma, uma *coloração*  $\mathbf{k}$  indica um determinado valor para o vetor  $\mathbf{k}$ . No que se segue usamos a mesma notação de [49].)

A seguir vamos mostrar a solução apresentada em [52] para a equação 4.12. Podemos dividir a metodologia usada na definição do algoritmo de [52] em três partes. Na primeira parte é apresentada uma recursão para  $\Gamma[n, \mathbf{k}]$ . Em seguida, é apresentada uma recursão para  $P[ACR(t) > r | n, \mathbf{k}]$ . E por último, a partir dessas duas recursões é feita uma agregação de termos na soma em  $\mathbf{k}$  de 4.12 de forma a reduzir a complexidade computacional.

Podemos escrever  $\Gamma[n, \mathbf{k}]$  como

$$\Gamma[n, \mathbf{k}] = \sum_{s \in \mathbf{S}} \Gamma_s[n, \mathbf{k}],$$

onde  $\Gamma_s[n, \mathbf{k}]$  corresponde à probabilidade de uma coloração  $\mathbf{k}$  após  $n$  transições

e onde  $s$  é o último estado visitado. Seja  $p_{s,s'}$  a probabilidade de transição do estado  $s$  para o estado  $s'$ . Uma recursão para  $\Gamma_s[n, \mathbf{k}]$  pode então ser obtida [49] por

$$\Gamma_s[n, \mathbf{k}] = \sum_{s' \in \mathbf{S}} \Gamma_{s'}[n-1, \mathbf{k} - \mathbf{1}_{c(s)}] p_{s',s}, \quad (4.13)$$

onde  $c(s)$  corresponde ao índice da recompensa associada ao estado  $s$  e  $\mathbf{1}_{c(s)}$  é um vetor de tamanho  $L+1$  com valor 1 na posição  $c(s)$  e valor zero nas outras posições. As condições iniciais para a recursão mostrada em 4.13 são

$$\Gamma_s[0, \mathbf{1}_i] = \begin{cases} \pi_s(0) & \text{se } i = c(s) \\ 0 & \text{se } i \neq c(s) \end{cases}$$

onde  $\pi_s(0)$  é a probabilidade do processo se encontrar inicialmente no estado  $s \in \mathbf{S}$ .

Precisamos agora encontrar uma recursão para a combinação linear de estatísticas de ordem. Vimos em 4.2.2 que

$$P[ACR(t) > r | n, \mathbf{k}] = \sum_{i: r_i > r} \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} f_i(r, \mathbf{k}). \quad (4.14)$$

Em [52] é apresentada uma recursão para o cálculo de cada termo do somatório mostrado em 4.14. Esta recursão é reproduzida a seguir.

Seja  $j$  um índice tal que  $k_j > 0$  e  $\mathbf{k} \in \mathcal{K}_n$ . Para  $n \geq 1$  temos

$$\frac{1}{l!} \frac{d^{(l)}}{dr_i} f_i(r, \mathbf{k}) =$$

$$\begin{cases} \left( \frac{1}{r_i - r_j} \right) \frac{1}{(l-1)!} \frac{d^{(l-1)}}{dr_i} (f_i(r, \mathbf{k} - \mathbf{1}_j) - f_i(r, \mathbf{k})) + \left( \frac{r_i - r}{r_i - r_j} \right) \frac{1}{l!} \frac{d^{(l)}}{dr_i} f_i(r, \mathbf{k} - \mathbf{1}_j) \\ \text{se } i \neq j \\ \\ \frac{1}{(l-1)!} \frac{d^{(l-1)}}{dr_i} f_i(r, \mathbf{k} - \mathbf{1}_j) + (r_i - r) \frac{1}{l!} \frac{d^{(l)}}{dr_i} f_i(r, \mathbf{k} - \mathbf{1}_j) \text{ se } i = j \end{cases}$$

e para  $n = 0$ ,  $k_j = 1$  e  $l > 0$  temos

$$\frac{1}{l!} \frac{d^{(l)}}{dr_i} f_i(r, \mathbf{k}) = \begin{cases} - \left( \frac{1}{r_i - r_j} \right) \frac{1}{(l-1)!} \frac{d^{(l-1)}}{dr_i} f_i(r, \mathbf{k}) & \text{se } i \neq j \\ 0 & \text{se } i = j \end{cases}$$

As condições iniciais da recursão acima são

$$\frac{d^{(0)}}{dr_i} f_i(r, \mathbf{k}) = \begin{cases} \frac{1}{r_i - r_j} & \text{se } i \neq j \\ 1 & \text{se } i = j \end{cases}$$

Note que temos uma recursão para  $\Gamma[n, \mathbf{k}]$  e uma recursão para  $P[ACR(t) > r | n, \mathbf{k}]$ . O próximo passo é combinar estas duas recursões em uma única recursão.

Para isso fazemos uso de duas definições:

$$G_g[i, n] = \{\mathbf{k} \in \mathcal{K}_n : k_i = g\} \quad (4.15)$$

e

$$F_{g,s}[i, n] = \{\mathbf{k} \in G_g[i, n] : k_{c(s)} > 0\}, \quad (4.16)$$

para  $n \geq 0$ ,  $i = 1, \dots, L+1$ ,  $g = 0, \dots, n+1$  e  $s \in \mathbf{S}$ . A definição 4.15 agrupa os vetores  $\mathbf{k}$  que possuem um mesmo valor  $k_i = g$ , enquanto a definição 4.16 agrupa os vetores  $\mathbf{k}$  que possuem um mesmo valor  $k_i = g$  e possuem valor  $k_{c(s)} > 0$  para

$s \in \mathbf{S}$ . Em [51] é mostrado que

$$\{\mathbf{k} - \mathbf{1}_{c(s)} : \mathbf{k} \in F_{g,s}[i, n]\} = \begin{cases} G_g[i, n-1] & \text{se } i \neq c(s) \text{ e } g \geq 0 \\ G_{g-1}[i, n-1] & \text{se } i = c(s) \text{ e } g > 0 \end{cases} \quad (4.17)$$

Para  $n \geq 0$ ,  $i = 1, \dots, L+1$ ,  $s \in \mathcal{S}$  e  $u \geq 0$ , podemos então definir

$$\Upsilon_s[i, n, u] = \sum_{g=0}^{n+1} \sum_{\mathbf{k} \in F_{g,s}[i, n]} \Gamma_s[n, \mathbf{k}] \frac{1}{(g+u-n-1)!} \frac{d^{(g+u-n-1)}}{dr_i} f_i(r, \mathbf{k}). \quad (4.18)$$

Os valores  $\Upsilon_s[i, n, u]$  são calculados recursivamente em [52]. Para  $n \geq 1$ ,  $u \geq 0$  e para  $n = 0$ ,  $u > 1$ , temos

$$\Upsilon_s[i, n, u] = \begin{cases} (\{\Upsilon[i, n-1, u-2] + w_i \Upsilon[i, n-1, u-1]\} \mathbf{P}[:s] - \Upsilon_s[i, n, u-1]) / w_{i, c(s)} & \text{se } c(s) \neq i \\ \{\Upsilon[i, n-1, u-1] + w_i \Upsilon[i, n-1, u]\} \mathbf{P}[:s] & \text{se } c(s) = i \end{cases} \quad (4.19)$$

onde  $w_{i,j} = r_i - r_j$  para  $i \neq j$ ,  $w_i = r_i - r$ ,  $\mathbf{P}[:s]$  é a  $s$ -ésima coluna da matriz de probabilidades de transição  $\mathbf{P}$  e  $\Upsilon[i, n, u] = \langle \Upsilon_{s_1}[i, n, u], \dots, \Upsilon_{s_M}[i, n, u] \rangle$ . As condições iniciais ( $n = 0, u = 0, 1$ ) para a recursão de 4.19 são

$$\Upsilon_s[i, 0, u] = \begin{cases} 0 & \text{se } c(s) \neq i \text{ e } u = 0 \\ \pi_s(0) / w_{i, c(s)} & \text{se } c(s) \neq i \text{ e } u = 1 \\ \pi_s(0) & \text{se } c(s) = i \text{ e } u = 0 \\ 0 & \text{se } c(s) = i \text{ e } u = 1 \end{cases}$$

O Teorema 1 de [52] mostra que:

$$P[ACR(t) > r] = \sum_{n=0}^{\infty} e^{-\Lambda t} \frac{(\Lambda t)^n}{n!} \sum_{s \in \mathcal{S}} \sum_{i: r_i > r} \Upsilon_s[i, n, n]. \quad (4.20)$$

Basicamente o resultado de 4.20 é provado substituindo 4.7 em 4.12, fazendo inversão dos somatórios e comparando com 4.18 para  $n = u$ .

Note que o cálculo de  $\Upsilon[i, n, u]$  só depende dos valores computados no passo  $n - 1$  e no próprio passo  $n$ . Sabemos que  $\Upsilon[i, n, u]$  é um vetor com  $M$  posições, onde  $M$  é o número de estados do processo markoviano sendo modelado. Portanto, na implementação do algoritmo são necessários dois vetores de tamanho  $(N + 1)M$ , onde  $N$  é o ponto de truncagem da série infinita mostrada na equação 4.20 (veja [52] para o cálculo do erro introduzido pela truncagem). O custo computacional do algoritmo para cálculo dos  $\Upsilon[i, n, u]$  é  $O(MN^2)$  para um dado valor de  $i$ .

#### 4.2.4 Processo de Renovação

Nesta seção faremos uma breve revisão de processo de renovação. Os conceitos aqui apresentados são usados pelo algoritmo proposto em [54] e que iremos discutir na próxima seção. Veja [60] para um estudo mais detalhado sobre este assunto.

Considere o processo  $\{N(t), t > 0\}$  que indica o número de eventos no intervalo  $(0, t)$ , onde o intervalo entre sucessivos eventos é independente e identicamente distribuído com uma distribuição arbitrária  $F$ . Suponha que o intervalo entre dois eventos independe dos intervalos entre eventos ocorridos anteriormente, embora possuam a mesma distribuição. O processo  $\{N(t), t > 0\}$  é chamado de processo de renovação [60].

Seja  $X_n, n \geq 1$ , uma v.a. contínua que representa o intervalo entre o  $(n - 1)$  e  $n$ -ésimo evento. Suponha que o tempo até o primeiro evento seja  $x$ , isto é,  $X_1 = x$ . Podemos então calcular o número médio de eventos (renovações) no intervalo  $(0, t)$  como

$$m(t) = \int_0^\infty E[N(t)|X_1 = x]f(x)dx,$$

onde  $f$  é a função densidade de  $F$ . Como a função distribuição  $F$  é comum para todos os intervalos, temos

$$E[N(t)|X_1 = x] = 1 + E[N(t - x)], \text{ se } x < t.$$

Portanto,

$$\begin{aligned} m(t) &= \int_0^t [1 + m(t - x)]f(x)dx \\ &= F(t) + \int_0^t m(t - x)f(x)dx. \end{aligned}$$

A equação acima é conhecida como *equação básica do processo de renovação*.

#### 4.2.5 Algoritmo 2

Seja  $\mathcal{X} = \{X(t) : t \geq 0\}$  uma cadeia de Markov homogênea de tempo contínuo com espaço de estados  $\mathcal{S} = \{s_i : i = 1, \dots, M\}$  e matriz de taxas de transição  $\mathbf{Q}$ . Seja  $\mathcal{R} = \langle r_1, r_2, \dots, r_{L+1} \rangle$  o conjunto de taxas de recompensa associado a  $\mathcal{X}$ , onde  $r_1 > r_2 > \dots > r_{L+1}$ . A recompensa associada ao estado  $s \in \mathbf{S}$  é  $r_{c(s)}$ ,  $c(s) = 1, \dots, L + 1$ . Suponha  $\mathcal{Z}$  a cadeia de Markov uniformizada no espaço de estados  $\mathcal{S}$  com taxa de uniformização  $\Lambda$  e matriz de probabilidades de transição  $\mathbf{P}$ .

Seja

$$F_s(r, t, n) = \text{Prob}[ACR(t) > r, N(t) = n | \pi_s(0) = 1],$$

onde  $\pi_s(0)$  é a probabilidade do processo se encontrar inicialmente no estado  $s \in \mathbf{S}$ . A distribuição da recompensa média acumulada no período  $(0, t)$  com  $n$  transições pode então ser definida [54] como

$$P[ACR(t) > r] = \sum_{s \in \mathbf{S}} \pi_s(0) \sum_{n=0}^{\infty} F_s(r, t, n). \quad (4.21)$$

Utilizando a equação básica do processo de renovação e após uniformizar a cadeia de Markov, dois casos são discutidos em [54]:

- se  $r < r_{c(s)}$ :

$$\begin{aligned} F_s(r, t, n) &= \sum_{s' \in \mathbf{S}} p_{s, s'} \int_0^{\frac{rt}{r_{c(s)}}} F_{s'}(rt - r_{c(s)}u, t - u, n - 1) \Lambda e^{-\Lambda u} du \\ &+ e^{-\Lambda t} \frac{\Lambda^n}{n!} \left(t - \frac{rt}{r_{c(s)}}\right)^n, \end{aligned}$$

- se  $r \geq r_{c(s)}$ :

$$F_s(r, t, n) = \sum_{s' \in \mathbf{S}} p_{s, s'} \int_0^{\frac{(r_1 - r)t}{r_1 - r_{c(s)}}} F_{s'}(rt - r_{c(s)}u, t - u, n - 1) \Lambda e^{-\Lambda u} du.$$

A partir do estudo dos dois casos mostrados acima, é provado em [54] que:

$$P[ACR(t) > r] = \sum_{n=0}^{\infty} e^{-\Lambda t} \frac{(\Lambda t)^n}{n!} \sum_{k=0}^n \binom{n}{k} s_j^k (1 - s_j)^{n-k} \sum_{s=1}^M \pi_s(0) b_s^j(n, k), \quad (4.22)$$

onde  $s_j = \frac{r - r_{j+1}}{r_j - r_{j+1}}$  para  $r_{j+1} \leq r < r_j$ . A prova baseia-se na solução recursiva da integral acima usando integral por partes e de longas manipulações algébricas sem a utilização de argumentos probabilísticos.

Os coeficientes  $b_s^j(n, k)$  são calculados recursivamente em [54] da seguinte forma:

- se  $1 \leq c(s) \leq j$  (portanto,  $r_{c(s)} \geq r_j$ )

$$b_s^L(n, 0) = 1,$$

$$b_s^j(n, 0) = b_s^{j+1}(n, n) \text{ se } j < L,$$

$$b_s^j(n, k) = \left( \frac{r_{c(s)} - r_j}{r_{c(s)} - r_{j+1}} \right) b_s^j(n, k-1) + \left( 1 - \frac{r_{c(s)} - r_j}{r_{c(s)} - r_{j+1}} \right) \sum_{s' \in \mathbf{S}} b_{s'}^j(n-1, k-1) p_{s, s'}.$$

- se  $j+1 \leq c(s) \leq L+1$  (portanto,  $r_{c(s)} < r_j$ )

$$b_s^1(n, n) = 0,$$

$$b_s^j(n, n) = b_s^{j-1}(n, 0) \text{ se } j > 1,$$

$$b_s^j(n, k) = \left( \frac{r_{j+1} - r_{c(s)}}{r_j - r_{c(s)}} \right) b_s^j(n, k+1) + \left( 1 - \frac{r_{j+1} - r_{c(s)}}{r_j - r_{c(s)}} \right) \sum_{s' \in \mathbf{S}} b_{s'}^j(n-1, k) p_{s, s'}.$$

Seja  $\mathbf{b}^j(n, k) = \langle b_{s_1}^j(n, k), b_{s_2}^j(n, k), \dots, b_{s_M}^j(n, k) \rangle$ , onde  $j = 1, \dots, L$  e  $s_i \in \mathbf{S}$  para  $i = 1, \dots, M$ . Podemos classificar os elementos do vetor  $\mathbf{b}^j(n, k)$  em dois grupos:  $G_j$  e  $L_j$ ,  $j = 1, \dots, L$ . O elemento  $b_s^j(n, k) \in G_j$  se  $r_{c(s)} \geq r_j$  e  $b_s^j(n, k) \in L_j$  se  $r_{c(s)} < r_j$ . No primeiro caso temos, pela definição do algoritmo, o valor inicial de  $b_s^j(n, 0)$  e portanto, podemos calcular  $b_s^j(n, 1)$ , depois  $b_s^j(n, 2)$ , e assim por diante até  $b_s^j(n, n)$ . No segundo caso temos o valor inicial de  $b_s^j(n, n)$  e portanto, podemos calcular  $b_s^j(n, n-1)$ , depois  $b_s^j(n, n-2)$ , e assim por diante até  $b_s^j(n, 0)$ . A Figura 4.3 mostra a sequência de cálculos feita no algoritmo, onde a célula  $(n, k)$  representa o vetor  $\mathbf{b}^j(n, k)$ . Note que o cálculo de  $b_s^j(n, k)$  só depende dos valores computados no passo  $n-1$  do algoritmo e no próprio passo  $n$ . Logo, na implementação do algoritmo são necessários dois vetores de tamanho  $(N+1)M(L-1)$ , onde  $N$  é o ponto de truncagem da série infinita mostrada na equação 4.22,  $M$  é o número de estados do processo e  $L+1$  é o número de recompensas do modelo. O custo computacional do algoritmo é  $O(LMN^2)$ .



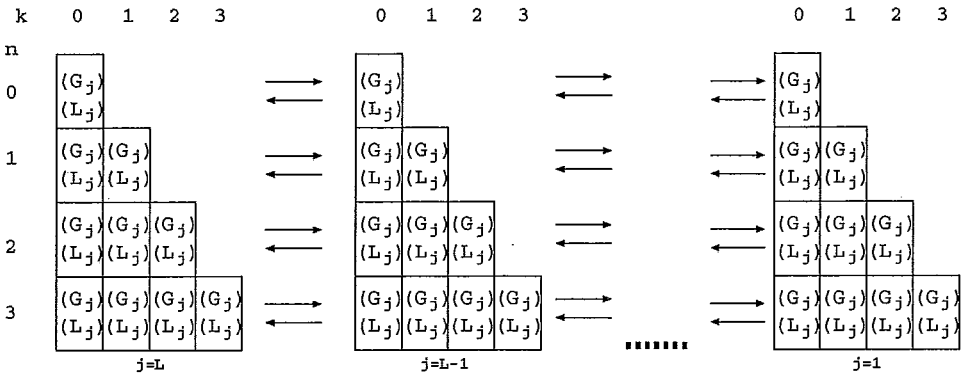


Figura 4.3: Cálculo de  $b^j(n, k)$ .

### 4.3 Novos Algoritmos de Combinação Linear de Estatísticas de Ordem

Vimos na seção 4.2.2 que a função de distribuição da combinação linear de estatísticas de ordem é definida como

$$P[c_{n+1} \leq r] = P \left[ \sum_{k=1}^{n+1} a_k U_{(k)} \leq r \right] = P \left[ \sum_{k=1}^{n+1} d_k Y_k \leq r \right]. \quad (4.23)$$

A necessidade de avaliar esta função aparece ao se estudar algumas medidas da área de análise de desempenho [49], embora este problema também seja estudado em outras áreas do conhecimento, como por exemplo, a teoria assintótica [58] e testes de Hipóteses e Estimção Estatística [55].

Em [56], Dempster e Kleyle apresentaram uma expressão analítica fechada para o caso onde  $a_1 > a_2 > \dots > a_{n+1} = 0$ . Em [57], Weisberg apresentou um algoritmo para o cálculo da combinação linear de um subconjunto das estatísticas de ordem para o caso onde  $a_k \geq 0$  e  $a_{n+1} = 0$ ,  $k = 1, \dots, n + 1$ . O problema com este algoritmo é a necessidade de avaliação recorrente de alguns termos do algoritmo. Uma expressão analítica fechada para o cálculo da combinação linear de um subconjunto das estatísticas de ordem foi obtida por Matsunawa [58]. O problema com esta solução é a complexidade do cálculo de alguns termos da expressão analítica. Em [59], Ramalingam define um algoritmo para o cálculo

desses termos a ser usado antes da expressão de Matsunawa.

Além da complexidade de alguns cálculos, os algoritmos de combinação linear de estatísticas de ordem citados acima apresentam duas características em comum que dificultam as implementações: trabalham com valores positivos e negativos e não existe limite de valor para os termos encontrados durante os cálculos (pode ocorrer *underflow/overflow* e perda de precisão). Nesta seção são apresentados dois novos algoritmos de combinação linear de estatísticas de ordem que possuem a grande vantagem de só trabalharem com números positivos e não maiores que 1.

Para compatibilizar a notação de 4.23 com o que foi visto na seção 4.2.2 (veja equação 4.6), vamos usar a seguinte notação para combinação de estatísticas de ordem no resto deste capítulo:

$$EO_{>}[\vec{r}, \mathbf{k}, r] = P \left[ \sum_{j=1}^L (r_{\xi(j)} - r_{\xi(j+1)}) U_{(n_j)} > r - r_{\xi(L+1)} \right] \quad (4.24)$$

e

$$EO_{\geq}[\vec{r}, \mathbf{k}, r] = P \left[ \sum_{j=1}^L (r_{\xi(j)} - r_{\xi(j+1)}) U_{(n_j)} \geq r - r_{\xi(L+1)} \right], \quad (4.25)$$

onde  $\vec{r}$  é o vetor de recompensas.

### 4.3.1 Algoritmo 1

O objetivo desta seção é apresentar o algoritmo do Teorema 1 (equação 4.35). Para facilitar a compreensão do leitor, iremos primeiro discutir o Lema 1 (equação 4.28) que serve de base para a prova do Teorema 1.

**Definição 1:** Para  $\|\mathbf{k}\| = n + 1$ ,  $k_i > 0$ ,  $r_i \in \mathcal{R}$  e  $r_i > r$ , seja

$$\Theta_i(r, \mathbf{k}) = \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} f_i(r, \mathbf{k}). \quad (4.26)$$

Para  $\|\mathbf{k}\| = 1$ , isto é, existe um único intervalo, temos  $k_i = 1$  para algum  $i$ .

Substituindo 4.9 em 4.26 temos

$$\Theta_i(r, \mathbf{1}_j) = \begin{cases} \frac{1}{0!} \frac{d^{(0)}}{dr_i} (r_i - r)^0 = 1 & \text{para } i = j \\ 0 & \text{para } i \neq j \end{cases} \quad (4.27)$$

**Lema 1:** Para  $\|\mathbf{k}\| = n + 1$  e para qualquer recompensa  $r_g \in \mathcal{R}$  e  $r_l \in \mathcal{R}$ ,  $r_g \geq r$  e  $r_l < r$ , onde  $k_g > 0$  e  $k_l > 0$ , temos

$$\Theta_i(r, \mathbf{k}) = \left( \frac{r_g - r}{r_g - r_l} \right) \Theta_i(r, \mathbf{k} - \mathbf{1}_l) + \left( 1 - \frac{r_g - r}{r_g - r_l} \right) \Theta_i(r, \mathbf{k} - \mathbf{1}_g), \quad k_i > 0. \quad (4.28)$$

**Prova.** Note que  $\|\mathbf{k}\| > 1$  pois  $k_g > 0$  e  $k_l > 0$ . De 4.26 temos que  $r_i > r$  e de 4.28 temos  $r_g \geq r$  e  $r_l < r$ . Portanto, precisamos mostrar que 4.28 é válido para:

a)  $r_g = r$  (e portanto,  $r_g \neq r_i$ ); b)  $r_g = r_i$  (e portanto,  $r_g > r$ ); c)  $r_g \neq r$  e  $r_g \neq r_i$  (e portanto,  $r_g > r$ ).

a) Para  $r_g = r$

Substituindo 4.9 em 4.26 temos

$$\Theta_i(r, \mathbf{k}) = \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} \left\{ \frac{(r_i - r)^n}{\prod_{\substack{j=1 \\ j \neq i}}^{L+1} (r_i - r_j)^{k_j}} \right\}.$$

Podemos então escrever  $\Theta_i(r, \mathbf{k})$  como

$$\Theta_i(r, \mathbf{k}) = \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} \left\{ \frac{(r_i - r)^n}{\prod_{\substack{j=1 \\ j \neq i, g}}^{L+1} \{(r_i - r_j)^{k_j}\} (r_i - r_g)^{k_g}} \right\}$$

$$= \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} \left\{ \frac{(r_i - r)^{n-1}}{\prod_{\substack{j=1 \\ j \neq i, g}}^{L+1} \{(r_i - r_j)^{k_j}\} (r_i - r_g)^{k_g-1}} \left( \frac{r_i - r}{r_i - r_g} \right) \right\}.$$

Como  $r_g = r$ ,

$$\begin{aligned} \Theta_i(r, \mathbf{k}) &= \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} \left\{ \frac{(r_i - r)^{n-1}}{\prod_{\substack{j=1 \\ j \neq i, g}}^{L+1} \{(r_i - r_j)^{k_j}\} (r_i - r_g)^{k_g-1}} \right\} \\ &= \Theta_i(r, \mathbf{k} - \mathbf{1}_g). \end{aligned}$$

Sabemos que  $(r_g - r)/(r_g - r_l) = 0$  pois  $r_g = r$ . Portanto, é fácil ver que a equação 4.28 é verdadeira para qualquer  $r_l \in \mathcal{R}$  quando  $r_g = r$ .

b) Para  $r_g = r_i$

$$\begin{aligned} \Theta_i(r, \mathbf{k}) &= \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} f_i(r, \mathbf{k}) \\ &= \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} \left\{ \left( \frac{r_i - r}{r_i - r_l} \right) f_i(r, \mathbf{k} - \mathbf{1}_l) \right\} \\ &= \frac{1}{(k_i - 1)!} \sum_{m=0}^{k_i-1} \binom{k_i-1}{m} \frac{d^{(m)}}{dr_i} \left( \frac{r_i - r}{r_i - r_l} \right) \frac{d^{(k_i-1-m)}}{dr_i} f_i(r, \mathbf{k} - \mathbf{1}_l) \\ &= \frac{1}{(k_i - 1)!} \left( \frac{r_i - r}{r_i - r_l} \right) \frac{d^{(k_i-1)}}{dr_i} f_i(r, \mathbf{k} - \mathbf{1}_l) + \\ &\quad \frac{1}{(k_i - 1)!} \sum_{m=1}^{k_i-1} \frac{(k_i - 1)!}{m!(k_i - 1 - m)!} \frac{d^{(m)}}{dr_i} \left( \frac{r_i - r}{r_i - r_l} \right) \frac{d^{(k_i-1-m)}}{dr_i} f_i(r, \mathbf{k} - \mathbf{1}_l). \end{aligned} \tag{4.29}$$

Seja

$$x_i[m, r, r_l] = \frac{d^{(m)}}{dr_i} \left( \frac{r_i - r}{r_i - r_l} \right).$$

Podemos escrever  $x_i[m, r, r_l]$  da seguinte forma:

$$\begin{aligned} x_i[m, r, r_l] &= \frac{d^{(m)}}{dr_i} \left\{ (r_i - r) \left( \frac{1}{r_i - r_l} \right) \right\} \\ &= \sum_{l=0}^m \binom{m}{l} \frac{d^{(l)}}{dr_i} (r_i - r) \frac{d^{(m-l)}}{dr_i} \left( \frac{1}{r_i - r_l} \right) \\ &= (r_i - r) \frac{d^{(m)}}{dr_i} \left( \frac{1}{r_i - r_l} \right) + \sum_{l=1}^m \binom{m}{l} \frac{d^{(l)}}{dr_i} (r_i - r) \frac{d^{(m-l)}}{dr_i} \left( \frac{1}{r_i - r_l} \right). \end{aligned}$$

É fácil notar que

$$\frac{d^{(m)}}{dr_i} \left( \frac{1}{r_i - r_l} \right) = \frac{-m}{r_i - r_l} \frac{d^{(m-1)}}{dr_i} \left( \frac{1}{r_i - r_l} \right)$$

e

$$\frac{d^{(l)}}{dr_i} (r_i - r) = \begin{cases} 1 & \text{se } l = 1 \\ 0 & \text{nos outros casos} \end{cases}$$

Portanto,

$$\begin{aligned} x_i[m, r, r_l] &= (r_i - r) \frac{-m}{r_i - r_l} \frac{d^{(m-1)}}{dr_i} \left( \frac{1}{r_i - r_l} \right) + \binom{m}{1} \frac{d^{(1)}}{dr_i} (r_i - r) \frac{d^{(m-1)}}{dr_i} \left( \frac{1}{r_i - r_l} \right) \\ &= (-m) \left( \frac{r_i - r}{r_i - r_l} \right) \frac{d^{(m-1)}}{dr_i} \left( \frac{1}{r_i - r_l} \right) + m \frac{d^{(m-1)}}{dr_i} \left( \frac{1}{r_i - r_l} \right) \\ &= m \left( 1 - \frac{r_i - r}{r_i - r_l} \right) \frac{d^{(m-1)}}{dr_i} \left( \frac{1}{r_i - r_l} \right). \end{aligned}$$

Resumindo temos

$$x_i[m, r, r_l] = \frac{d^{(m)}}{dr_i} \left( \frac{r_i - r}{r_i - r_l} \right) = m \left( 1 - \frac{r_i - r}{r_i - r_l} \right) \frac{d^{(m-1)}}{dr_i} \left( \frac{1}{r_i - r_l} \right). \quad (4.30)$$

Substituindo a expressão encontrada para  $x_i[m, r, r_l]$  em 4.29 temos

$$\begin{aligned} \Theta_i(r, \mathbf{k}) &= \left( \frac{r_i - r}{r_i - r_l} \right) \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} f_i(r, \mathbf{k} - \mathbf{1}_l) + \\ &\frac{1}{(k_i - 1)!} \sum_{m=1}^{k_i-1} \frac{(k_i - 1)!}{m!(k_i - 1 - m)!} m \left( 1 - \frac{r_i - r}{r_i - r_l} \right) \frac{d^{(m-1)}}{dr_i} \frac{1}{(r_i - r_l)} \times \\ &\frac{d^{(k_i-1-m)}}{dr_i} f_i(r, \mathbf{k} - \mathbf{1}_l) \\ &= \left( \frac{r_i - r}{r_i - r_l} \right) \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} f_i(r, \mathbf{k} - \mathbf{1}_l) + \\ &\left( 1 - \frac{r_i - r}{r_i - r_l} \right) \frac{1}{(k_i - 2)!} \sum_{m=1}^{k_i-1} \frac{(k_i - 2)!}{(m - 1)!(k_i - 1 - m)!} \frac{d^{(m-1)}}{dr_i} \frac{1}{(r_i - r_l)} \times \\ &\frac{d^{(k_i-1-m)}}{dr_i} f_i(r, \mathbf{k} - \mathbf{1}_l) \\ &= \left( \frac{r_i - r}{r_i - r_l} \right) \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} f_i(r, \mathbf{k} - \mathbf{1}_l) + \\ &\left( 1 - \frac{r_i - r}{r_i - r_l} \right) \frac{1}{(k_i - 2)!} \frac{d^{(k_i-2)}}{dr_i} \left\{ \frac{1}{(r_i - r_l)} f_i(r, \mathbf{k} - \mathbf{1}_l) \right\}. \end{aligned} \quad (4.31)$$

A partir da equação 4.11 temos que

$$f_i(r, \mathbf{k}) = (r_i - r) f_i(r, \mathbf{k} - \mathbf{1}_i) \text{ para } l = i$$

e

$$f_i(r, \mathbf{k}) = \frac{(r_i - r)}{(r_i - r_l)} f_i(r, \mathbf{k} - \mathbf{1}_l) \text{ para } l \neq i,$$

portanto,

$$\frac{1}{(r_i - r_l)} f_i(r, \mathbf{k} - \mathbf{1}_l) = f_i(r, \mathbf{k} - \mathbf{1}_i). \quad (4.32)$$

Substituindo 4.32 em 4.31,

$$\begin{aligned} \Theta_i(r, \mathbf{k}) &= \left( \frac{r_i - r}{r_i - r_l} \right) \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} f_i(r, \mathbf{k} - \mathbf{1}_l) + \\ &\quad \left( 1 - \frac{r_i - r}{r_i - r_l} \right) \frac{1}{(k_i - 2)!} \frac{d^{(k_i-2)}}{dr_i} f_i(r, \mathbf{k} - \mathbf{1}_i) \\ &= \left( \frac{r_i - r}{r_i - r_l} \right) \Theta_i(r, \mathbf{k} - \mathbf{1}_l) + \left( 1 - \frac{r_i - r}{r_i - r_l} \right) \Theta_i(r, \mathbf{k} - \mathbf{1}_i), \end{aligned}$$

onde a última igualdade vem da definição de  $\Theta_i(r, \mathbf{k})$  em 4.26. Note que de 4.11 e de 4.26, quando  $k_i = 1$  e  $r_g = r_i$ , temos

$$\Theta_i(r, \mathbf{k}) = \left( \frac{r_g - r}{r_g - r_l} \right) \Theta_i(r, \mathbf{k} - \mathbf{1}_l).$$

c) Para  $r_g \neq r$  e  $r_g \neq r_i$

$$\begin{aligned} \Theta_i(r, \mathbf{k}) &= \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} f_i(r, \mathbf{k}) \\ &= \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} \left\{ \left( \frac{r_i - r}{r_i - r_l} \right) f_i(r, \mathbf{k} - \mathbf{1}_l) \right\} \\ &= \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} \left\{ \left( \frac{r_i - r}{r_i - r_l} + \frac{r_g - r}{r_g - r_l} - \frac{r_g - r}{r_g - r_l} \right) f_i(r, \mathbf{k} - \mathbf{1}_l) \right\} \\ &= \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} \left\{ \left( \frac{r_g - r}{r_g - r_l} \right) f_i(r, \mathbf{k} - \mathbf{1}_l) \right\} + \\ &\quad \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} \left\{ \left( \frac{r_i - r}{r_i - r_l} - \frac{r_g - r}{r_g - r_l} \right) f_i(r, \mathbf{k} - \mathbf{1}_l) \right\} \\ &= \left( \frac{r_g - r}{r_g - r_l} \right) \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} f_i(r, \mathbf{k} - \mathbf{1}_l) + \\ &\quad \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} \left\{ \left( \frac{r - r_l}{r_g - r_l} \times \frac{r_i - r_g}{r_i - r_l} \right) f_i(r, \mathbf{k} - \mathbf{1}_l) \right\} \end{aligned}$$

$$= \left( \frac{r_g - r}{r_g - r_l} \right) \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} f_i(r, \mathbf{k} - \mathbf{1}_l) + \left( \frac{r - r_l}{r_g - r_l} \right) \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} \left\{ \left( \frac{r_i - r_g}{r_i - r_l} \right) f_i(r, \mathbf{k} - \mathbf{1}_l) \right\}.$$

Da definição de  $f_i(r, \mathbf{k})$  (equação 4.9) e como  $r_i \neq r_g$  podemos escrever:

$$\left( \frac{r_i - r_g}{r_i - r_l} \right) f_i(r, \mathbf{k} - \mathbf{1}_l) = f_i(r, \mathbf{k} - \mathbf{1}_g).$$

Portanto,

$$\begin{aligned} \Theta_i(r, \mathbf{k}) &= \left( \frac{r_g - r}{r_g - r_l} \right) \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} f_i(r, \mathbf{k} - \mathbf{1}_l) + \\ &\quad \left( 1 - \frac{r_g - r}{r_g - r_l} \right) \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} f_i(r, \mathbf{k} - \mathbf{1}_g) \\ &= \left( \frac{r_g - r}{r_g - r_l} \right) \Theta_i(r, \mathbf{k} - \mathbf{1}_l) + \left( 1 - \frac{r_g - r}{r_g - r_l} \right) \Theta_i(r, \mathbf{k} - \mathbf{1}_g). \end{aligned}$$

□

**Definição 2:** Para  $\|\mathbf{k}\| = n + 1$ ,

$$\begin{aligned} \Theta(r, \mathbf{k}) &= \sum_{i:r_i > r} \Theta_i(r, \mathbf{k}) \\ &= \sum_{i:r_i > r} \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} f_i(r, \mathbf{k}) \\ &= EO_{>}[\vec{r}, \mathbf{k}, r]. \end{aligned} \tag{4.33}$$

Sem perda de generalidade, estamos supondo que as variáveis aleatórias  $U_k$  estão uniformemente distribuídas no intervalo  $(0, 1)$ , onde  $Y_k = (U_{(k)} - U_{(k-1)})$  e  $k = 1, \dots, n$ . E portanto,

$$\frac{1}{t} P \left[ \sum_{k=1}^{n+1} d_k Y_k > r \right] = P \left[ \sum_{k=1}^{n+1} d_k Y_k > r \right] = EO_{>}[\vec{r}, \mathbf{k}, r]$$

e

$$\frac{1}{t} P \left[ \sum_{k=1}^{n+1} d_k Y_k \geq r \right] = P \left[ \sum_{k=1}^{n+1} d_k Y_k \geq r \right] = EO_{\geq}[\vec{r}, \mathbf{k}, r].$$



Além disso, temos (veja equação 4.27)

$$\Theta(r, \mathbf{1}_j) = \sum_{i:r_i > r} \Theta_i(r, \mathbf{1}_j) = \begin{cases} 1 & \text{se } r_j > r \\ 0 & \text{nos outros casos} \end{cases} \quad (4.34)$$

**Teorema 1:** Para qualquer recompensa  $r_g \in \mathcal{R}$  e  $r_l \in \mathcal{R}$ ,  $r_g \geq r$  e  $r_l < r$ , onde  $k_g > 0$  e  $k_l > 0$ , temos

$$\Theta(r, \mathbf{k}) = \left( \frac{r_g - r}{r_g - r_l} \right) \Theta(r, \mathbf{k} - \mathbf{1}_l) + \left( 1 - \frac{r_g - r}{r_g - r_l} \right) \Theta(r, \mathbf{k} - \mathbf{1}_g), \quad k_i > 0. \quad (4.35)$$

**Prova.** Como o Lema 1 (4.28) é válido com a condição de  $r_i > r$  e  $k_i > 0$ ,

$$\begin{aligned} \Theta(r, \mathbf{k}) &= \sum_{i:r_i > r} \Theta_i(r, \mathbf{k}) \\ &= \sum_{i:r_i > r} \left\{ \left( \frac{r_g - r}{r_g - r_l} \right) \Theta_i(r, \mathbf{k} - \mathbf{1}_l) + \left( 1 - \frac{r_g - r}{r_g - r_l} \right) \Theta_i(r, \mathbf{k} - \mathbf{1}_g) \right\} \\ &= \sum_{i:r_i > r} \left( \frac{r_g - r}{r_g - r_l} \right) \Theta_i(r, \mathbf{k} - \mathbf{1}_l) + \sum_{i:r_i > r} \left( 1 - \frac{r_g - r}{r_g - r_l} \right) \Theta_i(r, \mathbf{k} - \mathbf{1}_g) \\ &= \left( \frac{r_g - r}{r_g - r_l} \right) \sum_{i:r_i > r} \Theta_i(r, \mathbf{k} - \mathbf{1}_l) + \left( 1 - \frac{r_g - r}{r_g - r_l} \right) \sum_{i:r_i > r} \Theta_i(r, \mathbf{k} - \mathbf{1}_g) \\ &= \left( \frac{r_g - r}{r_g - r_l} \right) \Theta(r, \mathbf{k} - \mathbf{1}_l) + \left( 1 - \frac{r_g - r}{r_g - r_l} \right) \Theta(r, \mathbf{k} - \mathbf{1}_g). \end{aligned}$$

□

O custo computacional do algoritmo apresentado no Teorema 1 é  $O(N)$ , onde  $\|\mathbf{k}\| = N + 1$ . Além disso, note que no passo  $N$  do algoritmo só precisamos conhecer os termos calculados no passo  $N - 1$ .

### 4.3.2 Algoritmo 2

Como já discutimos anteriormente o nosso objetivo é calcular  $P[ACR(t) > r]$ . Embora o Teorema 1 nos forneça um algoritmo simples, eficiente e numericamente estável para o cálculo de combinação linear de estatísticas de ordem, não foi possível usá-lo para determinar  $P[ACR(t) > r]$  usando a metodologia de [49] de forma a obtermos uma interpretação probabilística. Nesta seção propomos outro algoritmo para combinação linear de estatísticas de ordem que, como veremos na seção 4.4, pode ser utilizado no cálculo de  $P[ACR(t) > r]$ . Este algoritmo é apresentado no Teorema 2 (equação 4.61). Os Lemas 2, 3 e 4 abaixo servirão de base para a prova do Teorema 2.

**Definição 3:** Para  $\|\mathbf{k}\| = n + 1$ ,  $m = 0, \dots, n$ ,  $j = 1, \dots, L$ ,  $k_i > 0$  e  $r_i \geq r_j$ , seja

$$\Omega_i(m, r_j, \mathbf{k}) = \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} \left\{ \left( \frac{r_i - r_j}{r_i - r_{j+1}} \right)^m f_i(r_{j+1}, \mathbf{k}) \right\}. \quad (4.36)$$

Antes de apresentar o Lema 2, é importante fazermos uma observação em relação ao valor de  $\Omega_i(n, r_j, \mathbf{k})$ . De 4.36 temos

$$\begin{aligned} \Omega_i(n, r_j, \mathbf{k}) &= \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} \left\{ (r_i - r_j)^n \times \frac{1}{(r_i - r_{j+1})^n} f_i(r_{j+1}, \mathbf{k}) \right\} \\ &= \frac{1}{(k_i - 1)!} \sum_{l=0}^{k_i-1} \binom{k_i - 1}{l} \frac{d^{(l)}}{dr_i} \left\{ (r_i - r_j)^n \right\} \times \\ &\quad \frac{d^{(k_i-1-l)}}{dr_i} \left\{ \frac{1}{(r_i - r_{j+1})^n} f_i(r_{j+1}, \mathbf{k}) \right\}. \end{aligned} \quad (4.37)$$

Para  $l = 1, \dots, k_{i-1}$ , é fácil ver que

$$\frac{d^{(l)}}{dr_i} (r_i - r_j)^n = n(n-1) \dots (n-l+1) (r_i - r_j)^{n-l} \quad (4.38)$$

Portanto, para  $r_j = r_i$ , temos

$$\frac{d^{(l)}}{dr_i}(r_i - r_j)^n \begin{cases} 0 & \text{se } l < n \\ n! & \text{se } l = n \end{cases} \quad (4.39)$$

Sabemos que  $l = 0, \dots, k_i - 1$ . Portanto  $l < n$  a menos que  $k_i = n + 1$ , ou seja, todos os intervalos de  $\mathbf{k}$  têm recompensa  $r_i$ . Para  $k_i = n + 1$ , temos (equação 4.9)

$$f_i(r_{j+1}, \mathbf{k}) = (r_i - r_{j+1})^n. \quad (4.40)$$

Substituindo 4.38, 4.39 e 4.40 em 4.37,

$$\begin{aligned} \Omega_i(n, r_j, \mathbf{k}) &= \frac{1}{(k_i - 1)!} \{(r_i - r_j)^n\} \frac{d^{(k_i-1)}}{dr_i} \left\{ \frac{1}{(r_i - r_{j+1})^n} f_i(r_{j+1}, \mathbf{k}) \right\} + \\ &\sum_{l=1}^{k_i-2} \binom{k_i-1}{l} \frac{d^{(l)}}{dr_i} \{(r_i - r_j)^n\} \times \frac{d^{(k_i-1-l)}}{dr_i} \left\{ \frac{1}{(r_i - r_{j+1})^n} f_i(r_{j+1}, \mathbf{k}) \right\} + \\ &\frac{d^{(k_i-1)}}{dr_i} \{(r_i - r_j)^n\} \times \left\{ \frac{1}{(r_i - r_{j+1})^n} f_i(r_{j+1}, \mathbf{k}) \right\}. \end{aligned} \quad (4.41)$$

Para  $i = j$  ( $r_i = r_j$ ),  $\mathbf{k} = \mathbf{1}_i \times (n + 1)$  ( $k_i = n + 1$ )

$$\begin{aligned} \Omega_i(n, r_i, \mathbf{1}_i) &= \frac{1}{(n)!} \frac{d^{(n)}}{dr_i} \{(r_i - r_j)^n\} \frac{d^{(0)}}{dr_i} \left\{ \frac{1}{(r_i - r_{j+1})^n} f_i(r_{j+1}, \mathbf{k}) \right\} \\ &= \frac{1}{(n)!} \{n! \times 1\} \left\{ \frac{1}{(r_i - r_{j+1})^n} (r_i - r_{j+1})^n \right\} \\ &= 1. \end{aligned}$$

Além disso, se  $k_i = 0$  pela definição 4.10,

$$\Omega_i(n, r_j, \mathbf{k}) = 0 \quad \text{para } n \geq 0, \forall j \text{ tal que } r_i > r_j$$

$$\Omega_i(0, r_j, \mathbf{1}_i) = f_i(r_{j+1}, \mathbf{1}_i) = 1 \quad \text{para } r_i > r_j.$$

Resumindo,

$$\Omega_i(n, r_i, \mathbf{k}) = \begin{cases} 1 & \text{se } n = 0, \forall j \text{ tal que } r_i > r_j, \mathbf{k} = \mathbf{1}_i \\ 1 & \text{se } n > 0, i = j, k_i = n + 1 \\ 0 & \text{nos outros casos} \end{cases} \quad (4.42)$$

**Lema 2:** Para  $\|\mathbf{k}\| = n + 1$ ,  $j = 1, \dots, L$  e  $m = 1, \dots, n$  e para qualquer recompensa  $r_g \in \mathcal{R}$ ,  $r_j \in \mathcal{R}$  e  $r_{j+1} \in \mathcal{R}$ , onde  $r_{j+1} < r_j \leq r_g$  e  $k_g > 0$ , temos

$$\Omega_i(m, r_j, \mathbf{k}) = \left( \frac{r_g - r_j}{r_g - r_{j+1}} \right) \Omega_i(m - 1, r_j, \mathbf{k}) + \left( 1 - \frac{r_g - r_j}{r_g - r_{j+1}} \right) \Omega_i(m - 1, r_j, \mathbf{k} - \mathbf{1}_g). \quad (4.43)$$

**Prova.** A equação 4.36 é definida para  $r_i \geq r_j$  e temos que  $r_g \geq r_j$ . Precisamos mostrar que 4.43 é válida para: a)  $r_g = r_j$  e  $r_g \neq r_i$ ; b)  $r_g = r_i$  ( $r_g = r_j$  ou  $r_g \neq r_j$ ); c)  $r_g \neq r_j$  e  $r_g \neq r_i$ . Antes de apresentarmos a prova do Lema 2, note que podemos escrever a equação 4.9, para  $r_i \geq r_j$  e  $r_j = r$ , na seguinte forma

$$\begin{aligned} f_i(r_j, \mathbf{k}) &= \left( \frac{r_i - r_j}{r_i - r_{j+1}} \right)^n \frac{(r_i - r_{j+1})^n}{\prod_{\substack{l=1 \\ l \neq i}}^{L+1} (r_i - r_l)^{k_l}} \\ &= \left( \frac{r_i - r_j}{r_i - r_{j+1}} \right)^n f_i(r_{j+1}, \mathbf{k}). \end{aligned} \quad (4.44)$$

a) Para  $r_g = r_j$  e  $r_g \neq r_i$

De 4.11, como  $i \neq g$

$$f_i(r_{j+1}, \mathbf{k}) = \left( \frac{r_i - r_{j+1}}{r_i - r_g} \right) f_i(r_{j+1}, \mathbf{k} - \mathbf{1}_g). \quad (4.45)$$

De 4.36 e 4.45 e como  $r_g = r_j$ ,

$$\begin{aligned}
\Omega_i(m, r_j, \mathbf{k}) &= \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} \left\{ \left( \frac{r_i - r_j}{r_i - r_{j+1}} \right)^m f_i(r_{j+1}, \mathbf{k}) \right\} \\
&= \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} \left\{ \left( \frac{r_i - r_j}{r_i - r_{j+1}} \right)^m \left( \frac{r_i - r_{j+1}}{r_i - r_g} \right) f_i(r_{j+1}, \mathbf{k} - \mathbf{1}_g) \right\} \\
&= \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} \left\{ \left( \frac{r_i - r_j}{r_i - r_{j+1}} \right)^{(m-1)} f_i(r_{j+1}, \mathbf{k} - \mathbf{1}_g) \right\} \\
&= \Omega_i(m - 1, r_j, \mathbf{k} - \mathbf{1}_g).
\end{aligned}$$

Como  $r_g = r_j$ , temos  $(r_g - r_j)/(r_g - r_{j+1}) = 0$  e portanto, é fácil ver que a equação 4.43 é verdadeira.

b) Para  $r_g = r_i$

$$\begin{aligned}
\Omega_i(m, r_j, \mathbf{k}) &= \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} \left\{ \left( \frac{r_i - r_j}{r_i - r_{j+1}} \right)^m f_i(r_{j+1}, \mathbf{k}) \right\} \\
&= \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} \left\{ \left( \frac{r_i - r_j}{r_i - r_{j+1}} \right) \left( \frac{r_i - r_j}{r_i - r_{j+1}} \right)^{(m-1)} f_i(r_{j+1}, \mathbf{k}) \right\} \\
&= \frac{1}{(k_i - 1)!} \sum_{l=0}^{k_i-1} \binom{k_i-1}{l} \frac{d^{(l)}}{dr_i} \left\{ \frac{r_i - r_j}{r_i - r_{j+1}} \right\} \times \\
&\quad \frac{d^{(k_i-1-l)}}{dr_i} \left\{ \left( \frac{r_i - r_j}{r_i - r_{j+1}} \right)^{(m-1)} f_i(r_{j+1}, \mathbf{k}) \right\} \\
&= \frac{1}{(k_i - 1)!} \left( \frac{r_i - r_j}{r_i - r_{j+1}} \right) \frac{d^{(k_i-1)}}{dr_i} \left\{ \left( \frac{r_i - r_j}{r_i - r_{j+1}} \right)^{(m-1)} f_i(r_{j+1}, \mathbf{k}) \right\} + \\
&\quad \frac{1}{(k_i - 1)!} \sum_{l=1}^{k_i-1} \binom{k_i-1}{l} \frac{d^{(l)}}{dr_i} \left\{ \frac{r_i - r_j}{r_i - r_{j+1}} \right\} \times \\
&\quad \frac{d^{(k_i-1-l)}}{dr_i} \left\{ \left( \frac{r_i - r_j}{r_i - r_{j+1}} \right)^{(m-1)} f_i(r_{j+1}, \mathbf{k}) \right\} \\
&= \left( \frac{r_i - r_j}{r_i - r_{j+1}} \right) \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} \left\{ \left( \frac{r_i - r_j}{r_i - r_{j+1}} \right)^{(m-1)} f_i(r_{j+1}, \mathbf{k}) \right\} +
\end{aligned}$$

$$\frac{1}{(k_i - 1)!} \sum_{l=1}^{k_i-1} \frac{(k_i - 1)!}{l!(k_i - 1 - l)!} \frac{d^{(l)}}{dr_i} \left\{ \frac{r_i - r_j}{r_i - r_{j+1}} \right\} \times \frac{d^{(k_i-1-l)}}{dr_i} \left\{ \left( \frac{r_i - r_j}{r_i - r_{j+1}} \right)^{(m-1)} f_i(r_{j+1}, \mathbf{k}) \right\}. \quad (4.46)$$

Usando a equação 4.30 temos

$$x_i[l, r_j, r_{j+1}] = \frac{d^{(l)}}{dr_i} \left( \frac{r_i - r_j}{r_i - r_{j+1}} \right) = l \left( 1 - \frac{r_i - r_j}{r_i - r_{j+1}} \right) \frac{d^{(l-1)}}{dr_i} \left( \frac{1}{r_i - r_{j+1}} \right). \quad (4.47)$$

Substituindo 4.47 em 4.46

$$\begin{aligned} \Omega_i(m, r_j, \mathbf{k}) &= \left( \frac{r_i - r_j}{r_i - r_{j+1}} \right) \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} \left\{ \left( \frac{r_i - r_j}{r_i - r_{j+1}} \right)^{(m-1)} f_i(r_{j+1}, \mathbf{k}) \right\} + \\ &\frac{1}{(k_i - 1)!} \sum_{l=1}^{k_i-1} \frac{(k_i - 1)!}{l!(k_i - 1 - l)!} l \left( 1 - \frac{r_i - r_j}{r_i - r_{j+1}} \right) \frac{d^{(l-1)}}{dr_i} \left\{ \frac{1}{r_i - r_{j+1}} \right\} \times \\ &\frac{d^{(k_i-1-l)}}{dr_i} \left\{ \left( \frac{r_i - r_j}{r_i - r_{j+1}} \right)^{(m-1)} f_i(r_{j+1}, \mathbf{k}) \right\} \\ &= \left( \frac{r_i - r_j}{r_i - r_{j+1}} \right) \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} \left\{ \left( \frac{r_i - r_j}{r_i - r_{j+1}} \right)^{(m-1)} f_i(r_{j+1}, \mathbf{k}) \right\} + \\ &\left( 1 - \frac{r_i - r_j}{r_i - r_{j+1}} \right) \frac{1}{(k_i - 2)!} \sum_{l=1}^{k_i-1} \frac{(k_i - 2)!}{(l - 1)!(k_i - 1 - l)!} \frac{d^{(l-1)}}{dr_i} \left\{ \frac{1}{r_i - r_{j+1}} \right\} \times \\ &\frac{d^{(k_i-1-l)}}{dr_i} \left\{ \left( \frac{r_i - r_j}{r_i - r_{j+1}} \right)^{(m-1)} f_i(r_{j+1}, \mathbf{k}) \right\} \\ &= \left( \frac{r_i - r_j}{r_i - r_{j+1}} \right) \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} \left\{ \left( \frac{r_i - r_j}{r_i - r_{j+1}} \right)^{(m-1)} f_i(r_{j+1}, \mathbf{k}) \right\} + \\ &\left( 1 - \frac{r_i - r_j}{r_i - r_{j+1}} \right) \frac{1}{(k_i - 2)!} \frac{d^{(k_i-2)}}{dr_i} \left\{ \frac{1}{r_i - r_{j+1}} \left( \frac{r_i - r_j}{r_i - r_{j+1}} \right)^{(m-1)} f_i(r_{j+1}, \mathbf{k}) \right\}. \end{aligned}$$

De 4.11,

$$\begin{aligned}\Omega_i(m, r_j, \mathbf{k}) &= \left( \frac{r_i - r_j}{r_i - r_{j+1}} \right) \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} \left\{ \left( \frac{r_i - r_j}{r_i - r_{j+1}} \right)^{(m-1)} f_i(r_{j+1}, \mathbf{k}) \right\} + \\ &\quad \left( 1 - \frac{r_i - r_j}{r_i - r_{j+1}} \right) \frac{1}{(k_i - 2)!} \frac{d^{(k_i-2)}}{dr_i} \left\{ \left( \frac{r_i - r_j}{r_i - r_{j+1}} \right)^{(m-1)} \times \right. \\ &\quad \left. f_i(r_{j+1}, \mathbf{k} - \mathbf{1}_i) \right\} \quad (4.48)\end{aligned}$$

$$\begin{aligned}&= \left( \frac{r_i - r_j}{r_i - r_{j+1}} \right) \Omega_i(m - 1, r_j, \mathbf{k}) + \\ &\quad \left( 1 - \frac{r_i - r_j}{r_i - r_{j+1}} \right) \Omega_i(m - 1, r_j, \mathbf{k} - \mathbf{1}_i). \quad (4.49)\end{aligned}$$

Note que a prova acima é válida para  $k_i = 1$ . De 4.42

$$\Omega_i(m, r_j, \mathbf{k}) = \left( \frac{r_i - r_j}{r_i - r_{j+1}} \right) \Omega_i(m - 1, r_j, \mathbf{k}).$$

c) Para  $r_g \neq r_j$  e  $r_g \neq r_i$

$$\begin{aligned}\Omega_i(m, r_j, \mathbf{k}) &= \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} \left\{ \left( \frac{r_i - r_j}{r_i - r_{j+1}} \right)^m f_i(r_{j+1}, \mathbf{k}) \right\} \\ &= \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} \left\{ \left( \frac{r_i - r_j}{r_i - r_{j+1}} \right) \left( \frac{r_i - r_j}{r_i - r_{j+1}} \right)^{(m-1)} f_i(r_{j+1}, \mathbf{k}) \right\} \\ &= \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} \left\{ \left( \frac{r_i - r_j}{r_i - r_{j+1}} + \frac{r_g - r_j}{r_g - r_{j+1}} - \frac{r_g - r_j}{r_g - r_{j+1}} \right) \times \right. \\ &\quad \left. \left( \frac{r_i - r_j}{r_i - r_{j+1}} \right)^{(m-1)} f_i(r_{j+1}, \mathbf{k}) \right\} \\ &= \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} \left\{ \left( \frac{r_g - r_j}{r_g - r_{j+1}} \right) \left( \frac{r_i - r_j}{r_i - r_{j+1}} \right)^{(m-1)} f_i(r_{j+1}, \mathbf{k}) \right\} + \\ &\quad \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} \left\{ \left( \frac{r_i - r_j}{r_i - r_{j+1}} - \frac{r_g - r_j}{r_g - r_{j+1}} \right) \times \right. \\ &\quad \left. \left( \frac{r_i - r_j}{r_i - r_{j+1}} \right)^{(m-1)} f_i(r_{j+1}, \mathbf{k}) \right\} \\ &= \left( \frac{r_g - r_j}{r_g - r_{j+1}} \right) \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} \left\{ \left( \frac{r_i - r_j}{r_i - r_{j+1}} \right)^{(m-1)} f_i(r_{j+1}, \mathbf{k}) \right\} + \\ &\quad \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} \left\{ \left( \frac{r_i - r_g}{r_i - r_{j+1}} \times \frac{r_j - r_{j+1}}{r_g - r_{j+1}} \right) \times \right.\end{aligned}$$

$$\left. \left( \frac{r_i - r_j}{r_i - r_{j+1}} \right)^{(m-1)} f_i(r_{j+1}, \mathbf{k}) \right\}.$$

De 4.11,

$$\left( \frac{r_i - r_g}{r_i - r_{j+1}} \right) f_i(r_{j+1}, \mathbf{k}) = f_i(r_{j+1}, \mathbf{k} - \mathbf{1}_g).$$

Portanto,

$$\begin{aligned} \Omega_i(m, r_j, \mathbf{k}) &= \left( \frac{r_g - r_j}{r_g - r_{j+1}} \right) \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} \left\{ \left( \frac{r_i - r_j}{r_i - r_{j+1}} \right)^{(m-1)} f_i(r_{j+1}, \mathbf{k}) \right\} + \\ &\quad \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} \left\{ \left( \frac{r_j - r_{j+1}}{r_g - r_{j+1}} \right) \left( \frac{r_i - r_j}{r_i - r_{j+1}} \right)^{(m-1)} f_i(r_{j+1}, \mathbf{k} - \mathbf{1}_g) \right\} \\ &= \left( \frac{r_g - r_j}{r_g - r_{j+1}} \right) \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} \left\{ \left( \frac{r_i - r_j}{r_i - r_{j+1}} \right)^{(m-1)} f_i(r_{j+1}, \mathbf{k}) \right\} + \\ &\quad \left( \frac{r_j - r_{j+1}}{r_g - r_{j+1}} \right) \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} \left\{ \left( \frac{r_i - r_j}{r_i - r_{j+1}} \right)^{(m-1)} f_i(r_{j+1}, \mathbf{k} - \mathbf{1}_g) \right\} \\ &= \left( \frac{r_g - r_j}{r_g - r_{j+1}} \right) \Omega_i(m - 1, r_j, \mathbf{k}) + \\ &\quad \left( 1 - \frac{r_g - r_j}{r_g - r_{j+1}} \right) \Omega_i(m - 1, r_j, \mathbf{k} - \mathbf{1}_g). \end{aligned}$$

□

**Lema 3:** Para  $\|\mathbf{k}\| = n + 1$ ,  $j = 1, \dots, L$  e  $m = 0, \dots, n - 1$  e para qualquer recompensa  $r_g \in \mathcal{R}$ ,  $r_j \in \mathcal{R}$  e  $r_{j+1} \in \mathcal{R}$ , onde  $r_g \leq r_{j+1} < r_j$  e  $k_g > 0$ , temos

$$\begin{aligned} \Omega_i(m, r_j, \mathbf{k}) &= \\ &\quad \left( \frac{r_{j+1} - r_g}{r_j - r_g} \right) \Omega_i(m + 1, r_j, \mathbf{k}) + \left( 1 - \frac{r_{j+1} - r_g}{r_j - r_g} \right) \Omega_i(m, r_j, \mathbf{k} - \mathbf{1}_g). \end{aligned} \quad (4.50)$$

**Prova.**

$$\Omega_i(m, r_j, \mathbf{k}) = \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} \left\{ \left( \frac{r_i - r_j}{r_i - r_{j+1}} \right)^m f_i(r_{j+1}, \mathbf{k}) \right\}$$



$$\begin{aligned}
&= \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} \left\{ \left( \frac{r_i - r_{j+1}}{r_i - r_j} \right) \left( \frac{r_i - r_j}{r_i - r_{j+1}} \right)^{(m+1)} f_i(r_{j+1}, \mathbf{k}) \right\} \\
&= \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} \left\{ \left( \frac{r_i - r_{j+1}}{r_i - r_j} + \frac{r_{j+1} - r_g}{r_j - r_g} - \frac{r_{j+1} - r_g}{r_j - r_g} \right) \times \right. \\
&\quad \left. \left( \frac{r_i - r_j}{r_i - r_{j+1}} \right)^{(m+1)} f_i(r_{j+1}, \mathbf{k}) \right\} \\
&= \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} \left\{ \left( \frac{r_{j+1} - r_g}{r_j - r_g} \right) \left( \frac{r_i - r_j}{r_i - r_{j+1}} \right)^{(m+1)} f_i(r_{j+1}, \mathbf{k}) \right\} + \\
&\quad \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} \left\{ \left( \frac{r_i - r_{j+1}}{r_i - r_j} - \frac{r_{j+1} - r_g}{r_j - r_g} \right) \times \right. \\
&\quad \left. \left( \frac{r_i - r_j}{r_i - r_{j+1}} \right)^{(m+1)} f_i(r_{j+1}, \mathbf{k}) \right\} \\
&= \left( \frac{r_{j+1} - r_g}{r_j - r_g} \right) \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} \left\{ \left( \frac{r_i - r_j}{r_i - r_{j+1}} \right)^{(m+1)} f_i(r_{j+1}, \mathbf{k}) \right\} + \\
&\quad \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} \left\{ \left( \frac{r_i - r_g}{r_i - r_j} \times \frac{r_j - r_{j+1}}{r_j - r_g} \right) \times \right. \\
&\quad \left. \left( \frac{r_i - r_j}{r_i - r_{j+1}} \right)^{(m+1)} f_i(r_{j+1}, \mathbf{k}) \right\}
\end{aligned}$$

De 4.11,

$$f_i(r_{j+1}, \mathbf{k}) = \left( \frac{r_i - r_{j+1}}{r_i - r_g} \right) f_i(r_{j+1}, \mathbf{k} - \mathbf{1}_g).$$

Portanto,

$$\begin{aligned}
\Omega_i(m, r_j, \mathbf{k}) &= \left( \frac{r_{j+1} - r_g}{r_j - r_g} \right) \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} \left\{ \left( \frac{r_i - r_j}{r_i - r_{j+1}} \right)^{(m+1)} f_i(r_{j+1}, \mathbf{k}) \right\} + \\
&\quad \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} \left\{ \left( \frac{r_i - r_g}{r_i - r_j} \times \frac{r_j - r_{j+1}}{r_j - r_g} \right) \left( \frac{r_i - r_j}{r_i - r_{j+1}} \right)^{(m+1)} \right. \\
&\quad \left. \left( \frac{r_i - r_{j+1}}{r_i - r_g} \right) f_i(r_{j+1}, \mathbf{k} - \mathbf{1}_g) \right\} \\
&= \left( \frac{r_{j+1} - r_g}{r_j - r_g} \right) \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} \left\{ \left( \frac{r_i - r_j}{r_i - r_{j+1}} \right)^{(m+1)} f_i(r_{j+1}, \mathbf{k}) \right\} +
\end{aligned}$$

$$\begin{aligned}
& \left( \frac{r_j - r_{j+1}}{r_j - r_g} \right) \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} \left\{ \left( \frac{r_i - r_g}{r_i - r_j} \right) \left( \frac{r_i - r_{j+1}}{r_i - r_g} \right) \left( \frac{r_i - r_j}{r_i - r_{j+1}} \right) \right. \\
& \left. \left( \frac{r_i - r_j}{r_i - r_{j+1}} \right)^m f_i(r_{j+1}, \mathbf{k} - \mathbf{1}_g) \right\} \\
= & \left( \frac{r_{j+1} - r_g}{r_j - r_g} \right) \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} \left\{ \left( \frac{r_i - r_j}{r_i - r_{j+1}} \right)^{(m+1)} f_i(r_{j+1}, \mathbf{k}) \right\} + \\
& \left( 1 - \frac{r_{j+1} - r_g}{r_j - r_g} \right) \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} \left\{ \left( \frac{r_i - r_j}{r_i - r_{j+1}} \right)^{(m)} f_i(r_{j+1}, \mathbf{k} - \mathbf{1}_g) \right\} \\
= & \left( \frac{r_{j+1} - r_g}{r_j - r_g} \right) \Omega_i(m + 1, r_j, \mathbf{k}) + \left( 1 - \frac{r_{j+1} - r_g}{r_j - r_g} \right) \Omega_i(m, r_j, \mathbf{k} - \mathbf{1}_g).
\end{aligned}$$

□

**Definição 4:** Para  $\|\mathbf{k}\| = n + 1$ ,  $j = 1, \dots, L$ ,  $m = 0, \dots, n$ ,  $k_i > 0$  e  $r_i \geq r_j$ , seja

$$\Omega(m, r_j, \mathbf{k}) = \sum_{i:r_i \geq r_j} \Omega_i(m, r_j, \mathbf{k}) \quad (4.51)$$

$$= \sum_{i:r_i \geq r_j} \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} \left\{ \left( \frac{r_i - r_j}{r_i - r_{j+1}} \right)^m f_i(r_{j+1}, \mathbf{k}) \right\}. \quad (4.52)$$

É importante observarmos o que acontece com a definição acima quando  $m = 0$  e quando  $m = n$ . Lembramos que  $r_{j+1} < r_j$  e  $n > 0$ .

a) Para  $m = 0$ , de 4.52 temos

$$\begin{aligned}
\Omega(0, r_j, \mathbf{k}) &= \sum_{i:r_i \geq r_j} \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} \left\{ \left( \frac{r_i - r_j}{r_i - r_{j+1}} \right)^0 f_i(r_{j+1}, \mathbf{k}) \right\} \\
&= \sum_{i:r_i \geq r_j} \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} \{f_i(r_{j+1}, \mathbf{k})\} \\
&= \sum_{i:r_i > r_{j+1}} \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} \{f_i(r_{j+1}, \mathbf{k})\}. \quad (4.53)
\end{aligned}$$

Onde a última igualdade é válida pela simples observação que os valores de  $i$ , tal

que  $r_i \geq r_j$ , são idênticos aos valores de  $i$ , tal que  $r_i > r_{j+1}$ , uma vez que  $r_j$  é a menor recompensa maior que  $r_{j+1}$ .

De 4.7 e 4.6

$$\Omega(0, r_j, \mathbf{k}) = EO_{>}[\vec{r}, \mathbf{k}, r_{j+1}],$$

ou seja,  $\Omega(0, r_j, \mathbf{k})$  fornece a probabilidade de uma combinação linear de estatísticas de ordem com  $n + 1$  intervalos ser maior que  $r_{j+1}$ .

b) Para  $m = n$ , de 4.52 temos

$$\Omega(n, r_j, \mathbf{k}) = \sum_{i:r_i \geq r_j} \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} \left\{ \left( \frac{r_i - r_j}{r_i - r_{j+1}} \right)^n f_i(r_{j+1}, \mathbf{k}) \right\}. \quad (4.54)$$

De 4.44,

$$\Omega(n, r_j, \mathbf{k}) = \sum_{i:r_i \geq r_j} \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} \{f_i(r_j, \mathbf{k})\}. \quad (4.55)$$

De 4.7 e 4.6

$$\Omega(n, r_j, \mathbf{k}) = EO_{\geq}[\vec{r}, \mathbf{k}, r_j]. \quad (4.56)$$

De 4.54 e 4.51

$$\Omega(n, r_j, \mathbf{k}) = \Omega_j(n, r_j, \mathbf{k}) + \sum_{i:r_i > r_j} \Omega_i(m, r_j, \mathbf{k}). \quad (4.57)$$

De 4.42 e 4.24, para  $k_j < n + 1$ ,

$$\begin{aligned}
\Omega(n, r_j, \mathbf{k}) &= 0 + \sum_{i:r_i > r_j} \Omega_i(m, r_j, \mathbf{k}) \\
&= EO_{>}[\vec{r}, \mathbf{k}, r_j].
\end{aligned}$$

Para  $k_j = n + 1$ ,

$$\begin{aligned}
\Omega(n, r_j, \mathbf{k}) &= \Omega_j(n, r_j, \mathbf{k}) + \sum_{i:r_i > r_j} \Omega_i(m, r_j, \mathbf{k}) \\
&= \Omega_j(n, r_j, \mathbf{k}) + 0 \\
&= \Omega_j(n, r_j, \mathbf{k}) \\
&= 1.
\end{aligned}$$

Resumindo,

$$\Omega(0, r_j, \mathbf{k}) = EO_{>}[\vec{r}, \mathbf{k}, r_{j+1}]. \quad (4.58)$$

e

$$\Omega(n, r_j, \mathbf{k}) = \begin{cases} EO_{>}[\vec{r}, \mathbf{k}, r_j] = EO_{\geq}[\vec{r}, \mathbf{k}, r_j] & \text{para qualquer } \mathbf{k} \text{ tal que } k_j < n + 1 \\ EO_{\geq}[\vec{r}, \mathbf{k}, r_j] & \text{para qualquer } \mathbf{k} \end{cases} \quad (4.59)$$

**Lema 4:** Para  $\|k\| = n + 1$ ,  $n > 0$ ,  $m = 0, \dots, n$ ,  $g = 1, \dots, L + 1$  e  $k_g > 0$  temos

$$\Omega(m, r_j, \mathbf{k}) = \begin{cases} \left( \frac{r_g - r_j}{r_g - r_{j+1}} \right) \Omega(m - 1, r_j, \mathbf{k}) + \left( 1 - \frac{r_g - r_j}{r_g - r_{j+1}} \right) \Omega(m - 1, r_j, \mathbf{k} - \mathbf{1}_g) \\ \quad \text{para qualquer } r_g \geq r_j \\ \left( \frac{r_{j+1} - r_g}{r_j - r_g} \right) \Omega(m + 1, r_j, \mathbf{k}) + \left( 1 - \frac{r_{j+1} - r_g}{r_j - r_g} \right) \Omega(m, r_j, \mathbf{k} - \mathbf{1}_g) \\ \quad \text{para qualquer } r_g < r_j \end{cases} \quad (4.60)$$

Para  $\|\mathbf{k}\| = n + 1 = 1$ , temos

$$\Omega(0, r_j, \mathbf{1}_g) = \begin{cases} 0 & \text{se } r_g < r_j \\ 1 & \text{se } r_g \geq r_j \end{cases}$$

e para  $\|\mathbf{k}\| = n + 1 > 1$

$$\Omega(m, r_j, \mathbf{k}) = \begin{cases} 1 & \text{se } m = 0, j = L \text{ e } \sum_{g=1}^L k_g > 0 \\ 0 & \text{se } m = n, j = 1 \text{ e } \sum_{g=2}^{L+1} k_g > 0 \\ \Omega(0, r_{j-1}, \mathbf{k}) & \text{se } m = n, j = 2, \dots, L \text{ e } k_j < n + 1 \end{cases}$$

onde lembramos que  $L$  é o índice das recompensas e temos  $r_1 > r_2 > \dots > r_L > r_{L+1} = 0$ .

**Prova.** Primeiro iremos apresentar a prova das condições iniciais definidas para o Lema 4 e em seguida mostraremos a prova do lema.

Para  $n = 0$ , a prova das condições iniciais é trivial. De 4.58 temos  $\Omega(0, r_j, \mathbf{1}_g) = P[d_1 Y_1 > r_{j+1}]$ . Como só existe um intervalo,  $\Omega(0, r_j, \mathbf{1}_g)$  é igual a 1 se  $r_g > r_{j+1}$ , e é igual a zero se  $r_g \leq r_{j+1}$  (note que esse resultado pode ser obtido diretamente de 4.53).

Para  $n > 0$ , temos 3 condições iniciais.

a)  $\Omega(0, r_L, \mathbf{k}) = 1$  se  $\sum_{g=1}^L k_g > 0$

De 4.58 temos  $\Omega(0, r_L, \mathbf{k}) = P[\sum_{k=1}^{n+1} d_k Y_k > r_{L+1}]$ . Por definição,  $r_{L+1} = 0$ . Além disso, como  $\sum_{g=1}^L k_g > 0$ , então existe pelo menos um intervalo com recompensa maior que zero. Portanto,  $\Omega(0, r_L, \mathbf{k}) = 1$ .

b)  $\Omega(n, r_1, \mathbf{k}) = 0$  se  $\sum_{g=2}^{L+1} k_g > 0$

De 4.59 temos  $\Omega(n, r_1, \mathbf{k}) = P[\sum_{k=1}^{n+1} d_k Y_k \geq r_1]$ . A recompensa  $r_1$  é a maior recompensa do modelo. Como  $\sum_{g=2}^{L+1} k_g > 0$ , isto significa que existe pelo menos um intervalo com recompensa menor que  $r_1$ . Portanto,  $\Omega(n, r_1, \mathbf{k}) = 0$ .

c)  $\Omega(n, r_j, \mathbf{k}) = \Omega(0, r_{j-1}, \mathbf{k})$  para  $j = 2, \dots, L$  e  $k_j < n + 1$

De 4.58 temos  $\Omega(0, r_j, \mathbf{k}) = P[\sum_{k=1}^{n+1} d_k Y_k > r_{j+1}]$  e de 4.59 temos  $\Omega(n, r_j, \mathbf{k}) = P[\sum_{k=1}^{n+1} d_k Y_k > r_j]$  se  $k_j < n + 1$ . Portanto,  $\Omega(n, r_j, \mathbf{k}) = \Omega(0, r_{j-1}, \mathbf{k})$  para  $j = 2, \dots, L$  e  $k_j < n + 1$ .

A prova da equação 4.60 do Lema 4 é obtida usando os Lemas 2 e 3. Para  $r_g \geq r_j$  (veja equação 4.43) temos

$$\begin{aligned}
\Omega(m, r_j, \mathbf{k}) &= \sum_{i:r_i \geq r_j} \Omega_i(m, r_j, \mathbf{k}) \\
&= \sum_{i:r_i \geq r_j} \left\{ \left( \frac{r_g - r_j}{r_g - r_{j+1}} \right) \Omega_i(m - 1, r_j, \mathbf{k}) + \right. \\
&\quad \left. \left( 1 - \frac{r_g - r_j}{r_g - r_{j+1}} \right) \Omega_i(m - 1, r_j, \mathbf{k} - \mathbf{1}_g) \right\} \\
&= \sum_{i:r_i \geq r_j} \left\{ \left( \frac{r_g - r_j}{r_g - r_{j+1}} \right) \Omega_i(m - 1, r_j, \mathbf{k}) \right\} + \\
&\quad \sum_{i:r_i \geq r_j} \left\{ \left( 1 - \frac{r_g - r_j}{r_g - r_{j+1}} \right) \Omega_i(m - 1, r_j, \mathbf{k} - \mathbf{1}_g) \right\} \\
&= \left( \frac{r_g - r_j}{r_g - r_{j+1}} \right) \sum_{i:r_i \geq r_j} \Omega_i(m - 1, r_j, \mathbf{k}) + \\
&\quad \left( 1 - \frac{r_g - r_j}{r_g - r_{j+1}} \right) \sum_{i:r_i \geq r_j} \Omega_i(m - 1, r_j, \mathbf{k} - \mathbf{1}_g) \\
&= \left( \frac{r_g - r_j}{r_g - r_{j+1}} \right) \Omega(m - 1, r_j, \mathbf{k}) + \\
&\quad \left( 1 - \frac{r_g - r_j}{r_g - r_{j+1}} \right) \Omega(m - 1, r_j, \mathbf{k} - \mathbf{1}_g).
\end{aligned}$$

Para  $r_g < r_j$  (veja equação 4.50) temos

$$\begin{aligned}
\Omega(m, r_j, \mathbf{k}) &= \sum_{i:r_i \geq r_j} \Omega_i(m, r_j, \mathbf{k}) \\
&= \sum_{i:r_i \geq r_j} \left\{ \left( \frac{r_{j+1} - r_g}{r_j - r_g} \right) \Omega_i(m + 1, r_j, \mathbf{k}) + \right. \\
&\quad \left. \left( 1 - \frac{r_{j+1} - r_g}{r_j - r_g} \right) \Omega_i(m, r_j, \mathbf{k} - \mathbf{1}_g) \right\} \\
&= \sum_{i:r_i \geq r_j} \left\{ \left( \frac{r_{j+1} - r_g}{r_j - r_g} \right) \Omega_i(m + 1, r_j, \mathbf{k}) \right\} + \\
&\quad \sum_{i:r_i \geq r_j} \left\{ \left( 1 - \frac{r_{j+1} - r_g}{r_j - r_g} \right) \Omega_i(m, r_j, \mathbf{k} - \mathbf{1}_g) \right\}.
\end{aligned}$$

$$\begin{aligned}
&= \left( \frac{r_{j+1} - r_g}{r_j - r_g} \right) \sum_{i:r_i \geq r_j} \Omega_i(m+1, r_j, \mathbf{k}) + \\
&\quad \left( 1 - \frac{r_{j+1} - r_g}{r_j - r_g} \right) \sum_{i:r_i \geq r_j} \Omega_i(m, r_j, \mathbf{k} - \mathbf{1}_g) \\
&= \left( \frac{r_{j+1} - r_g}{r_j - r_g} \right) \Omega(m+1, r_j, \mathbf{k}) + \left( 1 - \frac{r_{j+1} - r_g}{r_j - r_g} \right) \Omega(m, r_j, \mathbf{k} - \mathbf{1}_g).
\end{aligned}$$

□

**Teorema 2:** Para  $\|k\| = n + 1$ ,  $r_{j+1} \leq r < r_j$  e  $j = 1, \dots, L$ ,

$$EO_{>}[\vec{r}, \mathbf{k}, r] = \sum_{l=0}^n \binom{n}{l} \left( \frac{r - r_{j+1}}{r_j - r_{j+1}} \right)^l \left( 1 - \frac{r - r_{j+1}}{r_j - r_{j+1}} \right)^{(n-l)} \Omega(l, r_j, \mathbf{k}). \quad (4.61)$$

**Prova.**

$$\begin{aligned}
EO_{>}[\vec{r}, \mathbf{k}, r] &= \sum_{i:r_i > r} \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} f_i(r, \mathbf{k}) \\
&= \sum_{i:r_i > r} \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} \left\{ \left( \frac{r_i - r}{r_i - r_{j+1}} \right)^n f_i(r_{j+1}, \mathbf{k}) \right\} \\
&= \sum_{i:r_i > r} \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} \left\{ \left( 1 - \frac{r - r_{j+1}}{r_i - r_{j+1}} \right)^n f_i(r_{j+1}, \mathbf{k}) \right\} \\
&= \sum_{i:r_i > r} \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} \left\{ \left( 1 - \frac{r - r_{j+1}}{r_j - r_{j+1}} \times \frac{r_j - r_{j+1}}{r_i - r_{j+1}} \right)^n f_i(r_{j+1}, \mathbf{k}) \right\} \\
&= \sum_{i:r_i > r} \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} \left\{ \left( 1 - \frac{r - r_{j+1}}{r_j - r_{j+1}} \times \left( 1 - \frac{r_i - r_j}{r_i - r_{j+1}} \right) \right)^n \times \right. \\
&\quad \left. f_i(r_{j+1}, \mathbf{k}) \right\} \\
&= \sum_{i:r_i > r} \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} \left\{ \left( \left( 1 - \frac{r - r_{j+1}}{r_j - r_{j+1}} \right) + \right. \right. \\
&\quad \left. \left. \left( \frac{r - r_{j+1}}{r_j - r_{j+1}} \times \frac{r_i - r_j}{r_i - r_{j+1}} \right) \right)^n f_i(r_{j+1}, \mathbf{k}) \right\} \\
&= \sum_{i:r_i > r} \frac{1}{(k_i - 1)!} \frac{d^{(k_i-1)}}{dr_i} \left\{ \sum_{l=0}^n \binom{n}{l} \left( \frac{r - r_{j+1}}{r_j - r_{j+1}} \times \frac{r_i - r_j}{r_i - r_{j+1}} \right)^l \times \right. \\
&\quad \left. \left( 1 - \frac{r - r_{j+1}}{r_j - r_{j+1}} \right)^{(n-l)} f_i(r_{j+1}, \mathbf{k}) \right\}.
\end{aligned}$$

Mudando a posição do segundo somatório para o início da equação, temos

$$EO_{>}[\vec{r}, \mathbf{k}, r] = \sum_{i:r_i>r} \sum_{l=0}^n \frac{1}{(k_i-1)!} \frac{d^{(k_i-1)}}{dr_i} \left\{ \binom{n}{l} \left( \frac{r-r_{j+1}}{r_j-r_{j+1}} \right)^l \times \left( 1 - \frac{r-r_{j+1}}{r_j-r_{j+1}} \right)^{(n-l)} \left( \frac{r_i-r_j}{r_i-r_{j+1}} \right)^l f_i(r_{j+1}, \mathbf{k}) \right\}. \quad (4.62)$$

Podemos tirar do cálculo da derivada da equação 4.62 os termos independentes de  $r_i$ .

$$\begin{aligned} EO_{>}[\vec{r}, \mathbf{k}, r] &= \sum_{i:r_i>r} \sum_{l=0}^n \binom{n}{l} \left( \frac{r-r_{j+1}}{r_j-r_{j+1}} \right)^l \left( 1 - \frac{r-r_{j+1}}{r_j-r_{j+1}} \right)^{(n-l)} \times \\ &\quad \frac{1}{(k_i-1)!} \frac{d^{(k_i-1)}}{dr_i} \left\{ \left( \frac{r_i-r_j}{r_i-r_{j+1}} \right)^l f_i(r_{j+1}, \mathbf{k}) \right\} \\ &= \sum_{l=0}^n \binom{n}{l} \left( \frac{r-r_{j+1}}{r_j-r_{j+1}} \right)^l \left( 1 - \frac{r-r_{j+1}}{r_j-r_{j+1}} \right)^{(n-l)} \times \\ &\quad \sum_{i:r_i>r} \frac{1}{(k_i-1)!} \frac{d^{(k_i-1)}}{dr_i} \left\{ \left( \frac{r_i-r_j}{r_i-r_{j+1}} \right)^l f_i(r_{j+1}, \mathbf{k}) \right\} \\ &= \sum_{l=0}^n \binom{n}{l} \left( \frac{r-r_{j+1}}{r_j-r_{j+1}} \right)^l \left( 1 - \frac{r-r_{j+1}}{r_j-r_{j+1}} \right)^{(n-l)} \sum_{i:r_i>r} \Omega_i(l, r_j, \mathbf{k}). \end{aligned}$$

Sabemos que  $r_j + 1 \leq r < r_j$ . Então,

$$\begin{aligned} EO_{>}[\vec{r}, \mathbf{k}, r] &= \sum_{l=0}^n \binom{n}{l} \left( \frac{r-r_{j+1}}{r_j-r_{j+1}} \right)^l \left( 1 - \frac{r-r_{j+1}}{r_j-r_{j+1}} \right)^{(n-l)} \sum_{i:r_i \geq r_j} \Omega_i(l, r_j, \mathbf{k}) \\ &= \sum_{l=0}^n \binom{n}{l} \left( \frac{r-r_{j+1}}{r_j-r_{j+1}} \right)^l \left( 1 - \frac{r-r_{j+1}}{r_j-r_{j+1}} \right)^{(n-l)} \Omega(l, r_j, \mathbf{k}). \end{aligned}$$

□

O custo computacional do algoritmo é  $O(LN^2)$ , onde  $L + 1$  é o número de recompensas do modelo e  $\|\mathbf{k}\| = N + 1$ . Note que no passo  $N$  do algoritmo só precisamos conhecer os termos calculados no passo  $N - 1$ .



## 4.4 Recursão para $P[ACR(t) > r]$

O objetivo dessa seção é obter um algoritmo para  $P[ACR(t) > r]$  usando a mesma metodologia de [49]. Como vimos, esta metodologia encontra uma recursão para  $P[ACR(t) > r]$  a partir de uma recursão para  $\Gamma[n, \mathbf{k}]$  e de uma recursão para  $P[ACR(t) > r | n, \mathbf{k}]$ . Na seção 4.3 apresentamos duas recursões para combinação linear de estatísticas de ordem. A primeira recursão (Teorema 1) tem um custo da  $O(N)$  enquanto a segunda recursão (Teorema 2) tem um custo da  $O(LN^2)$ , onde  $\|\mathbf{k}\| = N + 1$  e  $L + 1$  é o número de recompensas do modelo.

Infelizmente não foi encontrada uma recursão para  $P[ACR(t) > r]$  usando o primeiro algoritmo. A explicação para isto é simples. Cada passo do algoritmo 1 para  $P[ACR(t) > r | n, \mathbf{k}]$  exige a eliminação de dois intervalos do vetor  $\mathbf{k}$ , enquanto o algoritmo para  $\Gamma[n, \mathbf{k}]$  exige a eliminação de apenas um intervalo. Este fato impossibilita o agrupamento dessas duas recursões como definida na metodologia de [49].

Portanto, usaremos o segundo algoritmo de combinação linear de estatísticas de ordem para encontrar uma recursão para  $P[ACR(t) > r]$ . Na realidade, o algoritmo encontrado é idêntico ao algoritmo proposto em [54]. A grande vantagem do uso da metodologia de [49] é a obtenção de uma interpretação probabilística para o algoritmo, mostrando que o algoritmo de [54] é encontrado diretamente de argumentos probabilísticos idênticos ao usado para se obter o algoritmo de [52].

Após mostrarmos o algoritmo de [54] obtido usando a metodologia de [49], apresentaremos uma modificação no algoritmo de [54] e mostraremos o algoritmo para a distribuição de recompensa acumulada com limites de [61]. A definição do algoritmo de [61] se tornou possível a partir da interpretação probabilística e da modificação do algoritmo de [54] aqui discutidas.

#### 4.4.1 Interpretação Probabilística

Inicialmente vamos definir a recursão  $\Phi[n, \mathbf{k}]$  similar à recursão  $\Gamma[n, \mathbf{k}]$  (equação 4.13), isto é,  $\Phi[n, \mathbf{k}]$  é a probabilidade de uma coloração  $\mathbf{k}$  dadas  $n$  transições. A recursão de  $\Phi[n, \mathbf{k}]$  é feita de acordo com o primeiro estado visitado, enquanto a recursão de  $\Gamma[n, \mathbf{k}]$  é feita pelo último estado visitado. Usaremos  $\Phi[n, \mathbf{k}]$  nesta seção pois é nosso objetivo obter o algoritmo de [54] usando a metodologia de [49]. Entretanto, toda a discussão feita a seguir também é válida para a recursão de  $\Gamma[n, \mathbf{k}]$ .

$$\Phi[n, \mathbf{k}] = \sum_{s \in \mathbf{S}} \pi_s(0) \Phi_s[n, \mathbf{k}], \quad (4.63)$$

onde

$$\Phi_s[n, \mathbf{k}] = \sum_{s' \in \mathbf{S}} \Phi_{s'}[n-1, \mathbf{k} - \mathbf{1}_{c(s)}] p_{s, s'} \quad (4.64)$$

e

$$\Phi_s[0, \mathbf{1}_i] = \begin{cases} 1 & \text{se } i = c(s) \\ 0 & \text{nos outros casos} \end{cases} \quad (4.65)$$

A prova de 4.64 é similar à prova da recursão de 4.13 e que foi apresentada em [49].

Vamos agora apresentar duas novas definições que serão utilizadas no decorrer desta seção, e em seguida mostraremos uma recursão a partir do Teorema 2 (equação 4.61) e da recursão de  $\Phi[n, \mathbf{k}]$  (equação 4.64).

**Definição 5:** para  $n \geq 0$ ,  $s \in \mathbf{S}$ ,  $r_{c(s)} \in \{r_1, \dots, r_{L+1}\}$ , seja

$$G_g[i, n] = \{\mathbf{k} \in \mathcal{K}_n : k_i = g\}, \quad (4.66)$$

onde  $\mathcal{K}_n = \{\mathbf{k} : \|\mathbf{k}\| = n+1\}$ . Para  $n > 0$ ,  $s \in \mathbf{S}$  e  $r_{c(s)} \in \mathcal{R}$ , temos [51]

$$\{\mathbf{k} - \mathbf{1}_{c(s)} : \mathbf{k} \in \mathcal{K}_n \text{ e } k_{c(s)} > 0\} = \begin{cases} G_g[i, n-1] & \text{se } i \neq c(s) \text{ e } g \geq 0 \\ G_{g-1}[i, n-1] & \text{se } i = c(s) \text{ e } g > 0 \end{cases} \quad (4.67)$$

**Definição 6:** para  $n \geq 0$ ,  $m = 0, \dots, n$ ,  $s \in \mathbf{S}$  e  $j = 1, \dots, L$ , seja

$$\begin{aligned} \Upsilon_s(n, m, j) &= \sum_{g=0}^{n+1} \sum_{\mathbf{k} \in G_g[c(s), n]} \Phi_s(n, \mathbf{k}) \Omega(m, r_j, \mathbf{k}) \\ &= \sum_{g=1}^{n+1} \sum_{\mathbf{k} \in G_g[c(s), n]} \Phi_s(n, \mathbf{k}) \Omega(m, r_j, \mathbf{k}). \end{aligned} \quad (4.68)$$

A segunda igualdade é verdadeira pois, para  $g = 0$ ,  $\Phi_s(n, \mathbf{k}) = 0$ . Note que como o primeiro estado visitado é  $s$ , necessariamente  $k_{c(s)} > 0$ , logo a probabilidade de uma coloração  $\mathbf{k}$  onde  $k_{c(s)} = 0$  é zero.

**Lema 5:** Para  $n > 0$ ,  $m = 0, \dots, n$  e  $j = 1, \dots, L$  e  $s \in \mathbf{S}$ ,

$$\Upsilon_s(n, m, j) = \begin{cases} \left( \frac{r_{c(s)} - r_j}{r_{c(s)} - r_{j+1}} \right) \Upsilon_s(n, m-1, j) + \left( 1 - \frac{r_{c(s)} - r_j}{r_{c(s)} - r_{j+1}} \right) \sum_{s' \in \mathbf{S}} \Upsilon_{s'}(n-1, m-1, j) p_{s, s'} \\ \quad \text{para qualquer } r_{c(s)} \geq r_j \\ \\ \left( \frac{r_{j+1} - r_{c(s)}}{r_j - r_{c(s)}} \right) \Upsilon_s(n, m+1, j) + \left( 1 - \frac{r_{j+1} - r_{c(s)}}{r_j - r_{c(s)}} \right) \sum_{s' \in \mathbf{S}} \Upsilon_{s'}(n-1, m, j) p_{s, s'} \\ \quad \text{para qualquer } r_{c(s)} < r_j \end{cases} \quad (4.69)$$

Para  $n = 0$  temos

$$\Upsilon_s(0, 0, j) = \begin{cases} 1 & \text{se } j \text{ é tal que } r_{c(s)} \geq r_j \\ 0 & \text{se } j \text{ é tal que } r_{c(s)} < r_j \end{cases}$$

e para  $n > 0$  temos

$$\Upsilon_s(n, m, j) = \begin{cases} 1 & \text{se } m = 0, r_{c(s)} \geq j \text{ e } j = L \\ 0 & \text{se } m = n, r_{c(s)} < j \text{ e } j = 1 \\ \Upsilon_s(n, 0, j-1) & \text{se } m = n, r_{c(s)} \neq r_j \text{ e } j = 2, \dots, L \end{cases} \quad (4.70)$$

**Prova.** Antes de apresentarmos a prova do Lema 5, é importante verificar como o cálculo dos  $\Upsilon(n, m, j)$  é implementado. A Figura 4.4 mostra a implementação

para um modelo onde  $\mathcal{R} = \{r_1, r_2, r_3, r_4\}$ . Lembramos que  $r_1 > r_2 > \dots > r_L > r_{L+1} = 0$ . As setas mostram a direção do cálculo dos  $\Upsilon_s(n, m, j)$ . Por exemplo, para  $r_{c(s)} = r_2$ , temos o valor inicial de  $\Upsilon_s(n, 0, 3)$  e podemos calcular  $\Upsilon_s(n, m, 3)$  para  $m = 1, \dots, n$ . De acordo com as condições iniciais, temos  $\Upsilon_s(n, n, 3) = \Upsilon_s(n, 0, 2)$ , então podemos calcular  $\Upsilon_s(n, m, 2)$  para  $m = 1, \dots, n$ . Temos também o valor inicial de  $\Upsilon_s(n, n, 1)$  e assim podemos calcular  $\Upsilon_s(n, m, 1)$  para  $m = n - 1, \dots, 0$ . Note que podemos ter  $\Upsilon_s(n, n, 2) \neq \Upsilon_s(n, 0, 1)$ , o motivo desta diferença será explicado na prova das condições iniciais.

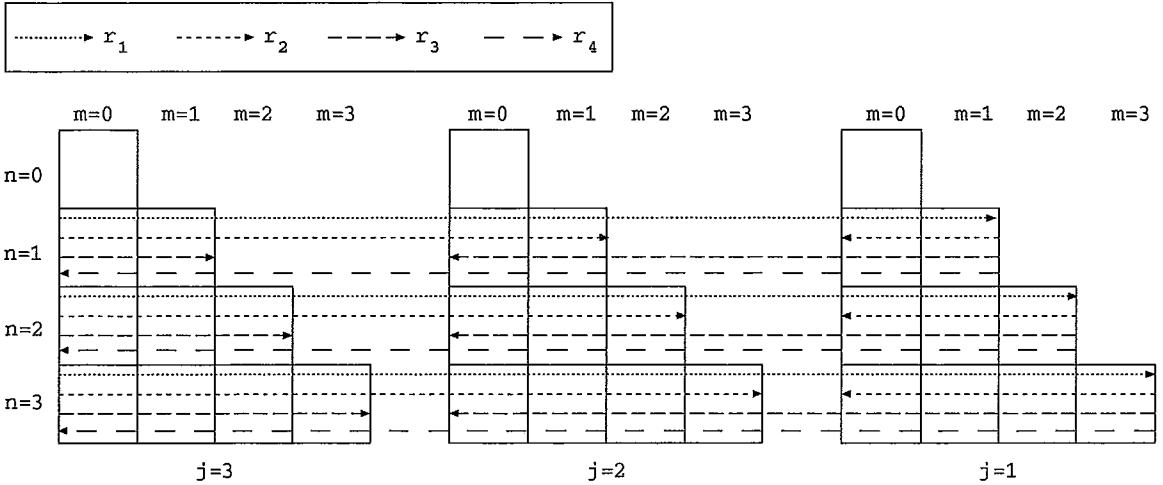


Figura 4.4: Cálculo dos  $\Upsilon_s(n, m, j)$ .

Iremos agora mostrar a prova das condições iniciais definidas para o Lema 5 e em seguida mostraremos a prova de 4.69.

Para  $n = 0$ , a prova das condições iniciais é trivial. De 4.65 temos  $\Phi_s[0, \mathbf{1}_{c(s)}] = 1$ , e de 4.58 temos  $\Omega(0, r_j, \mathbf{1}_{c(s)}) = 1$  se  $r_{c(s)} \geq r_j$ . Portanto,  $\Upsilon_s(0, 0, j)$  é igual a 1 se  $r_{c(s)} \geq r_j$  e é igual a zero nos outros casos.

Para  $n > 0$ , temos 3 condições iniciais. Vejamos a prova.

a)  $\Upsilon_s(n, 0, L) = 1$  se  $r_{c(s)} \geq r_L$

De 4.68 e 4.58 temos

$$\begin{aligned}
 \Upsilon_s(n, 0, L) &= \sum_{g=1}^{n+1} \sum_{\mathbf{k} \in G_g[c(s), n]} \Phi_s(n, \mathbf{k}) \Omega(0, r_L, \mathbf{k}) \\
 &= \sum_{g=1}^{n+1} \sum_{\mathbf{k} \in G_g[c(s), n]} \Phi_s(n, \mathbf{k}) EO_{>}[\vec{r}, \mathbf{k}, r_{L+1}] \\
 &= \sum_{g=1}^{n+1} \sum_{\mathbf{k} \in G_g[c(s), n]} \Phi_s(n, \mathbf{k}) EO_{>}[\vec{r}, \mathbf{k}, 0].
 \end{aligned}$$

Note que o primeiro estado visitado  $s$  tem recompensa  $r_{c(s)} \geq r_L > r_{L+1}$ , logo

$$\begin{aligned}
 \Upsilon_s(n, 0, j) &= \sum_{g=1}^{n+1} \sum_{\mathbf{k} \in G_g[c(s), n]} \Phi_s(n, \mathbf{k}) \times 1 \\
 &= \sum_{g=1}^{n+1} \sum_{\mathbf{k} \in G_g[c(s), n]} \Phi_s(n, \mathbf{k}) \\
 &= 1.
 \end{aligned}$$

A última igualdade corresponde à soma de todas as probabilidades dado que  $s$  é o primeiro estado visitado.

b)  $\Upsilon_s(n, n, 1) = 0$  se  $r_{c(s)} < r_1$

De 4.68 e 4.59 temos

$$\begin{aligned}
 \Upsilon_s(n, n, 1) &= \sum_{g=1}^{n+1} \sum_{\mathbf{k} \in G_g[c(s), n]} \Phi_s(n, \mathbf{k}) \Omega(n, r_1, \mathbf{k}) \\
 &= \sum_{g=1}^{n+1} \sum_{\mathbf{k} \in G_g[c(s), n]} \Phi_s(n, \mathbf{k}) EO_{\geq}[\vec{r}, \mathbf{k}, r_1].
 \end{aligned}$$

Sabemos que  $r_1$  é a maior recompensa do modelo. Além disso, existe um intervalo com recompensa  $r_{c(s)} < r_1$ . Portanto,  $EO_{\geq}[\vec{r}, \mathbf{k}, r_1] = 0$ .

$$\begin{aligned}
 \Upsilon_s(n, n, 1) &= \sum_{g=1}^{n+1} \sum_{\mathbf{k} \in G_g[c(s), n]} \Phi_s(n, \mathbf{k}) \times 0 \\
 &= 0.
 \end{aligned}$$

c)  $\Upsilon_s(n, n, j) = \Upsilon_s(n, 0, j - 1)$  se  $r_{c(s)} \neq r_j$  e  $j = 2, \dots, L$

Da definição 4.68

$$\Upsilon_s(n, n, j) = \sum_{g=1}^{n+1} \sum_{\mathbf{k} \in G_g[c(s), n]} \Phi_s(n, \mathbf{k}) \Omega(n, r_j, \mathbf{k}).$$

Das condições iniciais do Lema 4 (equação 4.60) e observando que, se  $c(s) \neq r_j$  e  $s$  é o último estado visitado,  $k_{c(s)} > 0$ , e portanto  $k_j < n + 1$ .

$$\begin{aligned} \Upsilon_s(n, n, j) &= \sum_{g=1}^{n+1} \sum_{\mathbf{k} \in G_g[c(s), n]} \Phi_s(n, \mathbf{k}) \Omega(0, r_{j-1}, \mathbf{k}) \\ &= \Upsilon_s(n, 0, j - 1). \end{aligned}$$

Vamos agora mostrar a prova do Lema 5 (equação 4.69) para  $r_{c(s)} \geq r_j$  e depois para  $r_{c(s)} < r_j$ .

a) Para  $r_{c(s)} \geq r_j$

$$\begin{aligned} \Upsilon_s(n, m, j) &= \sum_{g=1}^{n+1} \sum_{\mathbf{k} \in G_g[c(s), n]} \Phi_s(n, \mathbf{k}) \Omega(m, r_j, \mathbf{k}) \\ &= \sum_{g=1}^{n+1} \sum_{\mathbf{k} \in G_g[c(s), n]} \Phi_s(n, \mathbf{k}) \left\{ \left( \frac{r_{c(s)} - r_j}{r_{c(s)} - r_{j+1}} \right) \Omega(m - 1, r_j, \mathbf{k}) + \right. \\ &\quad \left. \left( 1 - \frac{r_{c(s)} - r_j}{r_{c(s)} - r_{j+1}} \right) \Omega(m - 1, r_j, \mathbf{k} - \mathbf{1}_{c(s)}) \right\} \\ &= \sum_{g=1}^{n+1} \sum_{\mathbf{k} \in G_g[c(s), n]} \left( \frac{r_{c(s)} - r_j}{r_{c(s)} - r_{j+1}} \right) \Omega(m - 1, r_j, \mathbf{k}) \Phi_s(n, \mathbf{k}) + \\ &\quad \sum_{g=1}^{n+1} \sum_{\mathbf{k} \in G_g[c(s), n]} \left( 1 - \frac{r_{c(s)} - r_j}{r_{c(s)} - r_{j+1}} \right) \Omega(m - 1, r_j, \mathbf{k} - \mathbf{1}_{c(s)}) \Phi_s(n, \mathbf{k}). \end{aligned} \tag{4.71}$$

Usando a equação 4.64 no segundo termo da equação 4.71,

$$\begin{aligned}
\Upsilon_s(n, m, j) &= \left( \frac{r_{c(s)} - r_j}{r_{c(s)} - r_{j+1}} \right) \sum_{g=1}^{n+1} \sum_{\mathbf{k} \in G_g[c(s), n]} \Omega(m-1, r_j, \mathbf{k}) \Phi_s(n, \mathbf{k}) + \\
&\quad \left( 1 - \frac{r_{c(s)} - r_j}{r_{c(s)} - r_{j+1}} \right) \sum_{g=1}^{n+1} \sum_{\mathbf{k} \in G_g[c(s), n]} \Omega(m-1, r_j, \mathbf{k} - \mathbf{1}_{c(s)}) \times \\
&\quad \sum_{s' \in \mathbf{S}} \Phi_{s'}(n-1, \mathbf{k} - \mathbf{1}_{c(s)}) p_{s, s'} \\
&= \left( \frac{r_{c(s)} - r_j}{r_{c(s)} - r_{j+1}} \right) \sum_{g=1}^{n+1} \sum_{\mathbf{k} \in G_g[c(s), n]} \Omega(m-1, r_j, \mathbf{k}) \Phi_s(n, \mathbf{k}) + \\
&\quad \left( 1 - \frac{r_{c(s)} - r_j}{r_{c(s)} - r_{j+1}} \right) \sum_{s' \in \mathbf{S}} \sum_{g=1}^{n+1} \sum_{\mathbf{k} \in G_g[c(s), n]} \Omega(m-1, r_j, \mathbf{k} - \mathbf{1}_{c(s)}) \times \\
&\quad \Phi_{s'}(n-1, \mathbf{k} - \mathbf{1}_{c(s)}) p_{s, s'}. \tag{4.72}
\end{aligned}$$

De 4.67 temos  $\{\mathbf{k} - \mathbf{1}_{c(s)} : \mathbf{k} \in \mathcal{K}_n \text{ e } k_{c(s)} > 0\} = G_{g-1}[c(s), n-1]$  para  $g = 1, \dots, n+1$ . Podemos então escrever 4.72 como

$$\begin{aligned}
\Upsilon_s(n, m, j) &= \left( \frac{r_{c(s)} - r_j}{r_{c(s)} - r_{j+1}} \right) \sum_{g=1}^{n+1} \sum_{\mathbf{k} \in G_g[c(s), n]} \Omega(m-1, r_j, \mathbf{k}) \Phi_s(n, \mathbf{k}) + \\
&\quad \left( 1 - \frac{r_{c(s)} - r_j}{r_{c(s)} - r_{j+1}} \right) \sum_{s' \in \mathbf{S}} \sum_{g=1}^{n+1} \sum_{\mathbf{k} \in G_{g-1}[c(s), n-1]} \Omega(m-1, r_j, \mathbf{k}) \times \\
&\quad \Phi_{s'}(n-1, \mathbf{k}) p_{s, s'}.
\end{aligned}$$

Note que

$$\begin{aligned}
&\sum_{g=1}^{n+1} \sum_{\mathbf{k} \in G_{g-1}[c(s), n-1]} \Omega(m-1, r_j, \mathbf{k}) \Phi_{s'}(n-1, \mathbf{k}) p_{s, s'} = \\
&\quad \sum_{\mathbf{k} \in \bigcup_{g=1}^{n+1} G_{g-1}[c(s), n-1]} \Omega(m-1, r_j, \mathbf{k}) \Phi_{s'}(n-1, \mathbf{k}) p_{s, s'}.
\end{aligned}$$

Além disso,

$$\{\mathbf{k} : \mathbf{k} \in \bigcup_{g=1}^{n+1} G_{g-1}[c(s), n-1]\} = \{\mathbf{k} : \mathbf{k} \in \bigcup_{g=0}^n G_g[c(s), n-1]\} = \{\mathbf{k} : \mathbf{k} \in \mathcal{K}_{n-1}\}.$$

Logo,

$$\begin{aligned} \Upsilon_s(n, m, j) &= \left( \frac{r_{c(s)} - r_j}{r_{c(s)} - r_{j+1}} \right) \sum_{g=1}^{n+1} \sum_{\mathbf{k} \in G_g[c(s), n]} \Omega(m-1, r_j, \mathbf{k}) \Phi_s(n, \mathbf{k}) + \\ &\left( 1 - \frac{r_{c(s)} - r_j}{r_{c(s)} - r_{j+1}} \right) \sum_{s' \in \mathbf{S}} \sum_{\mathbf{k} \in \mathcal{K}_{n-1}} \Omega(m-1, r_j, \mathbf{k}) \times \\ &\Phi_{s'}(n-1, \mathbf{k}) p_{s, s'}. \end{aligned}$$

Da definição 5 (equação 4.66) temos  $\mathcal{K}_{n-1} = \bigcup_{g=0}^n G_g[c(s), n-1]$  para qualquer  $s \in \mathcal{S}$ . Portanto,

$$\begin{aligned} \Upsilon_s(n, m, j) &= \left( \frac{r_{c(s)} - r_j}{r_{c(s)} - r_{j+1}} \right) \sum_{g=1}^{n+1} \sum_{\mathbf{k} \in G_g[c(s), n]} \Omega(m-1, r_j, \mathbf{k}) \Phi_s(n, \mathbf{k}) + \\ &\left( 1 - \frac{r_{c(s)} - r_j}{r_{c(s)} - r_{j+1}} \right) \sum_{s' \in \mathbf{S}} \sum_{g=0}^n \sum_{\mathbf{k} \in G_g[c(s'), n-1]} \Omega(m-1, r_j, \mathbf{k}) \times \\ &\Phi_{s'}(n-1, \mathbf{k}) p_{s, s'}. \end{aligned}$$

Como  $\Phi_{s'}(n-1, \mathbf{k})$  é zero para  $k_{c(s')} = 0$  (probabilidade de uma coloração  $\mathbf{k}$  dado que o primeiro estado visitado é  $s$  e o número de visitas ao estado  $s$  é zero), temos

$$\begin{aligned} \Upsilon_s(n, m, j) &= \left( \frac{r_{c(s)} - r_j}{r_{c(s)} - r_{j+1}} \right) \sum_{g=1}^{n+1} \sum_{\mathbf{k} \in G_g[c(s), n]} \Omega(m-1, r_j, \mathbf{k}) \Phi_s(n, \mathbf{k}) + \\ &\left( 1 - \frac{r_{c(s)} - r_j}{r_{c(s)} - r_{j+1}} \right) \sum_{s' \in \mathbf{S}} \sum_{g=1}^n \sum_{\mathbf{k} \in G_g[c(s'), n-1]} \Omega(m-1, r_j, \mathbf{k}) \times \\ &\Phi_{s'}(n-1, \mathbf{k}) p_{s, s'} \\ &= \left( \frac{r_{c(s)} - r_j}{r_{c(s)} - r_{j+1}} \right) \Upsilon_s(n, m-1, j) + \\ &\left( 1 - \frac{r_{c(s)} - r_j}{r_{c(s)} - r_{j+1}} \right) \sum_{s' \in \mathbf{S}} \Upsilon_{s'}(n-1, m-1, j) p_{s, s'}, \end{aligned}$$

onde a última igualdade é obtida da definição de  $\Upsilon_s(n, m, j)$  em 4.68.

b) Para  $r_{c(s)} < r_j$



$$\begin{aligned}
\Upsilon_s(n, m, j) &= \sum_{g=1}^{n+1} \sum_{\mathbf{k} \in G_g[c(s), n]} \Phi_s(n, \mathbf{k}) \Omega(m, r_j, \mathbf{k}) \\
&= \sum_{g=1}^{n+1} \sum_{\mathbf{k} \in G_g[c(s), n]} \Phi_s(n, \mathbf{k}) \left\{ \left( \frac{r_{j+1} - r_{c(s)}}{r_j + r_{c(s)}} \right) \Omega(m+1, r_j, \mathbf{k}) + \right. \\
&\quad \left. \left( 1 - \frac{r_{j+1} - r_{c(s)}}{r_j - r_{c(s)}} \right) \Omega(m, r_j, \mathbf{k} - \mathbf{1}_{c(s)}) \right\} \\
&= \sum_{g=1}^{n+1} \sum_{\mathbf{k} \in G_g[c(s), n]} \left( \frac{r_{j+1} - r_{c(s)}}{r_j - r_{c(s)}} \right) \Omega(m+1, r_j, \mathbf{k}) \Phi_s(n, \mathbf{k}) + \\
&\quad \sum_{g=1}^{n+1} \sum_{\mathbf{k} \in G_g[c(s), n]} \left( 1 - \frac{r_{j+1} - r_{c(s)}}{r_j - r_{c(s)}} \right) \Omega(m, r_j, \mathbf{k} - \mathbf{1}_{c(s)}) \Phi_s(n, \mathbf{k})
\end{aligned}$$

Utilizando a recursão para  $\Phi_s[n, \mathbf{k}]$  de 4.64 temos

$$\begin{aligned}
\Upsilon_s(n, m, j) &= \left( \frac{r_{j+1} - r_{c(s)}}{r_j - r_{c(s)}} \right) \sum_{g=1}^{n+1} \sum_{\mathbf{k} \in G_g[c(s), n]} \Omega(m+1, r_j, \mathbf{k}) \Phi_s(n, \mathbf{k}) + \\
&\quad \left( 1 - \frac{r_{j+1} - r_{c(s)}}{r_j - r_{c(s)}} \right) \sum_{g=1}^{n+1} \sum_{\mathbf{k} \in G_g[c(s), n]} \Omega(m, r_j, \mathbf{k} - \mathbf{1}_{c(s)}) \times \\
&\quad \sum_{s' \in \mathbf{S}} \Phi_{s'}(n-1, \mathbf{k} - \mathbf{1}_{c(s)}) p_{s, s'} \\
&= \left( \frac{r_{j+1} - r_{c(s)}}{r_j - r_{c(s)}} \right) \sum_{g=1}^{n+1} \sum_{\mathbf{k} \in G_g[c(s), n]} \Omega(m+1, r_j, \mathbf{k}) \Phi_s(n, \mathbf{k}) + \\
&\quad \left( 1 - \frac{r_{j+1} - r_{c(s)}}{r_j - r_{c(s)}} \right) \sum_{s' \in \mathbf{S}} \sum_{g=1}^{n+1} \sum_{\mathbf{k} \in G_g[c(s), n]} \Omega(m, r_j, \mathbf{k} - \mathbf{1}_{c(s)}) \times \\
&\quad \Phi_{s'}(n-1, \mathbf{k} - \mathbf{1}_{c(s)}) p_{s, s'}.
\end{aligned}$$

De 4.67 temos  $\{\mathbf{k} - \mathbf{1}_{c(s)} : \mathbf{k} \in \mathcal{K}_n \text{ e } k_{c(s)} > 0\} = G_{g-1}[c(s), n-1]$  para  $g = 1, \dots, n+1$ . Então,

$$\Upsilon_s(n, m, j) = \left( \frac{r_{j+1} - r_{c(s)}}{r_j - r_{c(s)}} \right) \sum_{g=1}^{n+1} \sum_{\mathbf{k} \in G_g[c(s), n]} \Omega(m+1, r_j, \mathbf{k}) \Phi_s(n, \mathbf{k}) +$$

$$\begin{aligned}
& \left(1 - \frac{r_{j+1} - r_{c(s)}}{r_j - r_{c(s)}}\right) \sum_{s' \in \mathbf{S}} \sum_{g=1}^{n+1} \sum_{\mathbf{k} \in G_{g-1}[c(s), n-1]} \Omega(m, r_j, \mathbf{k}) \times \\
& \Phi_{s'}(n-1, \mathbf{k}) p_{s, s'} \\
= & \left(\frac{r_{j+1} - r_{c(s)}}{r_j - r_{c(s)}}\right) \sum_{g=1}^{n+1} \sum_{\mathbf{k} \in G_g[c(s), n]} \Omega(m+1, r_j, \mathbf{k}) \Phi_s(n, \mathbf{k}) + \\
& \left(1 - \frac{r_{j+1} - r_{c(s)}}{r_j - r_{c(s)}}\right) \sum_{s' \in \mathbf{S}} \sum_{\mathbf{k} \in \mathcal{K}_{n-1}} \Omega(m, r_j, \mathbf{k}) \times \\
& \Phi_{s'}(n-1, \mathbf{k}) p_{s, s'}.
\end{aligned}$$

Da definição 5 (equação 4.66) temos  $\mathcal{K}_{n-1} = \bigcup_{g=0}^n G_g[c(s'), n-1]$ , para qualquer  $s' \in \mathcal{S}$ . Portanto,

$$\begin{aligned}
\Upsilon_s(n, m, j) &= \left(\frac{r_{j+1} - r_{c(s)}}{r_j - r_{c(s)}}\right) \sum_{g=1}^{n+1} \sum_{\mathbf{k} \in G_g[c(s), n]} \Omega(m+1, r_j, \mathbf{k}) \Phi_s(n, \mathbf{k}) + \\
& \left(1 - \frac{r_{j+1} - r_{c(s)}}{r_j - r_{c(s)}}\right) \sum_{s' \in \mathbf{S}} \sum_{g=0}^n \sum_{\mathbf{k} \in G_g[c(s'), n-1]} \Omega(m, r_j, \mathbf{k}) \times \\
& \Phi_{s'}(n-1, \mathbf{k}) p_{s, s'}.
\end{aligned}$$

Como  $\Phi_{s'}(n-1, \mathbf{k}) = 0$  para  $k_c(s') = 0$  (probabilidade de uma coloração  $\mathbf{k}$  dado que o primeiro estado visitado é  $s$  e o número de visitas ao estado  $s$  é zero), temos

$$\begin{aligned}
\Upsilon_s(n, m, j) &= \left(\frac{r_{j+1} - r_{c(s)}}{r_j - r_{c(s)}}\right) \sum_{g=1}^{n+1} \sum_{\mathbf{k} \in G_g[c(s), n]} \Omega(m+1, r_j, \mathbf{k}) \Phi_s(n, \mathbf{k}) + \\
& \left(1 - \frac{r_{j+1} - r_{c(s)}}{r_j - r_{c(s)}}\right) \sum_{s' \in \mathbf{S}} \sum_{g=1}^n \sum_{\mathbf{k} \in G_g[c(s'), n-1]} \Omega(m, r_j, \mathbf{k}) \times \\
& \Phi_{s'}(n-1, \mathbf{k}) p_{s, s'} \\
= & \left(\frac{r_{j+1} - r_{c(s)}}{r_j - r_{c(s)}}\right) \Upsilon_s(n, m+1, j) + \\
& \left(1 - \frac{r_{j+1} - r_{c(s)}}{r_j - r_{c(s)}}\right) \sum_{s' \in \mathbf{S}} \Upsilon_{s'}(n-1, m, j) p_{s, s'}.
\end{aligned}$$

□

**Teorema 3:** Seja  $r_{j+1} \leq r < r_j$  e  $\beta_j = (r - r_{j+1}) / (r_j - r_{j+1})$ , então

$$P[ACR(t) > r] = \sum_{n=0}^{\infty} e^{-\Lambda t} \frac{(\Lambda t)^n}{n!} \sum_{m=0}^n \binom{n}{m} \beta_j^m (1 - \beta_j)^{n-m} \sum_{s \in \mathbf{S}} \pi_s(0) \Upsilon_s(n, m, j). \quad (4.73)$$

**Prova.** Sabemos que

$$P[ACR(t) > r] = \sum_{n=0}^{\infty} e^{-\Lambda t} \frac{(\Lambda t)^n}{n!} \sum_{\mathbf{k} \in \mathcal{K}_n} \Phi(n, \mathbf{k}) P[ACR(t) > r | n, \mathbf{k}].$$

A partir das equações 4.7, 4.52 e 4.64 temos

$$P[ACR(t) > r] = \sum_{n=0}^{\infty} e^{-\Lambda t} \frac{(\Lambda t)^n}{n!} \sum_{\mathbf{k} \in \mathcal{K}_n} \sum_{s \in \mathbf{S}} \pi_s(0) \Phi_s(n, \mathbf{k}) EO_{>}[\vec{r}, \mathbf{k}, r].$$

Usando o Teorema 2 (equação 4.61) obtemos

$$\begin{aligned} P[ACR(t) > r] &= \sum_{n=0}^{\infty} e^{-\Lambda t} \frac{(\Lambda t)^n}{n!} \sum_{\mathbf{k} \in \mathcal{K}_n} \sum_{s \in \mathbf{S}} \pi_s(0) \Phi_s(n, \mathbf{k}) \times \\ &\quad \sum_{m=0}^n \binom{n}{m} \beta_j^m (1 - \beta_j)^{n-m} \Omega(m, r_j, \mathbf{k}) \\ &= \sum_{n=0}^{\infty} e^{-\Lambda t} \frac{(\Lambda t)^n}{n!} \sum_{m=0}^n \binom{n}{m} \beta_j^m (1 - \beta_j)^{n-m} \times \\ &\quad \sum_{s \in \mathbf{S}} \pi_s(0) \sum_{\mathbf{k} \in \mathcal{K}_n} \Phi_s(n, \mathbf{k}) \Omega(m, r_j, \mathbf{k}). \end{aligned}$$

Como  $\Phi_s(n, \mathbf{k})$  corresponde à probabilidade de uma coloração  $\mathbf{k}$  dadas  $n$  transições e o primeiro estado visitado é  $s$  e  $k_{c(s)}$  corresponde ao número de visitas ao estado  $s$ , temos  $\Phi_s(n, \mathbf{k}) = 0$  quando  $k_{c(s)} = 0$ , pois existe pelo menos uma visita ao estado  $s$ . Logo,

$$P[ACR(t) > r] = \sum_{n=0}^{\infty} e^{-\Lambda t} \frac{(\Lambda t)^n}{n!} \sum_{m=0}^n \binom{n}{m} \beta_j^m (1 - \beta_j)^{n-m} \times$$

$$\begin{aligned}
& \sum_{s \in \mathbf{S}} \pi_s(0) \sum_{g=1}^{n+1} \sum_{\mathbf{k} \in G_g[c(s), n]} \Phi_s(n, \mathbf{k}) \Omega(m, r_j, \mathbf{k}) \\
&= \sum_{n=0}^{\infty} e^{-\Lambda t} \frac{(\Lambda t)^n}{n!} \sum_{m=0}^n \binom{n}{m} \beta_j^m (1 - \beta_j)^{n-m} \times \\
& \quad \sum_{s \in \mathbf{S}} \pi_s(0) \Upsilon_s(n, m, r_j).
\end{aligned}$$

□

Podemos fazer duas observações em relação ao Teorema 3. Em primeiro lugar, a recursão de  $\Upsilon_s(n, m, j)$  apresentada em 4.69 é idêntica à recursão de  $b_j^j(n, m)$  apresentada na seção 4.2.5. Portanto, podemos afirmar que o Teorema 3 é igual à equação 4.22, ou seja, o Teorema 3 fornece a interpretação probabilística para o algoritmo apresentado em [54]. Em segundo lugar, para  $r = r_{j+1}$ ,  $j = 1, \dots, L$ , temos  $\beta_j = 0$  e

$$P[ACR(t) > r_{j+1}] = \sum_{n=0}^{\infty} e^{-\Lambda t} \frac{(\Lambda t)^n}{n!} \sum_{s \in \mathbf{S}} \pi_s(0) \Upsilon_s(n, 0, j).$$

Portanto, podemos eliminar os termos combinatórios apresentados em 4.73 (Teorema 3) se  $r \in \mathcal{R}$ . A vantagem em eliminarmos esses termos combinatórios é que passaremos a trabalhar apenas com valores positivos não maiores que 1. A idéia é então incluir a recompensa  $r$  no conjunto de recompensas  $\mathcal{R}$  se  $\exists r_i \in \mathcal{R}$ , tal que  $r_i = r$  para  $i = 1, \dots, L + 1$ . A seguir mostraremos a definição formal desta alteração no Teorema 3.

#### 4.4.2 Alteração na Recursão para $P[ACR(t) > r]$

O Lema 4 (equação 4.60) mostra uma recursão para combinação linear de estatísticas de ordem para o caso em que o valor  $r$  é igual a um dos valores das possíveis recompensas atribuídas aos estados, isto é,  $r \in \mathcal{R}$ .

O Teorema 2 (equação 4.61) mostra uma recursão para combinação linear de estatísticas de ordem cujo valor de  $r$  é qualquer ( $r_{j+1} \leq r < r_j$ ,  $j = 1, \dots, L + 1$ ).

Esta recursão foi usada no Teorema 3 (equação 4.73) para provar que a recursão obtida para  $P[ACR(t) > r]$  em [54] pode ser obtida através da metodologia de [49] que usa apenas argumento probabilístico.

Nesta seção mostraremos que:

- a recursão obtida do Lema 4 para  $r \in \mathcal{R}$  pode ser estendida para o caso em que  $r$  é qualquer. Isto é, a mesma recursão é válida tanto para  $r \in \mathcal{R}$  quanto para  $r \notin \mathcal{R}$ .
- a partir deste resultado, obteremos uma nova recursão para  $P[ACR(t) > r]$ . Essa nova recursão é diferente de [54], com a vantagem de ser ligeiramente mais simples por não incluir os termos combinatoriais da equação 4.73. Note que esta nova recursão evidencia a importância da interpretação probabilística, pois mostra que novas recursões para combinação linear de estatísticas de ordem podem ser usadas no cálculo de  $P[ACR(t) > r]$ .

Seja  $r^*$  uma recompensa tal que  $r^* \notin \mathcal{R}$  e  $r_{j+1} < r^* < r_j$ , para  $r_{j+1} \in \mathcal{R}$  e  $r_j \in \mathcal{R}$ ,  $j = 1, \dots, L$ . Podemos então definir  $\mathcal{R}^* = \mathcal{R} \cup \{r^*\} = \{r'_1, \dots, r'_{L+2}\}$ , onde  $r'_1 > r'_2 > \dots > r'_{L+1} > r'_{L+2} = 0$ . Além disso, seja  $\mathbf{k}^*$  um vetor idêntico ao vetor  $\mathbf{k}$  acrescido do elemento  $k^*$ , onde  $k^* = 0$ , pois não existe intervalo associado à recompensa  $r^*$ . Podemos então escrever  $\mathbf{k}^* = \langle k'_1, \dots, k'_{L+2} \rangle$ .

**Lema 6:** para  $n \geq 0$ ,  $m = 0, \dots, n$ ,  $j = 1, \dots, L + 1$ , seja

$$\Upsilon_s^*(n, m, j) = \sum_{\mathbf{k}^*: \mathbf{k} \in G_s[n]} \Phi_s(n, \mathbf{k}^*) \Omega(m, r'_j, \mathbf{k}^*). \quad (4.74)$$

Então, a recursão apresentada em 4.69 também é válida para  $\Upsilon_s^*(n, m, j)$ .

**Prova.** A recursão  $\Omega(m, r_j, \mathbf{k})$  é válida para qualquer valor de  $\mathbf{k}$ , independente da dimensão de  $\mathbf{k}$ . Da mesma forma, a recursão de  $\Phi[n, \mathbf{k}]$  também é válida, independente do número de recompensas, sejam elas associadas ou não a um estado. Portanto, a recursão para  $\Upsilon_s(n, m, j)$  também é válida substituindo  $\mathbf{k}$

por  $\mathbf{k}^*$ , assim como todas as interpretações probabilísticas já discutidas na seção anterior.  $\square$

É importante notar que quando  $r^*$  é adicionada ao conjunto de recompensas, temos um bloco a mais (veja Figura 4.4) em comparação com os cálculos feitos sem essa recompensa.

**Teorema 4:** Para  $l = 1, \dots, L + 1$  e para qualquer recompensa  $r_l \in \mathcal{R}^*$ ,

$$P[ACR(t) > r_l] = \sum_{n=0}^{\infty} e^{-\Lambda t} \frac{(\Lambda t)^n}{n!} \sum_{s \in \mathcal{S}} \pi_s(0) \Upsilon_s^*(n, 0, l - 1). \quad (4.75)$$

**Prova.** Como a recursão para  $\Upsilon_s(n, m, j)$  é válida pelo lema anterior, então é imediata a obtenção de 4.75.  $\square$

A partir da interpretação probabilística do algoritmo de [54] apresentada neste capítulo, o Teorema 4 foi utilizado em [61] para obter um limite superior e um limite inferior para a recompensa acumulada pelo sistema. Este resultado foi obtido da interpretação probabilística por nós obtida, mudando-se apenas as condições de contorno do algoritmo de cálculo da recompensa acumulada. No algoritmo de [61] foi definido

$$\Upsilon_s(n, m, j) = \sum_{g=1}^{n+1} \sum_{\mathbf{k} \in G_g[c(s), n]} \Gamma_s(n, \mathbf{k}) \Omega(m, r_j, \mathbf{k}),$$

ou seja, os vetores  $\mathbf{k}$  são agregados de acordo com o último estado visitado. A seguir vamos discutir um pouco este algoritmo. Iremos exemplificar o seu uso na próxima seção.

Suponha que no intervalo  $(0, t)$  o sistema acumula no mínimo uma recompensa  $r_l t$  e no máximo uma recompensa  $r_u t$ . O sistema possui recompensas positivas e negativas, onde  $r_l$  é maior que a maior recompensa negativa do modelo e  $r_u$  é menor que a menor recompensa positiva do modelo. Não existe recompensa com valor zero. Portanto,

$$r_{L+1} < r_L < \dots < r_l \leq 0 \leq r_u < \dots < r_2 < r_1. \quad (4.76)$$

Esta restrição foi definida em [61] como forma de inicializar o cálculo dos  $\Upsilon_s(n, m, j)$ . Suponha que queremos calcular  $P[CR(t) > r_j]$  onde  $r_l < r_j < r_u$ .

- $\Upsilon_s(n, 0, u + 1) = 1$  se  $r_{c(s)} > 0$ ,

pois a probabilidade do sistema acumular uma recompensa maior que  $r_u t$  dado que se encontra no estado  $s$  (que tem recompensa  $r_{c(s)} > 0$ ) é 1.

- $\Upsilon_s(n, n, u) = 0$  se  $r_{c(s)} < 0$

pois a probabilidade do sistema acumular uma recompensa maior ou igual a  $r_u t$  dado que se encontra no estado  $s$  (que tem recompensa  $r_{c(s)} < 0$ ) é zero. Note que se existisse uma recompensa  $r_{c(s)} \in \mathcal{R}$  tal que  $0 < r_{c(s)} < r_u$ , não teríamos o valor inicial de  $\Upsilon_s(n, n, u)$ , já que é possível acumularmos uma recompensa igual a  $r_u t$  antes da visita ao estado  $s$ . Este é o motivo da restrição feita pelo algoritmo aos limites  $r_u$  e  $r_l$  (equação 4.76).

Note que as recompensas  $r_l$ ,  $r_u$  e  $r_j$  não pertencem ao conjunto  $\mathcal{R}$  de recompensas. Entretanto, de acordo com o Lema 6 o algoritmo pode ainda ser aplicado. Além disso, na implementação do algoritmo de [61] precisamos calcular apenas  $\Upsilon_s(n, m, j)$  e  $\Upsilon_s(n, m, u)$ , para  $m = 0, \dots, n$ , pois temos os valores de  $\Upsilon_s(n, 0, j)$  quando  $r_{c(s)} > 0$  e de  $\Upsilon_s(n, n, u)$  quando  $r_{c(s)} < 0$ . A seguir reproduziremos um dos exemplos apresentados em [61] para este algoritmo.

### 4.4.3 Aplicação

Uma das medidas usada para definir a qualidade de serviço a ser oferecida aos usuários de uma rede de comunicação de dados é a taxa de perda de pacotes. Infelizmente, garantir uma taxa máxima (ou média) de perdas de pacotes para

uma conexão não é um serviço fácil de se implementar pois a perda de pacotes é causada principalmente por congestionamentos na rede. Isto significa que em um dado momento um nó da rede recebe mais pacotes do que é capaz de atender e portanto, pacotes são descartados. Por este motivo, é importante o desenvolvimento e o estudo de modelos sobre o comportamento dos nós da rede.

Existem na literatura vários artigos que estudam este problema. Entretanto, a maioria dos trabalhos apresentados se concentram em obter medidas em estado estacionário. O problema com este tipo de medidas é que elas mostram apenas o comportamento médio do sistema. Utilizando a análise transiente podemos investigar o sistema em situações críticas onde ocorrem as quedas na qualidade de serviço, como por exemplo, os momentos em que o *buffer* está cheio. A seguir discutiremos o exemplo apresentado em [61] de uma transmissão de vídeo sobre uma rede ATM. O nosso objetivo é estudar a ocupação do *buffer* em um comutador ATM.

Considere a transmissão de um tráfego de vídeo em uma rede ATM. As células são geradas por um servidor de vídeo e enviadas para o comutador ATM mais próximo. As células são então armazenadas no *buffer* do comutador ATM e transmitidas. A taxa de geração de células pelo servidor é variável com o tempo. Células que chegam no comutador e encontram o *buffer* cheio são descartadas.

O comportamento do sistema é modelado em [61] com dois objetos: uma fonte de dados e um servidor com uma fila (*buffer*). O processo de geração de células é representado por um conjunto de fontes do tipo *OnOff* [1] e é modelado por uma cadeia de Markov de nascimento e morte com  $M + 1$  estados, onde  $M$  corresponde ao número de fontes *OnOff*. A taxa em que uma fonte passa do estado *Off* para o estado *On* é  $\alpha$  (nascimento) e a taxa em que uma fonte passa do estado *On* para o estado *Off* é  $\beta$  (morte). A taxa de geração de células depende do número de fontes no estado *On* e é definida como  $n\lambda$  cel/ms,  $n = 0, \dots, M$ . O servidor possui capacidade de armazenamento de  $B$  células e taxa de serviço constante.



Na modelagem tradicional de cadeia de Markov para este exemplo, cada estado do modelo é caracterizado por duas variáveis: a primeira corresponde ao número de *buffers* ocupados e a segunda corresponde ao número de fontes no estado  $On$ . Portanto, a cadeia de Markov possui  $(B + 1) \times (M + 1)$  estados. A ocupação do *buffer*, em qualquer instante  $t$ , é dada pelo valor da primeira variável de estado.

No modelo markoviano com recompensas para este exemplo, os estados são caracterizados apenas por uma variável que representa o número de fontes no estado  $On$ . Portanto, a cadeia de Markov correspondente possui  $M + 1$  estados. A taxa de recompensa associada a um estado representa a taxa de variação da ocupação do *buffer* (taxa de chegada de células menos taxa de serviço do comutador). As taxas de recompensa são positivas quando a taxa de chegada de células é maior que a capacidade de serviço do canal, e são negativas nos outros casos. A ocupação do *buffer*, em qualquer instante  $t$ , é dada pela recompensa acumulada durante o intervalo  $(0, t)$ . Note que para este modelo a recompensa acumulada tem um valor mínimo (limite inferior) e um valor máximo possíveis (limite superior) que são, respectivamente, zero e  $B$ . (veja [62] e [63] para exemplos de outros modelos).

A Figura 4.5 (de [61]) mostra a ocupação do *buffer* durante 1000 transmissões simultâneas do filme *Starwars* calculados para os dois modelos discutidos acima. Os parâmetros usados neste exemplo são:  $t = 20\text{ms}$ ;  $M = 20$ ;  $B = 500$ ;  $\lambda = 50$  cel/ms;  $\alpha = 0.0712$ ;  $\beta = 0.1525$ ; taxa média de geração de células = 318.28 cel/ms; carga média = 0.77 (taxa média de geração de células / capacidade do comutador). O estado inicial ( $t = 0$ ) de todas as fontes é *Off*.

Note que o modelo com recompensas fornece uma boa aproximação para o modelo com fila. A grande vantagem do modelo com recompensas é que o estado da fila não é representado no modelo. Por esta razão, o modelo com recompensas possui apenas 21 estados, enquanto que o outro modelo possui mais de 10500 estados. Isto significa que a solução do modelo com recompensas não é afetada pelo tamanho do *buffer* do sistema analisado e podemos usá-lo na análise de

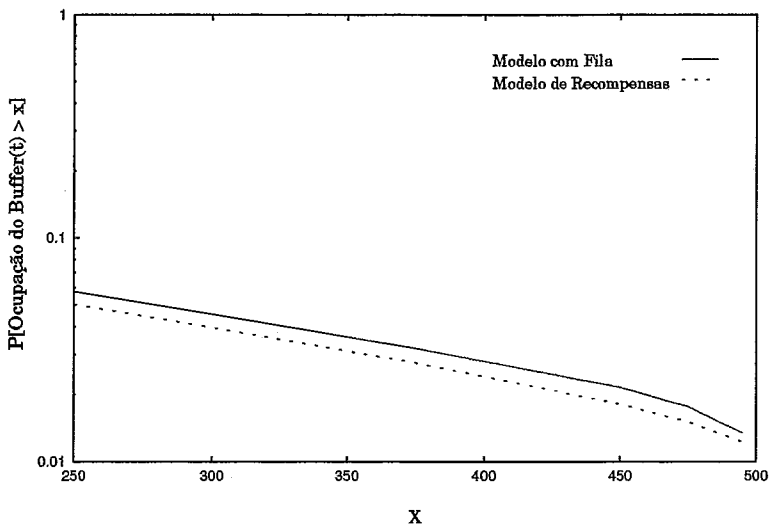


Figura 4.5: Ocupação do *Buffer*.

sistemas com dimensões reais.

## 4.5 Conclusões

Neste capítulo foram analisados dois algoritmos para o cálculo da distribuição da recompensa acumulada em um tempo finito. O primeiro algoritmo foi proposto por E. de Souza e Silva e H.Richard Gail em [52]. O principal problema deste algoritmo é trabalhar com números positivos e negativos, podendo haver perda de precisão dependendo dos parâmetros do modelo. Para evitar estes problemas, algumas soluções como agrupamento de recompensas são sugeridas em [52]. O segundo algoritmo foi proposto por Nabli e Sericola em [54]. Este algoritmo possui como grande vantagem só trabalhar com números positivos e não maiores que 1. O algoritmo entretanto não fornece uma interpretação probabilística para os termos calculados. Isto significa que os termos apresentados na solução não possuem um significado associado ao comportamento do sistema sendo modelado.

Este Capítulo mostrou que o algoritmo de [54] pode ser provado por argumentos probabilísticos usando a metodologia de [49]. A partir dessa prova matemática

foi desenvolvido um novo algoritmo para recompensa acumulada com limite e que foi apresentado em [61]. Também apresentamos nesse Capítulo dois novos algoritmos para o cálculo da distribuição da combinação linear de estatísticas de ordem. O interesse por esta medida pode também ser encontrada em outras áreas, como por exemplo, no estudo da teoria assintótica [58].

# Capítulo 5

## Conclusões

No Capítulo 2, dois modelos para o estudo do controle do *jitter* no destino foram propostos e analisados. O primeiro modelo armazena  $N_{min}$  pacotes antes de iniciar a entrega dos pacotes recebidos, e o segundo modelo espera por um tempo *fixo*  $T_1$  após a chegada do primeiro pacote antes de iniciar o processo de entrega. Os resultados apresentados mostram que é possível reduzir o *jitter* com o armazenamento de poucos pacotes no destino. Os dois modelos têm bom desempenho quando o desvio padrão do *jitter* não é muito grande. Para um tráfego com grande desvio padrão no *jitter*, o primeiro modelo tem bom desempenho mesmo para valores pequenos de  $N_{min}$ , enquanto os resultados do segundo modelo indicam que ele tem um desempenho inferior ao do primeiro esquema.

Podemos fazer três observações em relação aos modelos apresentados no Capítulo 2. Primeiro, os resultados da análise foram validados através de testes com uma ferramenta de comunicação de áudio. Segundo, a abordagem feita nesse capítulo é original pois do nosso conhecimento não existem na literatura outros modelos analíticos que obtenham a distribuição do *jitter* usando-se o controle feito apenas no destino. Terceiro, podemos usar os modelos apresentados neste capítulo para estudar outros problemas. Por exemplo, podemos estudar a eficácia do equipamento *proxy* em garantir uma melhor QoS nas transmissões

de áudio/vídeo [25]. Além disso, podemos usar os modelos para estudar como a reordenação de pacotes feita no nó de destino afeta o desempenho de aplicações que exigem o recebimento ordenado dos pacotes (veja discussão em [26]). Esta terceira observação sobre o Capítulo 2 faz parte dos trabalhos ainda a serem desenvolvidos.

No Capítulo 3, propomos uma aproximação para o método GTH de forma que ele possa ser usado na solução de matrizes esparsas com milhares de estados. O método proposto tem como base dois métodos de aproximação existentes na literatura ([31] e [32]). O novo método utiliza a idéia que probabilidades de transição com valores muito pequenos, podem ser eliminadas (ou redirecionadas) e mesmo assim termos uma boa aproximação para a solução exata do modelo. O objetivo em introduzir uma perturbação na matriz de probabilidades é diminuir o custo computacional da solução do modelo. Os resultados apresentados mostram que o novo algoritmo fornece uma boa aproximação para a solução de modelos markovianos e de modelos não markovianos. Sendo que apenas para os modelos markovianos foi obtido um limite para o erro cometido pelo novo algoritmo.

São propostas de trabalho a serem desenvolvidas a partir do estudo feito no Capítulo 3:

- ao usarmos o novo algoritmo na solução de modelos não markovianos, obtivemos um limite para as probabilidades estacionárias dos pontos embutidos, mas não conseguimos definir um *bound* para as probabilidades dos estados do modelo. Faz parte dos trabalhos futuros encontrar um *bound* para a solução de modelos não markovianos usando o novo algoritmo.
- definir um algoritmo *geral* para encontrar o grupo de estados que fazem parte da partição  $S_0$  do novo algoritmo (veja definição no Capítulo 3).
- usar o novo algoritmo em outros métodos de solução como, por exemplo, Power, SOR, Jacobi, etc.

No Capítulo 4 foram analisados dois algoritmos para o cálculo da distribuição da recompensa acumulada em um tempo finito apresentados em [52] e em [54]. Este capítulo mostrou uma interpretação probabilística para o algoritmo de [54] usando a mesma metodologia de [49]. A partir dessa prova matemática, mudando apenas condições de contorno, foi apresentado em [61] um novo algoritmo para recompensa acumulada com limites. Além disso, fizemos uma alteração no algoritmo de [54] e obtivemos um algoritmo mais simples que não apresenta os termos combinatórios do algoritmo de [54]. Também apresentamos nesse capítulo dois novos algoritmos para o cálculo da distribuição da combinação linear de estatísticas de ordem. O interesse por esta medida pode também ser encontrada em outras áreas, como por exemplo, no estudo da teoria assintótica [58] e testes de Hipóteses e Estimação Estatística [55].

Vimos que o algoritmo proposto em [54] tem como grande vantagem só trabalhar com números positivos e não maiores que 1. O problema é que este algoritmo trabalha apenas com taxas de recompensa. Por outro lado, o algoritmo apresentado em [52] permite definições de cadeias de Markov com taxas de recompensa e/ou recompensas por impulso. O problema é que este algoritmo trabalha com números negativos e maiores que 1. Dando prosseguimento ao estudo de cadeias de Markov com recompensas queremos incluir recompensas por impulso no algoritmo de [54] usando a metodologia de [49].

# Bibliografia

- [1] Bae, J. J., Suda, T., "Survey of Traffic Control Schemes and Protocols in ATM Networks", *Proceedings of the IEEE*, vol.79, n.2, pp. 170-189, February 1991.
- [2] Partridge, C., *Gigabit Networking*, Massachusetts, Addison-Wesley Publishing Company, 1993.
- [3] Golestani, S. J., "A Stop-and-Go Queueing Framework for Congestion Management", *ACM SIGCOMM*, pp. 8-18, September 1990.
- [4] Guillemin, F., Roberts, J., "Jitter and Bandwidth Enforcement". In:*IEEE GLOBECOM*, pp. 261-265, 1991.
- [5] Roberts, J., Guillemin, F., "Jitter in ATM Networks and its Impact on Peak Rate Enforcement", *Performance Evaluation*, v.16, pp. 35-48, 1992.
- [6] Guillemin, F., Monin, W., "Management of Cell Delay Variation in ATM Networks". In:*IEEE GLOBECOM*, pp. 128-132, 1992.
- [7] Landry, R., Stavrakakis, I., "A Queueing Study of Peak-Rate Enforcement for Jitter Reduction in ATM Networks". In: *IEEE GLOBECOM*, pp. 619-623, 1994.
- [8] Verma, D., Ferrari, D., "Delay Jitter Control for Real-Time Communication in a Packet Switching Network", *Proc. TriComm*, pp. 35-43, 1991.
- [9] Ferrari, D., "Delay Jitter Control Scheme in Packet-Switching Internetworks", *Computer Communications*, V. 15, N. 6, july/august 1992.

- [10] Zhang, H., Ferrari, D., "Rate Controlled Static-Priority Queueing". In: *IEEE INFOCOM*, pp. 227-236, 1993.
- [11] Kröner, H., Eberspächer, M., Theimer, T.H., Kühn, P.J., Briem, U., "Approximate Analysis of the End-to-End Delay in ATM Networks". In: *IEEE INFOCOM*, pp. 978-986, 1992.
- [12] Matragi, W., Bisdikian, C., Sohraby, K., "On the Jitter and Delay Analysis in ATM Multiplexer". In: *IEEE ICC*, pp. 738-744, 1994.
- [13] Matragi, W., Bisdikian, C., Sohraby, K., "Jitter Calculus in ATM Networks: Single Node Case". In: *IEEE INFOCOM*, pp. 232-241, 1994.
- [14] Matragi, W., Bisdikian, C., Sohraby, K., "Jitter Calculus in ATM Networks: Multiple Node Case". In: *IEEE INFOCOM*, pp. 12-16, 1994.
- [15] Bisdikian, C., Matragi, W., Sohraby, K., "A Framework for Jitter Analysis in Cell Based Multiplexers". *Performance Evaluation*, pp. 257-277, V. 22, 1995.
- [16] Figueiredo, D. R., Duarte, F. P., de Souza e Silva, E., "Implementação de Um Aplicativo de Voz com Análise e Caracterização de seu Tráfego (A Voice Tool and its Traffic Analysis)". In: *XXIV Symposium of the Brazilian Computer Society*, pp.531-540, agosto de 1997.
- [17] de Souza e Silva, E., Gail, H. R., Muntz, R. R., "Efficient Solutions for a Class of Non-Markovian Models". *Computations with Markov Chains*, 1 ed., Kluwer Academic Publishers, pp. 483-506, 1995.
- [18] Berson, S., de Souza e Silva, E., Muntz, R. R., *Numerical Solution of Markov Chains*, chapter: An Object Oriented Methodology for the Specification of Markov Models, pp. 11-36, Marcel Dekker, 1991.
- [19] Diniz, M. C., de Souza e Silva, E., "Ferramenta de Análise para uma Classe de Modelos Não Markovianos e Aplicações a Redes de Alta Velocidade". In: *SBC'95/Panel'95, Anais I*, pp. 275-286, 1995.



- [20] Carmo, R. M. L. R., Carvalho, L. R. de, de Souza e Silva, E., Diniz, M. C., Muntz, R. R., "TANGRAM-II: A Performability Modeling Environment Tool". In: *Proceedings of the 9th International Conference of Performance Evaluation Modelling Techniques and Tools*, pp 6-18, France, junho de 1997.
- [21] Carmo, R. M. L. R., Carvalho, L. R. de, de Souza e Silva, E., Diniz, M. C., Muntz, R. R., "Performance/availability modeling with the TANGRAM-II modeling Environment", *Performance Evaluation*, n.740, pp. 1-20, 1998.
- [22] LI, S., "Study of Information Loss in Packet Voice Systems", *IEEE Transactions on Communications*, v.37, N.11, pp.1192-1202, November 1989.
- [23] Diniz, M. C., de Souza e Silva, E., "Análise de Mecanismos de Controle de Jitter em Redes ATM". In: *SEMISH-SBC*, pp.203-214, 1996.
- [24] Diniz, M. C., de Souza e Silva, E., "Models for Jitter Control at Destination", In: *Proceedings of IEEE-ITS*, pp.118-122, 1996.
- [25] Sen, S., Rexford, J., Towsley, D., "Proxy Prefix Caching for Multimedia Streams". In: *IEEE INFOCOM*, 9D-4, 1999.
- [26] Jean-Marie, A., Tidball, M., Escalante, M., Leoni, V., Ponce de León, H., "On the Influence of Resequencing on the Regularity of Service". *Performance Evaluation*, n.36-37, pp. 115-135, 1999.
- [27] de Souza e Silva, E., Gail, H. R., "Transient Solutions for Markov Chains". *Computational Probability*, editor: W. Grassmann, pp. 43-81, Kluwer Academic Publishers, 2000.
- [28] de Souza e Silva, E., Muntz, R. R., *Métodos Computacionais de Solução de Cadeias de Markov: Aplicações a Sistemas de Computação e Comunicação*. VIII Escola de Computação, 1992.
- [29] Grassmann, W., Taksar, M., Heyman, D., "Regenerative Analysis and Steady State Distributions for Markov Chains", *Operation Research*, V. 33, N.5, pp. 1107-1116, 1985.

- [30] Golub, G. H., Loan, C. F. V., *Matrix Computation*, 2 ed., The John Hopkins University Press, 1989.
- [31] Franceschinis, G., Muntz, R. R., "Bounds for Quasi-lumpable Markov Chains", *Performance Evaluation*, 20(1994), pp. 223-243.
- [32] de Souza e Silva, E., Ochoa, P. M., "State Space Exploration in Markov Models", *Performance Evaluation Review*, V. 20, N. 1, pp. 152-166, June 1992.
- [33] Jensen, A., "Markoff Chains as an Aid in the Study of Markoff Processes", *Skandinavisk Aktuarietidskrift*, V. 36, pp. 87-91, 1953.
- [34] de Souza e Silva, E., Gail, H. R., "The Uniformization Method in Performability Analysis". In R. Marie, B. Haverkort, G. Rubino, Nico Van Dijk, and K. Trivedi, editors, *Second International Workshop on Performability Modeling of Computer and Communication Systems*, França, Junho 1993.
- [35] Meo, M., de Souza e Silva, E., Marsan, M. A., "Efficient Solution for a Class of Markov Chain Models of Telecommunication Systems", *Performance Evaluation*, 27&28, pp. 603-625, 1996.
- [36] Courtois, P. J., Semal, P., "Bounds for the Positive Eigenvectors of Nonnegative Matrices and for their Approximations", *Journal of the ACM*, V.31, pp. 804-825, 1984.
- [37] Courtois, P. J., Semal, P., "Computable Bounds for Conditional Steady-state Probabilities in Large Markov Chains and Queueing Models, *IEEE Journal on Selected Areas in Communications*, SAC-4(6), pp. 926-937, 1986.
- [38] Muntz, R. R., de Souza e Silva, E., Goyal, A., "Bounding Availability of Repairable Computer Systems". *IEEE Transactions on Computers*, V. 38, N. 12, pp. 1714-1723, Dezembro 1989.

- [39] Lui, J. C. S., Muntz, R. R., "Computing Bounds on Steady State Availability of Repairable Computer Systems", *Journal of the ACM*, V. 41, N. 4, pp. 676-707, July 1994.
- [40] Kaufman, L., "Matrix Methods for Queueing Problems", *SIAM Journal Sci. Stat. Comput.*, V. 4, N. 3, pp. 525-552, September 1983.
- [41] Sriram, K., "Dynamic bandwidth allocation and congestion control schemes for voice and data multiplexing in wideband and packet technology". In: *ICC-90*, pages 1003-1009, 1990.
- [42] Takagi, H., *Analysis of Polling Systems*, MIT Press, 1986.
- [43] de Souza e Silva, E., Gail, H. R., Muntz, R. R., "Polling Systems with Server Timeouts", *IEEE/ACM Transactions on Networking*, V. 3, N. 5, pp. 560-575, 1995.
- [44] de Souza e Silva, E., Gail, H. R., Muntz, R. R., "Gated Time-Limited Polling Systems", *Proceedings of the International Conference on the Performance and Management of Complex Communication Networks*, pp. 251-270, Japan, November 1997.
- [45] de Souza e Silva, E., Gail, H. R., Golubchik, L., Lui, J. C. S., "Analytical Models for Mixed Workload Multimedia Storage Servers". *Performance Evaluation*, 36-37 (1999), 185-211.
- [46] Iyer, B. R., Donatiello, L., Heidelberger, P., "Analysis of Performability for Stochastic Models of Fault-Tolerant Systems", *IEEE Trans. on Computers*, V.C-35, N.10, pp. 902-907, October 1986.
- [47] de Souza e Silva, E., Gail, H. R., "Calculating Cumulative Operational Time Distributions of Repairable Computer Systems", *IEEE Trans. on Computers*, V.c-35, N.4, pp. 323-332, April 1986.

- [48] Smith, R.M., Trivedi, K. S., Ramesh, A. V., "Performability Analysis: measures, an algorithm, and a case study", *IEEE Trans. on Computers*, V.C-37, N.4, pp. 406-417, April 1988.
- [49] de Souza e Silva, E., Gail, H. R., "Calculating Availability and Performability Measures of Repairable Computer Systems Using Randomization", *Journal ACM*, V.36, N.1, pp. 171-193, January 1989.
- [50] Donatiello, L., Grassi, V., "On Evaluating the Cumulative Performance Distribution of Fault-Tolerant Computer Systems", *IEEE Trans. on Computers*, V.40, N.11, pp. 1301-1307, November 1991.
- [51] de Souza e Silva, E., Gail, H. R., *Calculating Transient Distributions of Cumulative Reward*, Technical Report, UCLA CDS-930033, September 1993.
- [52] de Souza e Silva, E., Gail, H. R., Campos, R. V., "Calculating Transient Distributions of Cumulative Reward". In: *Proceedings of Sigmetrics'95 - Performance'95*, pp. 231-240, Ottawa, Canada, Maio 1995.
- [53] de Souza e Silva, E., Gail, H. R., *An Algorithm to Calculate Transient Distributions of Cumulative Rate and Impulse Based Reward*, Technical Report, UCLA CDS-940021, May 1994.
- [54] Nabli, H., Sericola, B., *Performability Analysis of Fault-Tolerant Computer System*, Technical Report, INRIA, N. 2254, May 1996.
- [55] David, H. A., *Order Statistics*. John Wiley & Sons, 1981.
- [56] Dempster, A. P., Kleyale, M., "Distributions Determined by Cutting a Simplex with Hiperplanes", *The Annals of Mathematical Statistics*, V.39, N.5, pp. 1473-1478, 1968.
- [57] Weisberg, H., "The Distribution of Linear Combinations of Order Statistics from the Uniform Distribution", *The Annals of Mathematical Statistics*, V.42, N.2, pp. 704-709, 1971.

- [58] Matsunawa, T., "The Exact and Aproximate Distributions of Linear Combinations of Selected Order Statistics from a Uniform Distribution", *Ann. Inst. Statist. Math*, V.37, pp. 1-16, 1985.
- [59] Ramalingam, T., "Symbolic Computing The Exact Distributions of L-Statistics from a Uniform Distribution", *Ann. Inst. Statist. Math*, V.41, N.4, pp. 677-681, 1989.
- [60] Ross, S. M., *Introduction to Probability Models*. Academic Press, INC. 4 ed., 1989.
- [61] Fisch de Brito, C. E., *Medidas Transientes para Recompensa Acumulada Limitada: Novas Técnicas de Solução e Aplicações para Redes de Alta Velocidade*. Tese de Mestrado, COPPE-Sistemas, UFRJ, Julho de 1999.
- [62] Robert, S., Boudec, J. On a Markov Modulated Chain Exhibiting Self-similarities over Finite Timescale. *Performance Evaluation*, 27:159–173, 1996.
- [63] Leão, R.M.M., de Souza e Silva, Edmundo, Lucena S. C., A Set of Tools for Traffic Modeling, Analysis and Experimentation Tools'2000, a ser publicado.