



## INFERINDO A QUALIDADE DE SERVIÇO EM REDES VIA APRENDIZADO POR REFORÇO

Nicolas André Alves da Silva

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientador: Rosa Maria Meri Leão

Rio de Janeiro  
Março de 2021

INFERINDO A QUALIDADE DE SERVIÇO EM REDES VIA APRENDIZADO  
POR REFORÇO

Nicolas André Alves da Silva

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Orientador: Rosa Maria Meri Leão

Aprovada por: Prof. Rosa Maria Meri Leão  
Prof. Edmundo Albuquerque de Souza e Silva  
Prof. Daniel Sadoc Menasché

RIO DE JANEIRO, RJ – BRASIL  
MARÇO DE 2021

Silva, Nicolas André Alves da

Inferindo a qualidade de serviço em redes via  
aprendizado por reforço/Nicolas André Alves da Silva. –  
Rio de Janeiro: UFRJ/COPPE, 2021.

XI, 56 p.: il.; 29, 7cm.

Orientador: Rosa Maria Meri Leão

Dissertação (mestrado) – UFRJ/COPPE/Programa de  
Engenharia de Sistemas e Computação, 2021.

Referências Bibliográficas: p. 53 – 56.

1. Redes de acesso residencial. 2. Previsão de taxa  
de perda de pacotes. 3. Aprendizado por reforço. I.  
Leão, Rosa Maria Meri. II. Universidade Federal do Rio  
de Janeiro, COPPE, Programa de Engenharia de Sistemas  
e Computação. III. Título.

# Agradecimentos

Gostaria de agradecer aos meus pais, Andréa e Nelson, que sempre me apoiaram e me deram todas as oportunidades de estudar; a minha esposa Camila por todo incentivo dado em momentos difíceis e por sempre ter acreditado em mim; aos meus irmãos Junior e Davi pelo companheirismo desde o início dos meus estudos; e a toda minha família pelo apoio e suporte na vida.

Agradeço à minha orientadora, prof. Rosa, que tanto me ensinou ao longo de todo o curso de mestrado; ao prof. Edmundo e aos demais professores e alunos do laboratório LAND/COPPE que me auxiliaram muito nas reuniões de projeto; e ao programa PESC que forneceu todo o suporte para a realização das disciplinas, eventos e palestras que contribuíram para a minha formação acadêmica no curso de mestrado.

Por fim, a todos aqueles que contribuíram, direta ou indiretamente, para a realização desta dissertação de mestrado, o meu sincero agradecimento.

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

## INFERINDO A QUALIDADE DE SERVIÇO EM REDES VIA APRENDIZADO POR REFORÇO

Nicolas André Alves da Silva

Março/2021

Orientador: Rosa Maria Meri Leão

Programa: Engenharia de Sistemas e Computação

O aumento do número de usuários da internet e o crescimento da quantidade de tráfego consumido por cada um deles, têm gerado nos últimos anos, uma demanda por melhorias no gerenciamento da qualidade de serviço (QoS) de provedores de serviço de Internet (ISPs). O principal objetivo deste trabalho é modelar as condições da rede de um ISP e inferir se uma mudança na condição da rede do ISP permanecerá em instantes futuros. Desta forma, o sistema de gerenciamento do ISP poderá ser acionado quando houver previsão de que a degradação da condição da rede se manterá por um certo tempo. A partir de séries temporais de taxas de perda de pacotes coletadas em roteadores domésticos do ISP, é definido um modelo de Markov oculto (HMM) e os estados do modelo são mapeados em condições da rede do ISP. Então, uma técnica de Aprendizado por Reforço é usada nas séries temporais de condições da rede, para inferir o desempenho da rede no futuro e, possivelmente, indicar o acionamento de um alarme de manutenção. A partir do resultado do algoritmo de Aprendizado por Reforço, o gerenciador da rede pode acionar ações de manutenção preditiva ou mudanças nos parâmetros de operação da rede. Resultados obtidos utilizando um dataset real de um ISP mostram bom desempenho do método através das medidas de acurácia, precisão, *recall* e *f1-score*.

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

## INFERRING NETWORK QUALITY OF SERVICE VIA REINFORCEMENT LEARNING

Nicolas André Alves da Silva

March/2021

Advisor: Rosa Maria Meri Leão

Department: Systems Engineering and Computer Science

The increase in the number of Internet users and the growth in the amount of traffic consumed by each of them, has generated in recent years, a demand for improvements in the quality of service (QoS) provided by Internet service providers (ISPs). The main objective of this work is to model the network conditions of an ISP and to infer if a change in these conditions will remain in the future. In this way, the ISP's management system can be activated when it is predicted that the degradation of the network condition will remain for a certain time. A hidden Markov model (HMM) is defined from time series of packet loss rates collected on ISP's home routers, and the model states are mapped to ISP network conditions. Then, a Reinforcement Learning technique is used in the time series of network conditions, to infer the performance of the network in the future and, possibly, indicate the triggering of a maintenance alarm. From the result of the Reinforcement Learning algorithm, the network manager can trigger maintenance actions or changes in network operating parameters. Results obtained using a real ISP dataset show good performance of the method through measures of accuracy, precision, recall and f1-score.

# Sumário

<b>Lista de Figuras</b>	<b>ix</b>
<b>Lista de Tabelas</b>	<b>xi</b>
<b>1 Introdução</b>	<b>1</b>
1.1 Motivação . . . . .	1
1.2 Objetivo . . . . .	2
1.3 Contribuições . . . . .	2
1.4 Organização . . . . .	3
<b>2 Revisão Bibliográfica</b>	<b>5</b>
2.1 Séries temporais e detecção de suas anomalias . . . . .	5
2.2 Aprendizado por reforço aplicado a redes . . . . .	8
2.3 Relacionando com a dissertação . . . . .	9
<b>3 Conceitos Básicos</b>	<b>11</b>
3.1 Cadeia de <i>Markov</i> oculta (HMM) . . . . .	11
3.2 Aprendizado por Reforço . . . . .	14
<b>4 Metodologia Geral</b>	<b>20</b>
4.1 1ª etapa do método: Coleta dos dados . . . . .	22
4.2 2ª etapa do método: HMM e algoritmo de <i>Viterbi</i> . . . . .	24
4.3 3ª etapa do método: MDP e algoritmo <i>Value Iteration</i> . . . . .	30
<b>5 Experimentos e análise de resultados</b>	<b>39</b>
5.1 Métricas de avaliação de desempenho . . . . .	39
5.2 Experimentos . . . . .	41
5.2.1 Experimento 1 . . . . .	43
5.2.2 Experimento 2 . . . . .	46
5.3 Análise de Resultados . . . . .	49
<b>6 Conclusão</b>	<b>51</b>





# Lista de Figuras

3.1	Exemplo de estados ocultos da HMM e respectivas observações . . . . .	12
3.2	Exemplo de cadeia de <i>Markov</i> oculta . . . . .	13
3.3	Exemplo de MDP: arestas azuis correspondem a $a=0$ e vermelhas, $a=1$ . 15	
3.4	MDP representando a interação entre agente e ambiente. . . . .	16
4.1	Etapas da Metodologia . . . . .	20
4.2	Infraestrutura de medição . . . . .	22
4.3	Esquema do modelo de <i>Markov</i> oculto utilizado . . . . .	25
4.4	Distribuição da prob. de taxas de perda de pacotes no estado oculto $x_1$	25
4.5	Distribuição da prob. de taxas de perda de pacotes no estado oculto $x_2$	26
4.6	Distribuição da prob. de taxas de perda de pacotes no estado oculto $x_3$	26
4.7	Distribuição da prob. de taxas de perda de pacotes no estado oculto $x_4$	26
4.8	Série temporal de taxa de perda mapeada em estados ocultos do HMM	27
4.9	Simplificação da série de estados ocultos em 2 condições de rede . . .	28
4.10	Nova cadeia de <i>Markov</i> oculta utilizada na metodologia. . . . .	29
4.11	Distribuição da prob. de taxas de perda de pacotes no estado $x_b$ . . .	29
4.12	Distribuição da prob. de taxas de perda de pacotes no estado $x_r$ . . .	29
4.13	Exemplo de vetor $v^{(t)}$ numa série temporal de condições de rede. . . .	31
4.14	Exemplo de uma transição entre dois vetores adjacentes $\tilde{v}_i$ e $\tilde{v}_j$ numa série . . . . .	33
5.1	Exemplo de série temporal fictícia do tipo 1 . . . . .	40
5.2	Exemplo de série temporal fictícia do tipo 2 . . . . .	40
5.3	Exemplo de uma série temporal usada no experimento 1. . . . .	43
5.4	Histograma do número de eventos em cada série do <i>dataset</i> do experi- mento 1 . . . . .	44
5.5	Histograma da duração dos eventos das séries do <i>dataset</i> do experi- mento 1 . . . . .	44
5.6	Exemplo de uma série temporal usada no experimento 2. . . . .	46
5.7	Histograma do número de eventos em cada série no <i>dataset</i> do experi- mento 2 . . . . .	47

5.8	Histograma da duração dos eventos das séries do experimento 2. . . .	47
-----	--	----

# Lista de Tabelas

5.1	Tabela ou matriz de confusão . . . . .	45
5.2	Medidas de desempenho do método . . . . .	45
5.3	Tabela ou matriz de confusão . . . . .	48
5.4	Medidas de desempenho do método . . . . .	49

# Capítulo 1

## Introdução

### 1.1 Motivação

O aumento do número de usuários da internet e o crescimento da quantidade de tráfego consumido por cada um deles têm gerado, nos últimos anos, uma demanda de melhorias no gerenciamento da qualidade de serviço (QoS) de provedores de serviço de Internet (ISPs).

Segundo o relatório de acompanhamento do setor de telecomunicações emitido pela Agência Nacional de Telecomunicações (ANATEL) referente ao terceiro trimestre de 2019 [1], constata-se que há ainda um crescimento considerável no número percentual de usuários da internet, considerando uma análise nacional e internacional. No ano de 2008, por exemplo, este percentual mundial era de 23,1%, sofrendo uma evolução para em torno de 57,0% em 2019. Os valores dessa estatística variam bastante de acordo com a classificação de países desenvolvidos e subdesenvolvidos, apesar de que em ambos os casos a tendência continua crescente. O Brasil, como um país emergente, já possuía um percentual de 51,0% em 2013. Em 2019, em torno de 72,7% dos brasileiros possuem acesso à internet.

Simultaneamente, há também um crescimento do número de empresas de pequeno porte no *market share* brasileiro de prestadoras de serviço de banda larga fixa, segundo [1]. Em 2013, elas representavam 9,6% desse *market share*. Em 2019, porém, elas representam 29,0%.

Sendo assim, para sobreviver neste mercado complexo e competitivo, os ISPs de pequeno porte devem se preocupar ainda mais com o gerenciamento de desempenho de suas redes. Eles devem garantir uma aceitável QoS a seus clientes e, concomitantemente, devem pensar em otimização de custos da prestação de serviço.

## 1.2 Objetivo

Uma das principais métricas de desempenho de rede capazes de estimar a QoS de um serviço da Internet é a taxa de perda de pacotes. Na literatura, encontra-se uma vasta gama de artigos relacionados a esta análise, tais como [2], [3], [4], [5] e [6].

A medição de taxa de perda de pacotes se mostra importante para estimar a QoS da rede e a inferência dessa taxa futura pode permitir a sinalização de um problema num futuro próximo e possibilita uma rápida reação para mitigar o problema. Por exemplo, o ISP terá mais tempo para realizar procedimentos de resolução de problemas nessa rede antes que estes se tornem mais críticos.

Portanto, o principal objetivo deste trabalho é modelar as condições de rede de um ISP utilizando séries temporais de taxas de perda de pacotes coletadas nos roteadores domésticos dos seus clientes e, com isso, poder inferir probabilisticamente se será duradoura uma mudança nessas condições de rede em instantes futuros. Desta forma, um sistema de manutenção deste ISP poderá ser acionado não somente pela constatação de que a rede passou de uma condição boa para uma condição ruim, mas também pela previsão de que esta mudança se manterá por um certo tempo.

O método, que será proposto nos próximos capítulos, também irá considerar a limitação diária de recursos do sistema de manutenção do ISP. Portanto, este sistema pode decidir entre a ação de acionar uma reparação na rede ou a ação de não realizar nenhum procedimento. Esta ação se baseia na inferência de condições ruins de rede para os próximos instantes e no número de acionamentos já realizados no dia.

## 1.3 Contribuições

Este trabalho é baseado em [7]. Nele, os autores descrevem um método usando técnicas de aprendizado de máquina não supervisionado para fazer uma análise e inferir a qualidade de experiência (QoE) sentida pelos usuários da rede. Tal método utiliza conceitos de cadeia de *Markov* oculta (HMM).

Assim, como neste trabalho, a aplicação do método apresentado em [7] é exemplificada usando um conjunto de dados reais contendo medições realizadas por roteadores residenciais instalados em clientes de um ISP de médio porte. Ao final de alguns dias de coleta de dados, cada roteador possui uma série temporal de medidas de taxa de perda de pacotes por minuto. Então, utilizando HMM, foram obtidas sequências de estados ocultos correspondentes a cada medida. Tais estados ocultos foram associados a possíveis condições da rede a cada minuto. Sendo assim, alguns desses estados ocultos podem ser associados a uma condição de rede ruim (níveis altos de taxa de perda de pacotes) enquanto outros deles podem ser associados a uma condição de rede dita boa (níveis baixos de taxa de perda de pacotes).

Desse modo, analisando essas sequências de estados ocultos, os autores de [7] realizam a inferência da QoE de cada cliente ou de determinada sub-rede (região) do ISP. Eles demonstram, ainda, que os resultados de suas inferências de QoE na rede estão correlacionados com o número de chamados técnicos realizados para o serviço de *callcenter* do ISP.

A principal contribuição desta dissertação é a aplicação de uma técnica de *Reinforcement Learning* (RL) nas séries temporais de estados ocultos, geradas pela modelagem de [7], para inferir o nível de qualidade da rede em instantes futuros. Com isso, o gerenciador da rede pode otimizar as ações de manutenção preditiva na rede ou mudanças de modo de operação.

Uma metodologia que usa uma modelagem HMM seguida de MDP e RL também pode ser definida como mais uma contribuição desta dissertação. Esta sequência de modelagens também pode ser utilizada na solução de outros problemas.

Uma característica diferenciada deste trabalho é o uso de dados reais nos experimentos. O conjunto de dados é composto por 68849 séries temporais de medidas de taxa de perda de pacotes por minuto. As séries foram coletadas em roteadores domésticos entre os dias 01/10/2018 e 31/10/2018. O método é aplicável tanto para uma análise em alto nível da qualidade da rede do provedor quanto para análises direcionadas usando filtragem de dados por data ou por usuário.

Uma outra contribuição é a possibilidade de implementação e adaptação da técnica de RL utilizada neste trabalho para outros tipos de séries temporais de medidas de QoS, como por exemplo, medidas de vazão e de retardo em transmissões de dados. A ideia principal deste trabalho é diminuir o custo de ações relacionadas a manutenção e gerenciamento de redes.

## 1.4 Organização

Para melhor compreensão do texto, segue uma lista de como está estruturado este trabalho de dissertação, a partir deste capítulo. Logo, em cada item, abaixo, há um breve resumo relacionado ao conteúdo presente em cada capítulo:

- No próximo capítulo, Revisão Bibliográfica, serão descritos trabalhos nos quais os autores utilizam métodos de RL e técnicas de detecção de pontos de mudança em séries temporais;
- No capítulo seguinte, Conceitos Básicos, será feita uma breve explanação do conhecimento necessário para o entendimento da técnica que será apresentada. Tais conceitos se tratam de cadeia de *Markov* oculta (HMM), *Markov Decision Process* (MDP) e *Reinforcement Learning* (RL);

- A Metodologia Geral do trabalho é descrita no Capítulo 4. São apresentadas as etapas de coleta do *dataset* de taxa de perda de pacotes, de modelagem das séries de taxa de perda usando HMM e de inferência utilizando RL;
- No Capítulo 5, são apresentados resultados de experimentos com dados reais e sintéticos. Em seguida são descritos os trabalhos futuros e as conclusões deste trabalho.

# Capítulo 2

## Revisão Bibliográfica

Este capítulo tem o objetivo de descrever trabalhos relacionados a esta dissertação. Os assuntos dos trabalhos estão divididos em dois tópicos descritos abaixo.

No primeiro tópico deste capítulo, estão apresentados trabalhos relacionados com detecção de anomalias ou pontos de mudança em séries temporais de medidas de desempenho de rede. Dentre os trabalhos citados neste tópico, podemos encontrar um breve resumo do método que serviu de solução para a primeira parte da metodologia geral desta dissertação, além de outros métodos semelhantes que também possuem esta mesma finalidade.

No segundo tópico deste capítulo, estão descritos alguns trabalhos que apresentam técnicas de *Reinforcement Learning* (RL) aplicadas em análises de redes. Revisaremos *Markov Decision Process* (MDP) e *Q-Learning*, ambas relacionadas com a técnica de inferência encontrada na segunda parte da metodologia geral desta dissertação.

### 2.1 Séries temporais e detecção de suas anomalias

Os autores de [7] utilizam um HMM para modelar a perda de pacotes de um ISP do estado do Rio de Janeiro. Para realizar esta modelagem, em [7], foram utilizadas séries temporais com a taxa de perda de pacotes de usuários de um ISP, ao longo de 47 dias. Estas medidas representam as observações do modelo HMM utilizado em [7]. A partir das observações e dos estados ocultos, estima-se a distribuição da taxa de perda associada a cada estado. Com base nas distribuições obtidas durante o período de treinamento, é proposta a identificação dos estados ocultos com condições simples de QoS de rede.

Após a modelagem utilizando HMM, o trabalho [7] emprega o algoritmo de *Viterbi* para obter a sequência de estados ocultos, cada um representando uma condição de rede, a partir de uma série de taxas de perda de pacotes de um cliente. Com a sequência de condições da rede, é possível analisar os instantes em que os



usuários da rede presenciaram uma condição ruim de rede, ou seja, instantes em que provavelmente um usuário obteve alta taxa de perda de pacotes.

A finalidade de [7] é inferir problemas na rede do ISP. O trabalho utiliza a correlação espaço-temporal das séries de condições de rede de clientes conectados em determinados nós da rede com o objetivo de agrupar as regiões da rede com padrões estatísticos similares.

A avaliação da eficácia da técnica de [7] é feita através da identificação de condições de rede em dias antes e depois da realização de manutenções na rede do ISP. Outra avaliação do trabalho realizada pelos autores foi a coleta de informações de chamados técnicos realizados pelos clientes do ISP que notificam problemas na sua conexão à rede. Essa análise dos chamados mostrou que tais notificações dos clientes são majoritariamente relacionadas às regiões, identificadas pelo método, que apresentam condições ruins de rede.

Esta dissertação é baseada no trabalho de [7] porque ela usa o mesmo modelo HMM para obter as sequências de estados ocultos a partir das séries de perdas. Todavia, nesta dissertação, será dado um enfoque na previsão da qualidade de serviço (QoS) da rede do ISP, através da inferência da sequência de estados ocultos em instantes de tempo no futuro.

O artigo [8] também aborda o tópico de detecção de pontos de mudança em séries temporais de medidas de QoS em redes. O objetivo de [8], porém, é propor um *framework* a partir de medições fim-a-fim em redes e *traceroutes* para detectar e localizar automaticamente eventos de rede.

Em [8], são chamados de eventos de rede, por exemplo: um congestionamento transitório em uma ou mais regiões da rede; mudanças de rotas; alguma mudança no funcionamento de um equipamento como uma falha parcial, intermitente ou total em um roteador. Portanto, tal evento de rede se refere a qualquer mudança que possa interferir na QoS percebida pelo usuário final.

O *framework* de análise de dados, proposto em [8], é capaz de detectar automaticamente mudanças estatísticas nas séries temporais de medições de QoS realizadas em diferentes clientes. A detecção da localização do evento se dá considerando a correlação espaço-temporal das mudanças encontradas nas séries e a estimativa da topologia de rede a partir de medições de *traceroutes*.

Diferentemente desta dissertação e de [7], a técnica de detecção de ponto de mudança utilizada em [8] não é feita usando HMM e aplicação do algoritmo de *Viterbi*. Em [8], para detectar pontos de mudança nas séries, é utilizado um processo de otimização definido em [9]. Isso porque os autores de [8] julgaram a abordagem de [9] como eficiente computacionalmente para lidar com um grande número de séries disponíveis e, também, com bom desempenho na eliminação de *outliers* (pontos isolados com pequenas alterações elásticas na série e que não configuram pontos de

mudança de padrão estatístico na série).

O *framework* de [8] foi aplicado em séries temporais reais de medidas de QoS coletadas por mais de 2000 clientes voluntários num espaço de tempo de 10 dias. As métricas de QoS consideradas foram latência, taxa de perda de pacotes e a vazão *upstream*. Os resultados mostram que foi possível identificar e localizar os eventos utilizando apenas as métricas de QoS fim-a-fim e as medições de *traceroute* de diferentes clientes.

Desta forma, podemos perceber que já existem diversos algoritmos para solucionar o problema de detecção de pontos de mudança em séries temporais. Este tema é bastante explorado na literatura. O trabalho [10] é um *survey* contendo diversas abordagens sobre este assunto. O artigo começa, em sua introdução, com uma breve exemplificação de problemas reais que utilizam a detecção de pontos de mudança em séries temporais como suas soluções práticas. Esta lista de exemplos se divide nos seguintes tópicos: monitoramento de condições médicas, detecção de mudanças climáticas, reconhecimento de voz, processamento de imagens e análise de atividades humanas.

Dentro do tópico de análise de atividades humanas utilizando séries, o artigo [10] exemplifica a detecção de transições de padrões em séries temporais de dados de sensores numa rede de uma *smarthome* ou em séries temporais de dados de dispositivos móveis. O objetivo desta dissertação se assemelha a este tópico pelo fato de também se tratar de uma análise e previsão de padrões em séries temporais de uma métrica de QoS em diferentes nós de uma rede. Dentre os diversos métodos citados em [10], encontra-se modelos HMM, a mesma técnica utilizada nesta dissertação para mapear sequências de padrões relativas a séries de perdas.

Quanto à classificação da capacidade da técnica em ser aplicada de forma *online*, os autores de [10] classificam a abordagem da modelagem HMM como próxima do tempo real, considerando apenas um pequeno retardo de alguns instantes de tempo na série. Esta capacidade da modelagem HMM também é citada em [7]. Este pequeno retardo se deve à utilização de pelo menos alguns pontos adjacentes ao instante que se encontra em análise na série.

Quanto à escalabilidade dos métodos de acordo com o aumento da dimensão das séries temporais, os autores de [10] afirmam que, em geral, HMMs são computacionalmente mais custosos do que o algoritmo *Nearest neighbor*, métodos baseados em grafos, métodos Bayesianos, entre outros.

Utilizando diversos tipos de *datasets*, os autores de [10] avaliam os métodos apresentados. A abordagem HMM para detecção de anomalias em séries obteve precisão de 93% em seus resultados.

Devido às vantagens da modelagem HMM para detecção de mudanças em séries temporais, optamos por esta modelagem nesta dissertação.

## 2.2 Aprendizado por reforço aplicado a redes

Além da análise de séries temporais e de modelagem de cadeias de *Markov*, o presente trabalho de dissertação também se encontra fortemente relacionado com trabalhos envolvendo algoritmos de aprendizado por reforço (ou, em inglês, *Reinforcement Learning*, RL).

Algoritmos de RL podem ser utilizados para solucionar diversos problemas. Normalmente, a solução desses problemas envolve um aprendizado de um mapeamento de ações ótimas para cada situação encontrada num sistema a cada instante. Maiores explicações sobre este conceito serão apresentadas no próximo capítulo.

Visto que este trabalho usa aprendizado por reforço para auxiliar na solução de um problema de redes, serão apresentados, nesta seção, trabalhos de RL também aplicados a redes.

No artigo [11], um *workflow* de algoritmos de aprendizado de máquina é apresentado para resolver o problema de *service composition* numa arquitetura orientada a serviços (SOA). Os algoritmos de aprendizado de máquina citados em [11] são redes neurais recorrentes e um algoritmo de RL baseado em Q-Learning.

Segundo [11], uma SOA é uma arquitetura em que *softwares* e sistemas simples são tratados como serviços *web* disponíveis para serem utilizados por outros sistemas mais complexos. *Service composition* é a tecnologia que cria a combinação dos serviços simples existentes num sistema complexo principal.

Logo, o método apresentado por [11] tem o objetivo de aprender e adaptar a combinação de serviços simples que otimizam a QoS de um sistema principal. Essa combinação é realizada através de técnicas de *Machine Learning* e considera o ambiente dinâmico de redes.

O método adaptativo de *service composition*, o algoritmo QP-RL proposto em [11], funciona através da combinação da previsão de QoS e *Reinforcement Learning*. Os valores da tabela Q derivada do método de Q-Learning são atualizados com a previsão da QoS do instante futuro calculado por uma rede neural LSTM.

O trabalho [11] demonstra, através da comparação de três cenários experimentais, que seu algoritmo QP-RL converge mais rápido e de maneira mais estável do que utilizando o algoritmo Q-Learning padrão para o mesmo problema.

O autor principal de [11] possui outro trabalho anterior [12], que possui o mesmo objetivo de [11] porém utilizando somente *Reinforcement Learning*, diferentemente de [11] que utiliza uma mistura de redes neurais recorrentes e RL.

Em [12], o algoritmo proposto é mais simples do que em [11]. Primeiramente, o trabalho [12] propõe uma modelagem usando *Markov Decision Process* (MDP) que contém os seguintes parâmetros: os estados representam conjuntos de possíveis serviços *web* disponíveis a cada instante, as ações representam as escolhas entre os

serviços *web* disponíveis em cada estado, a matriz de probabilidade de transição de estados e, por fim, as recompensas são dadas a cada transição de estado segundo algumas métricas de QoS no momento dessa transição.

Esta dissertação também usa um MDP para modelar estados da rede baseados em métricas de QoS assim como em [12] porém isto é feito de maneira diferente. Nesta dissertação, os valores da métrica de QoS analisada são mapeados em estados ocultos de um HMM e, depois, sequências de estados ocultos desse HMM são variáveis de estado de um MDP.

Como nesta dissertação, os autores de [12] aplicam uma técnica de RL na modelagem MDP. A técnica de RL usada é o *Q-Learning* e o resultado é o mapeamento do serviço *web* ótimo de acordo com cada estado.

Em [13], é apresentado o algoritmo chamado LearnQoS. Assim como nesta dissertação, o trabalho [13] utiliza uma técnica de RL para tomar ações no sistema de um ISP a fim de maximizar a qualidade de serviço (QoS) da rede oferecida para os seus clientes. No artigo, a técnica de RL que foi utilizada se baseia no algoritmo de *Q-Learning* e as ações ótimas obtidas pelo modelo são implementadas numa rede definida por software (SDN); enquanto, nesta dissertação, a técnica de RL usada é o *Value Iteration* e as ações ótimas são implementadas no sistema de alarme de manutenção do ISP.

Segundo [14], redes definidas por software (SDNs) são redes que possuem um programa de software central que dita as ações e ligações desta rede para realizar o transporte de dados. Logo, o algoritmo LearnQoS aprende, após seu devido treinamento, o mapeamento da ação que a SDN do ISP deve tomar a cada instante de transmissão, de acordo com a circunstância momentânea da rede. As ações possíveis são as seguintes: não fazer nada e manter a transmissão do mesmo jeito como ela se encontra; reduzir a taxa de transmissão; aumentar a taxa de transmissão; e redirecionar o roteamento da transmissão para um caminho alternativo com o melhor fluxo. O LearnQoS [13] apresentou, em seus experimentos, uma menor taxa de perda de pacotes e menor latência na transmissão de vídeo em comparação com o mesmo tipo de transmissão utilizando uma rede SDN padrão (sem o algoritmo LearnQoS).

Portanto, tal como nos trabalhos citados nesta seção, uma técnica de *Reinforcement Learning* foi escolhida para a dissertação a fim de obter um mapeamento de ações que otimizam a QoS da rede.

## 2.3 Relacionando com a dissertação

Os trabalhos enumerados neste capítulo utilizam técnicas semelhantes às técnicas usadas na metodologia desta dissertação. O primeiro trabalho relacionado enumerado [7] foi aquele no qual esta dissertação foi baseada. Nele, assim como na segunda

etapa da metodologia desta dissertação, os autores utilizam um HMM para inferir mudanças de padrões em séries de perdas de pacotes. Em [8], é apresentada outra técnica para detecção de pontos de mudança em séries de medidas de desempenho de redes. E, em [10], temos um *survey* contendo o mesmo tema de detecção de pontos de mudança em séries aplicado a redes, porém utilizando diferentes técnicas de *Machine Learning*.

Na terceira etapa da metodologia desta dissertação usamos a técnica de *Reinforcement Learning* (RL). Os trabalhos [11] e [13] utilizam técnicas de RL para realizar ações de otimização de QoS em redes. Enquanto que, em [11] e [13], o treinamento da técnica de RL encontra ações ótimas diretamente relacionadas às transmissões de dados na rede; nesta dissertação essa etapa de RL otimiza o acionamento de um alarme que informa problemas na rede.

# Capítulo 3

## Conceitos Básicos

O objetivo deste capítulo é apresentar os conceitos que serão utilizados na metodologia e nos experimentos que se encontram nos capítulos seguintes.

Serão descritos o modelo de cadeia de *Markov* oculta (HMM) e o algoritmo de *Viterbi*. Serão apresentados também os seguintes algoritmos de *Reinforcement Learning* (RL): *Value Iteration* e *Policy Iteration*. No *Value Iteration*, temos a aplicação iterativa das equações de Bellman para descobrir os valores que convergem e montar a política ótima do sistema; enquanto, o *Policy Iteration* realiza iterações com uma etapa de avaliação da política momentânea e outra etapa de melhoria da política, até a convergência ser atingida. Este capítulo também será importante para exemplificar e apresentar a notação utilizada nesta dissertação.

### 3.1 Cadeia de *Markov* oculta (HMM)

Uma cadeia de *Markov* oculta (HMM) [15] apresenta características similares a uma cadeia de *Markov* [16]. Um HMM é um processo estocástico com um espaço de estados  $\Omega_1 = \{X_1, X_2, \dots, X_{n_1}\}$ , um vetor  $\lambda$  de probabilidade inicial para os estados, uma matriz de probabilidade de transição de estados  $T_1$  com dimensão  $n_1 \times n_1$ , e possui a propriedade de *Markov*  $P(X^{(t)}|X^{(t-1)}, \dots, X^{(1)}, X^{(0)}) = P(X^{(t)}|X^{(t-1)})$ .

A diferença do HMM com relação a uma cadeia de *Markov* é que, nele, os estados não são observáveis. Cada estado possui uma probabilidade de emitir um símbolo observável do conjunto  $\Omega_2 = \{O_1, O_2, \dots, O_{n_2}\}$ . Uma matriz de probabilidade de emissão  $T_2$  com dimensões  $n_1 \times n_2$  mapeia cada estado com a probabilidade de emitir cada símbolo observável. Dessa forma, temos os seguintes parâmetros que compõem um HMM:

- espaço de estados ocultos  $\Omega_1 = \{X_1, X_2, \dots, X_{n_1}\}$ ;
- espaço de símbolos observáveis  $\Omega_2 = \{O_1, O_2, \dots, O_{n_2}\}$ ;

- vetor  $\lambda$  de probabilidade inicial para os estados ocultos;
- matriz de transição de estados  $T_1$  com dimensão  $n_1 \times n_1$ ;
- matriz de probabilidade de emissão de observações  $T_2$  com dimensão  $n_1 \times n_2$ .

Em uma cadeia de *Markov* oculta, é conhecida somente a sequência de símbolos observáveis ao longo do tempo. Uma aplicação clássica de um modelo HMM é estimar qual a sequência de estados ocultos mais provável de ter gerado essa sequência de símbolos observáveis.

Seja uma sequência de símbolos observáveis  $\Theta = (O^{(0)}, O^{(1)}, O^{(2)}, \dots, O^{(\tau)})$ , logo, teremos uma sequência de estados desconhecidos  $\Phi = (X^{(0)}, X^{(1)}, X^{(2)}, \dots, X^{(\tau)})$ ,  $t = 1, 2, \dots, \tau$ . Cada elemento de  $\Phi$  é um estado oculto pertencente ao espaço de estados  $\Omega_1$  e cada símbolo da sequência visível  $\Theta$  é um elemento do conjunto de observações  $\Omega_2$ . A Figura 3.1 ilustra uma sequência de observações e os respectivos estados ocultos.

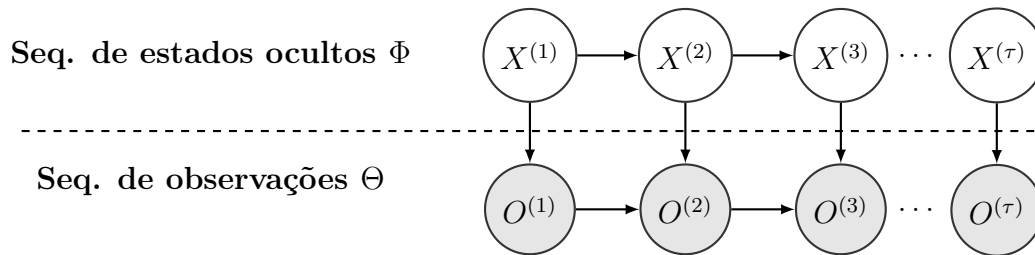


Figura 3.1: Exemplo de estados ocultos da HMM e respectivas observações

Um exemplo clássico e simples de modelagem usando cadeia de *Markov* oculta é descrito a seguir. Uma pessoa tem uma moeda  $X_1$  e uma moeda  $X_2$ . Ao serem lançadas numa superfície qualquer, ambas podem cair com o lado cara voltado para cima ou com o lado coroa. Um lado da moeda é representado pelo símbolo observável  $O_1$  e o outro, pelo símbolo  $O_2$ . As moedas  $X_1$  e  $X_2$  são viciadas e, portanto, ao serem lançadas,  $X_1$  possui uma probabilidade  $p_a$  de emitir observação do lado  $O_1$ , enquanto a moeda  $X_2$  possui uma probabilidade  $p_b$  de emitir observação do lado  $O_2$ .

Esta pessoa decide lançar uma moeda por vez (sem saber qual é a moeda) e anotar, em sequência, apenas as observações de qual lado da moeda estava voltado para cima. Considere que se a moeda  $X_1$  for lançada num instante qualquer, então a moeda  $X_2$  será lançada no próximo instante com probabilidade  $p_c$ . De modo semelhante, se a moeda  $X_2$  for lançada em outro instante qualquer, então a moeda  $X_1$  será lançada no próximo instante com probabilidade  $p_d$ . O vetor de probabilidade inicial de estados ocultos é  $\lambda_a$ . Os parâmetros do modelo HMM estão apresentados abaixo.

- espaço de estados ocultos  $\Omega_1 = \{X_1, X_2\}$ ;
- espaço de símbolos observáveis  $\Omega_2 = \{O_1, O_2\}$ ;
- vetor de probabilidade inicial de estados ocultos é  $\lambda_a$ ;
- matriz de probabilidade de transição de estados  $T_1 = \begin{pmatrix} 1 - p_c & p_c \\ p_d & 1 - p_d \end{pmatrix}$ ;
- matriz de probabilidade de emissão de observações  $T_2 = \begin{pmatrix} p_a & 1 - p_a \\ 1 - p_b & p_b \end{pmatrix}$ .

Veja o esquema gráfico deste modelo HMM.

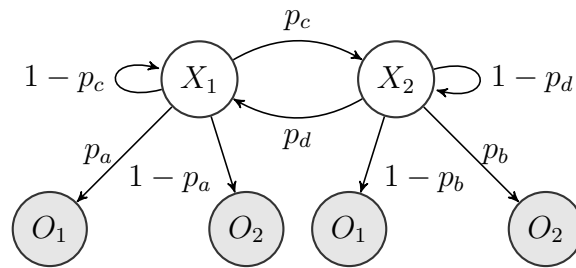


Figura 3.2: Exemplo de cadeia de *Markov* oculta

A sequência desconhecida de qual moeda foi lançada por vez, representada por  $\Phi = (X^{(0)}, X^{(1)}, X^{(2)}, \dots, X^{(\tau)})$  tal que  $X^{(t)} \in \{X_1, X_2\}$ , pode ser estimada a partir da sequência de lados da moeda observados em cada lançamento,  $\Theta = (O^{(0)}, O^{(1)}, O^{(2)}, \dots, O^{(\tau)})$ , tal que  $O^{(t)} \in \{O_1, O_2\}$ , para todo  $t = 1, 2, \dots, \tau$ .

O algoritmo de *Viterbi* é utilizado para inferir a sequência de estados ocultos  $\Phi$  mais provável, de acordo com o HMM e a sequência de observações  $\Theta$ . Se o modelo HMM não está completo, alguns de seus parâmetros podem ser estimados através da aplicação do algoritmo *Baum-Welch*.

O algoritmo *Baum-Welch* é um caso particular do algoritmo *Expectation-Maximization* (EM) aplicado a um HMM [17].

O algoritmo de *Viterbi* foi desenvolvido em 1967 [15, 18]. O principal objetivo desse algoritmo é descobrir a sequência de estados ocultos mais provável de ter gerado uma sequência de observações, dado um modelo de cadeia de *Markov* oculta.

Voltemos ao exemplo do início desta seção, o de duas moedas viciadas. Considere uma quantidade  $t$  de lançamentos das moedas onde a troca das moedas é realizada segundo as probabilidades de transição entre os estados ocultos do HMM. A sequência  $\Phi = (X^{(1)}, X^{(2)}, X^{(3)}, \dots, X^{(\tau)})$  de estados ocultos é desconhecida pois não sabemos a moeda usada no lançamento. O que conhecemos é o resultado do experimento, cara ou coroa, ou seja, as observações.



Um estado oculto qualquer  $X^{(t)}$  (tal que  $t = 1, 2, \dots, \tau$ ) na sequência  $\Phi$  pode representar uma das moedas viciadas,  $X_1$  ou  $X_2$ ; já uma observação qualquer  $O^{(t)}$ ,  $t = 1, 2, \dots, \tau$ , é conhecida e representa o resultado do experimento cara  $O_1$  ou coroa  $O_2$ .

Além da sequência de observações, o algoritmo de *Viterbi* necessita de todas as probabilidades relacionadas à cadeia de *Markov* oculta para que, assim, seja possível inferir a sequência de estados ocultos que melhor explica a sequência observada. Veja, a seguir, a entrada e saída do algoritmo.

- Entradas do algoritmo de *Viterbi*:
  - O conjunto de estados  $\Omega_1 = \{X_1, X_2, \dots, X_{n_1}\}$ ;
  - O conjunto de observações  $\Omega_2 = \{O_1, O_2, \dots, O_{n_2}\}$ ;
  - O vetor  $\lambda$  de probabilidade inicial de cada estado oculto.
  - A matriz  $T_1$  de probabilidade de transição entre estados ocultos;
  - A matriz  $T_2$  de probabilidades de emissão de cada símbolo para cada estado;
  - A sequência de observações  $\Theta = (O^{(1)}, O^{(2)}, \dots, O^{(t)})$ .
- Saída do algoritmo de *Viterbi*:
  - A sequência de estados ocultos  $\Phi = (X^{(1)}, X^{(2)}, \dots, X^{(\tau)})$  mais provável.

Voltando ao exemplo das moedas, a partir da sequência de observações, usando o algoritmo de *Viterbi*, é possível inferir a sequência de moedas mais provável usada nos experimentos.

## 3.2 Aprendizado por Reforço

Aprendizado por Reforço (*Reinforcement Learning*) é uma área de aprendizado por máquina onde agentes devem realizar ações em um ambiente com o objetivo de maximizar a recompensa acumulada. Como exemplo de problemas solucionáveis com algoritmos de aprendizado por reforço, temos: o aprendizado de quais movimentos se deve realizar numa partida de xadrez e outros jogos de tabuleiro [19–21]; o aprendizado de um robô que deve encontrar um objeto numa sala em menor tempo possível [22–24]; o aprendizado do controle de estoque de um determinado produto de uma loja [25, 26]; entre outros.

Essa técnica é adequada para o problema desta dissertação porque ela permite um aprendizado em ambientes com incertezas de dados a cada momento. Em problemas de RL, os estados evoluem no tempo e o sistema deve aprender entre obter a melhor

recompensa instantânea ou esperar para realizar outra ação futura que maximiza a recompensa acumulada a longo prazo, ou seja, um *trade-off* entre recompensa instantânea e recompensa acumulada. Essa característica pode ser encontrada no problema desta dissertação através do aprendizado de quando é o melhor momento de acionar o alarme na série, dado um estado no instante atual.

Um Processo de Decisão de Markov (Markov Decision Process (MDP)) é um formalismo matemático usado para representar a dinâmica de um sistema de Aprendizado por Reforço. Segundo [27, 28], um MDP é um modelo matemático que permite a representação dos agentes que aprendem através de interações com o ambiente visando atingir um objetivo.

Um Processo de Decisão de Markov é uma n-upla  $\langle \mathcal{S}, \mathcal{A}, \mathbf{P}, \mathcal{R} \rangle$ , onde:

- $\mathcal{S}$  é um conjunto finito de estados;
- $\mathcal{A}$  é um conjunto finito de ações;
- $\mathbf{P}$  é a matriz de transição entre estados da cadeia de Markov do processo tal que:  $\mathbf{p}_{s,s'}^a = P[S_{t+1} = s' | S_t = s, A_t = a]$ ;
- $\mathcal{R}$  é a função de recompensa,  $R_s^a = E[R_{t+1} | S_t = s, A_t = a]$ ;

Além disso, seja  $\pi(a|s) = P[A_t = a | S_t = s]$  ( $\pi$  então é uma distribuição sobre as ações).

Abaixo, na Figura 3.3, é apresentado um exemplo de um modelo MDP com as probabilidades de transição entre estados para cada ação (em cores diferentes por ação) e os valores de recompensa instantânea recebida para cada transição entre estados e ação. O sistema contém apenas dois estados e duas ações.

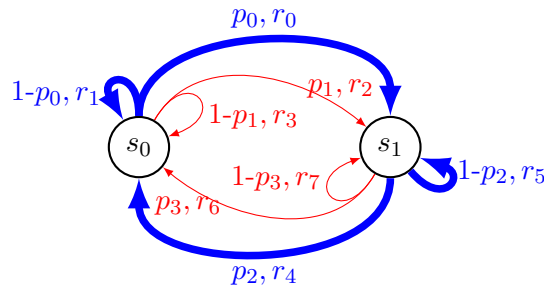


Figura 3.3: Exemplo de MDP: arestas azuis correspondem a  $a=0$  e vermelhas,  $a=1$ .

Conforme pode ser visto na Figura 3.3, as ações de um MDP definem as probabilidades de transição entre os estados, logo, elas também definem a recompensa instantânea do sistema e a recompensa acumulada devido à troca de estados que a ação instantânea gerou.

A Figura 3.4 abaixo [28] representa a interação do agente com o ambiente usando a notação de um MDP. O agente tem as funções de aprender e tomar as decisões. Todos os elementos com os quais o agente interage fazem parte do ambiente. O agente e o ambiente interagem continuamente, o agente seleciona as ações que deve tomar e o ambiente responde a essas ações apresentando novas situações para o agente. O ambiente também fornece recompensas para o agente, que tenta maximizá-las através da escolha de suas ações ao longo do tempo. O agente e o ambiente interagem em instantes discretos de tempo,  $t = 0, 1, 2, \dots$ . Em cada instante  $t$ , o agente recebe informações do estado  $S_t$  e baseado nessas informações seleciona uma ação  $A_t$ . No instante de tempo  $(t + 1)$  o agente recebe uma recompensa,  $R_{t+1}$ , e um novo estado  $S_{t+1}$  é alcançado.

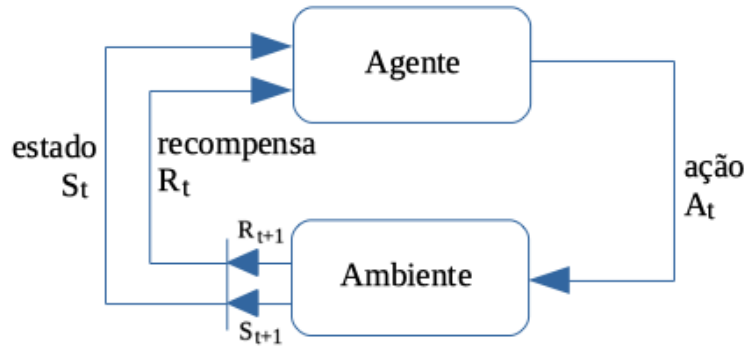


Figura 3.4: MDP representando a interação entre agente e ambiente.

Conforme dito anteriormente, o objetivo do agente é maximizar a recompensa acumulada ao longo do tempo. No caso mais simples, a função a ser maximizada seria a soma das recompensas instantâneas recebidas em cada instante de tempo que pode ser definida como  $G_t = R_{t+1} + R_{t+2} + \dots + R_\tau$ , onde  $\tau$  é o instante final. No entanto, para sistemas onde  $\tau \rightarrow \infty$ , a função  $G_t$  poderia também tender a infinito. Por isso, é necessária a introdução de um fator de desconto  $\gamma$  nas parcelas de  $G_t$ , onde  $0 \leq \gamma \leq 1$ . A ideia do fator de desconto é que o agente selecione ações de modo que a soma das recompensas com desconto que ele receberá no futuro seja maximizada. Em particular, o agente escolhe  $A_t$  para maximizar a nova função  $G_t$ , agora com o fator de desconto, definida abaixo.

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} \dots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \quad (3.1)$$

A maioria dos algoritmos de aprendizado por reforço envolve a estimativa de funções de valor, ou seja, funções de estados (ou de pares estado-ação) que estimam o quão bom é para o agente estar em um determinado estado (ou o quão bom é

executar uma determinada ação em um determinado estado). A noção de “quão bom” é definida em termos das recompensas que o agente pode ganhar no futuro. As funções de valor são definidas de acordo com uma política. Uma política é uma função que modela o comportamento do agente, ou seja, pode ser vista como um conjunto de regras que indicam para um determinado estado que ação deve ser tomada.

Chamaremos de  $v_\pi(s)$  a função de valor de um estado  $s$  sob uma política  $\pi$ . Essa função representa o valor médio do retorno partindo do estado  $s$  e seguindo a política  $\pi$ . Uma propriedade fundamental de funções de valor é que elas podem ser computadas usando recursividade. A função  $v_\pi(s)$  pode ser computada conforme descrito a seguir. Note que a Equação 3.2 é a equação de *Bellman* para  $v_\pi$ .

$$\begin{aligned}
v_\pi(s) &= E_\pi[G_t | S_t = s] \\
&= E_\pi[R_{t+1} + \gamma G_{t+1} | S_t = s] \\
&= \sum_{a \in \mathcal{A}} \pi(a|s) \left[ \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} p_{s,s'}^a v_\pi(s') \right] \tag{3.2}
\end{aligned}$$

Similarmente, podemos definir a função  $q_\pi(s, a)$  que representa o valor médio do retorno a partir do estado  $s$ , escolhendo a ação  $a$ , e depois seguindo a política  $\pi$ . A Equação 3.3 é a equação de *Bellman* para  $q_\pi$ .

$$\begin{aligned}
q_\pi(s, a) &= E_\pi[G_t | S_t = s, A_t = a] \\
&= E_\pi[R_{t+1} + \gamma G_{t+1} | S_t = s, A_t = a] \\
&= \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} p_{s,s'}^a \sum_{a' \in \mathcal{A}} \pi(a'|s') q_\pi(s', a') \tag{3.3}
\end{aligned}$$

Uma das tarefas de aprendizado por reforço é aprender uma política de ações  $\pi^*$  ótima, que maximiza a função  $v_\pi(s)$  ou a função  $q_\pi(s, a)$ . Em outras palavras, de que maneira o agente deve agir para maximizar as suas recompensas futuras.

Alguns algoritmos iterativos podem ser usados para encontrar a política ótima. Dois métodos fundamentais para encontrar a política ótima são: *Policy Iteration* e *Value Iteration* [29], usaremos este último neste trabalho.

O algoritmo *Policy Iteration* é inicializado com uma política aleatória  $\pi$ . Logo após, são realizadas alternadamente sucessivas etapas de *Policy Evaluation* ( $\xrightarrow{E}$ ) e *Policy Improvement* ( $\xrightarrow{I}$ ), atualizando a política e as funções  $v_\pi(s)$  até o método convergir para a política ótima  $\pi^*$ . Abaixo é ilustrada a sequência das etapas.

$$\pi \xrightarrow{E} v_\pi \xrightarrow{I} \pi' \xrightarrow{E} v_{\pi'} \xrightarrow{I} \pi'' \xrightarrow{E} \dots \xrightarrow{I} \pi^* \xrightarrow{E} v_{\pi^*}$$

Na etapa de *Policy Evaluation*, o valor de  $v_\pi$  é atualizado de acordo com a aplicação da política da iteração anterior na Equação 3.4.

$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \left[ \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} p_{s,s'}^a v_\pi(s') \right] \quad (3.4)$$

Na etapa de *Policy Improvement*, uma nova política  $\pi'$  é gerada conforme a Equação 3.5.

$$\pi'(s) = \operatorname{argmax}_{a \in \mathcal{A}} \left[ \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} p_{s,s'}^a v_\pi(s') \right] \quad (3.5)$$

Após algumas iterações, a Equação 3.5 passa a manter a mesma política. Nesse caso, dizemos que o método convergiu e a política que se mantém é a política ótima  $\pi^*$ .

Embora o algoritmo *Policy Iteration* calcule a política ótima para um determinado MDP em um tempo finito, ele é relativamente ineficiente. Em particular, a primeira etapa, a etapa de avaliação da política, é computacionalmente cara. Funções de valor para todas as políticas intermediárias são calculadas, o que envolve múltiplas varreduras através do espaço de estados completo por iteração.

De acordo com [28], a etapa de *Policy Evaluation* do algoritmo *Policy Iteration* pode ser truncada de várias maneiras sem perder a garantia de convergência do método. Um caso especial importante é quando a etapa de *Policy Evaluation* é interrompida após apenas uma varredura (uma atualização de cada estado). Esse algoritmo é chamado de *Value Iteration* e consiste de uma operação de atualização da função de valor onde são combinadas as etapas de *Policy Improvement* e *Policy Evaluation* truncada.

O algoritmo inicia com valores aleatórios para a função de valor  $v_\pi(s)$ . Para cada estado a função  $v_\pi(s)$  é atualizada conforme a Equação 3.6. Ou seja, tomamos a ação que maximiza o retorno esperado  $\mathcal{R}_s^a$  e o valor esperado dos retornos dos possíveis estados vizinhos  $v_\pi(s')$  que podem ser alcançados tomando a ação  $a$  (considerando o fator de desconto  $\gamma$ ), maximizando o valor de  $v_\pi(s)$  para a iteração atual.

$$v_\pi(s) = \max_{a \in \mathcal{A}} \left[ \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} p_{s,s'}^a v_\pi(s') \right] \quad (3.6)$$

Essas iterações são realizadas até a convergência, isto é, quando a diferença entre a função de valor de duas iterações consecutivas é menor que um valor  $\delta$ . A política ótima pode ser obtida de acordo com a seguinte expressão:

$$\pi(s) = \operatorname{argmax}_{a \in \mathcal{A}} \left[ \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} p_{s,s'}^a v_\pi(s') \right] \quad (3.7)$$

# Capítulo 4

## Metodologia Geral

A principal aplicação do método proposto nessa dissertação é dar suporte a um provedor de serviço de internet (ISP) para sinalizar a necessidade de procedimentos de reparos ou de manutenção preventiva em sua rede. Este sistema de decisão se baseia, apenas, na informação das séries de medidas de desempenho dessa rede que, neste caso, se tratam de séries temporais de taxa de perda de pacotes.

As etapas que compõe a metodologia proposta estão apresentadas na Figura 4.1.

**1ª etapa – Coleta dos dados:**

- Medição de taxa de perda de pacotes dos clientes do ISP;
- Agrupamento das medidas em séries temporais diárias de taxa de perda de pacotes.



**2ª etapa – HMM e algoritmo de *Viterbi*:**

- Modelo HMM para as séries de temporais de taxa de perda de pacotes;
  - Estados ocultos e observações representam condições da rede e taxa de perda de pacotes.
- Séries de condições de rede (algoritmo de *Viterbi*).



**3ª etapa: MDP e algoritmo *Value Iteration*:**

- Modelo MDP para as séries de condições de rede e um alarme de manutenção:
  - Ação num instante  $t$ : ligar/desligar o alarme ou mantê-lo ligado/desligado;
  - Estado em  $t$ : informações do alarme em  $t-1$  e algumas condições da rede até  $t$ .
- Otimização do alarme (algoritmo *Value Iteration*)



**Output:** Mapeamento da ação ótima de um alarme de manutenção para cada conjunto de últimas condições de rede, de acordo com o orçamento do ISP para reparos.

Figura 4.1: Etapas da Metodologia

A primeira etapa da metodologia, bem importante para a confiabilidade dos resultados, é a coleta de dados. Essa coleta é feita nos próprios roteadores residenciais

dos clientes do ISP. A cada instante de tempo, uma medida é coletada de um usuário até completar uma série diária. Deste modo, o *dataset* é formado por séries temporais resultantes da análise de vários dias e de vários usuários.

A partir das séries temporais que compõe o *dataset*, é definido e parametrizado um HMM. São definidos os estados ocultos, cada um com uma distribuição de probabilidade de emissão de observações. As observações correspondem aos valores de taxa de perda de pacotes encontrados nas séries temporais. Por isso, os estados ocultos modelados são associados a condições de rede. Por exemplo, uma boa condição de rede possui maior probabilidade de emitir baixas taxas de perda de pacotes, enquanto uma condição ruim de rede tem maior probabilidade de emitir altas taxas de perda.

Ainda nesta etapa, de acordo com a modelagem HMM, são obtidas séries temporais de condições de rede correspondentes às séries temporais de taxa de perda de pacotes através do algoritmo de *Viterbi*. Essas séries temporais de condições de rede compõe o novo *dataset* que será usado na próxima etapa da metodologia. Esse *dataset* contém informações sobre a qualidade de serviço (QoS) da rede do ISP e pode ser usado com diferentes objetivos. O trabalho [7], por exemplo, usa esse *dataset* para identificar regiões na rede do ISP que apresentam uma condição ruim de rede ou falhas.

O objetivo da metodologia desta dissertação visa a antecipação de ações corretivas e pró-atividade do ISP no combate de falhas na sua rede. Neste caso, porém, segue-se uma abordagem diferente de [7], através de *Reinforcement Learning* (RL). Então, após as etapas de coleta de dados e de obtenção de séries temporais de condições de rede, há uma etapa que utiliza uma técnica de aprendizado por reforço para decidir sobre a ação de alarmar um sistema de manutenção do ISP. A técnica de RL decide sobre o alarme a cada instante de tempo de uma série temporal de condições de rede em análise. Essa decisão a cada instante de uma série se baseia nos instantes de tempo adjacentes anteriores presentes na mesma série.

Ainda sobre a última etapa da metodologia, a decisão sobre disparar ou não um alarme para o ISP é restrita a um número máximo de alarmes (orçamento). Além da informação de condições de rede a cada instante e nos instantes adjacentes anteriores, também será considerada no treinamento da técnica de RL, a informação sobre o número de alarmes que já foram acionadas naquela mesma série.

Por exemplo, seja um *dataset* de séries de condições de rede em que, de modo recorrente, as condições ruins sejam momentâneas; ou seja, nas séries temporais desse *dataset*, há geralmente sequências curtas de condições ruins de rede dentre os muitos instantes de condições boas. Neste caso, após o treinamento do algoritmo de RL, o sistema aprende que, num dado momento da série, ao encontrar uma mudança de condição boa de rede para uma condição ruim, ele deve analisar também alguns



instantes seguintes antes da decisão de acionar o alarme. Portanto, um alarme só é acionado se a mudança persiste durante um certo intervalo de tempo.

O uso da técnica de RL, ilustrada pelo exemplo acima, permite a tomada de decisão baseada na observação das condições da rede. As tomadas de decisão poderiam ser mais simples, sem o algoritmo de RL, bastaria apenas um simples mapeamento do alarme para instantes da série com condições ruins de rede. Porém, com a inclusão do algoritmo de RL, pode-se diminuir o número de alarmes acionados sem perda de qualidade para os clientes do ISP.

O problema de otimização, presente na última etapa da metodologia, será modelado através de um *Markov Decision Process* (MDP). A ação ótima do alarme será aprendida pela aplicação de um algoritmo de RL, o *Value Iteration* (VI).

## 4.1 1<sup>a</sup> etapa do método: Coleta dos dados

De acordo com [30], medir o desempenho de uma rede de um ISP a partir dos roteadores domésticos de seus clientes se mostra mais eficaz em estimar a qualidade de serviço (QoS) experimentada por seus usuários. Medições realizadas diretamente pelo gerenciador da rede ou, até mesmo, nas mediações da rede do ISP podem desconsiderar falhas que só ocorrem próximas aos roteadores domésticos finais dos usuários.

A Anlix, empresa incubada na COPPE/UFRJ, parceira do laboratório LAND, desenvolveu um software para coleta de diversas métricas de desempenho de rede. Este software foi instalado nos roteadores domésticos do provedor Gigalink, parceiro do laboratório LAND.

Diversas medidas são coletadas pelo software de medição, como tráfego de *download* e *upload*, latência, taxa de perda de pacotes, entre outros. As medidas coletadas são transmitidas de cada roteador doméstico para um servidor no laboratório LAND/COPPE que recebe todas medições de todos os usuários. O esquema da Figura 4.2 abaixo ilustra o ambiente de medição.

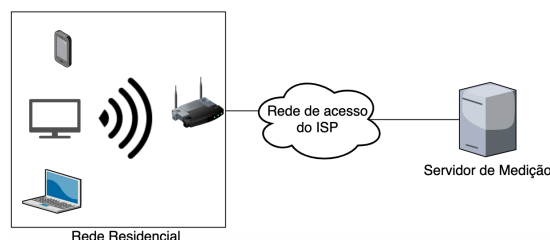


Figura 4.2: Infraestrutura de medição

O servidor de coleta das medições armazena uma série temporal diária para cada métrica e cada usuário. Logo, cada série temporal do *dataset* possui 1440 amostras,

uma para cada minuto do dia, e cada uma delas pode ser identificada com o vetor de informações: usuário da rede, data de medição, identificação da métrica.

Segundo [31] e [32], a variação da taxa de perda de pacotes tem maior influência na QoE do que a latência. Por isso, neste trabalho de dissertação somente é utilizada, como medida de desempenho de rede, a taxa de perda de pacotes. A metodologia, porém, poderia ser aplicada para outras medidas de desempenho de rede em conjunto com a taxa de perda.

Para obter uma medida de taxa de perda, uma rajada de 100 pacotes ICMP é enviada dos roteadores domésticos dos clientes para um servidor localizado na rede do ISP. As rotas utilizadas durante as medições incluem apenas equipamentos que se encontram dentro da rede do ISP. A cada minuto, para cada roteador, é feita uma simples contagem de quantos pacotes foram enviados e quantos foram recebidos. E, assim, as séries temporais diárias de perda de pacotes são construídas no servidor de coleta.

Se, em um minuto qualquer, o tráfego gerado pelo próprio cliente for muito alto; a medição dessa amostra na série pode ser afetada mesmo que sua rede esteja em boas condições. Isto acontece porque as possíveis perdas que forem medidas durante um alto consumo de tráfego no roteador podem ocorrer na própria fila do roteador da rede naquele momento.

Sendo assim, é importante filtrar os resultados coletados nos minutos nos quais o tráfego do cliente é maior do que um determinado valor limiar escolhido proporcionalmente à vazão nominal dos clientes da rede. Neste trabalho, este limiar escolhido foi um tráfego de 2,5 Mbps pois este valor se mostrou suficientemente baixo para garantir acurácia na medida sendo coletada.

Os minutos filtrados, por esse motivo apresentado ou por outras perdas de amostragem quaisquer, são sinalizados e, posteriormente, participarão da modelagem HMM realizada na próxima etapa. A fim de garantir um número mínimo de amostras com medições de taxa de perda de pacotes, são descartadas do *dataset* as séries temporais com menos de 1000 medições, lembrando que cada série têm o potencial de 1440 amostras com medições.

Portanto, seguindo essas regras de coleta de dados, o *dataset* total disponível conta com 68849 séries temporais diárias. A medição foi realizada ao longo de 31 dias corridos e possui um número variável de roteadores domésticos participantes por dia.

Apesar da metodologia ter sido testada utilizando todos os dados coletados, no capítulo 5, ela foi aplicada em dois subconjuntos distintos contidos neste *dataset* de 68849 séries. Além disso, cada subconjunto ainda é dividido entre séries de treinamento e séries de teste. Desta forma, o desempenho da metodologia é comparado entre resultados obtidos em subconjuntos distintos. No capítulo seguinte, estão

descritos mais detalhes dos dados utilizados em cada experimento.

Essa foi uma coleta de dados largamente baseada na coleta do trabalho de [7]. O *dataset* desta dissertação não contém as mesmas séries temporais presentes no *dataset* de [7] pois cada trabalho utiliza uma sequência diferente de dias de coleta. As séries do *dataset* desta dissertação possuem dados um pouco mais recentes do que os dados de [7], porém, com as mesmas características de medição.

## 4.2 2ª etapa do método: HMM e algoritmo de *Viterbi*

A partir da coleta de dados de modo como foi descrito no item anterior, podem ser encontradas séries com grandes oscilações em amostras de medições adjacentes. Instabilidades na rede geram taxas de perda de pacotes (ou quaisquer outras medidas de desempenho de rede) que podem variar muito de minuto a minuto. A otimização das ações do alarme da terceira etapa do método necessita de uma filtragem de *outliers* que ocorre nesta segunda etapa.

Exemplificando, se houver apenas uma amostra da série que corresponda a uma taxa alta de perda de pacotes em seis horas de medição, espera-se que o sistema considere que a rede está em condições boas ao longo dessas seis horas. Visto que esse único instante com taxa alta de perda de pacotes foi um evento raro, a condição de rede não deve ser considerada ruim durante essas 6 horas. Mas se, após estas seis horas, o sistema começar a apresentar taxas altas de perda de pacotes em amostras consecutivas, o sistema pode considerar que a condição da rede mudou para uma condição ruim, uma vez que o padrão das amostras coletadas se alterou.

Para solucionar o problema descrito nos parágrafos anteriores, iremos utilizar um modelo de *Markov* oculto (HMM). Os estados ocultos do HMM serão associados às possíveis condições da rede e os símbolos emitidos pelo HMM serão associados às medições de taxas de perda de pacotes, da mesma forma como foi feito em [7].

O HMM é definido por um conjunto  $\Omega_1$  de 4 estados ocultos,  $\Omega_1 = \{x_1, x_2, x_3, x_4\}$ , e um conjunto de símbolos observáveis  $\Omega_2 = \{o_0, \dots, o_{19}, o_{20}, o_{21}\}$ . Os elementos do conjunto  $\Omega_1$ , os estados ocultos  $x_1, x_2, x_3$  e  $x_4$ , representam respectivamente uma condição boa da rede, uma condição mediana, uma ruim e uma condição de indisponibilidade da rede. Enquanto, os elementos do conjunto  $\Omega_2$  de  $o_0$  a  $o_{20}$  representam as taxas de perda de pacotes por minuto. O símbolo  $o_{21}$  representa *missing values*, que geralmente ocorrem quando os roteadores de medição estão com altas perdas de pacotes e falham em enviar dados para o servidor de coleta de medição. A Figura 4.3 ilustra a estrutura do modelo HMM, utilizado em [7] e nesta dissertação, para representar as condições da rede ao longo do tempo.

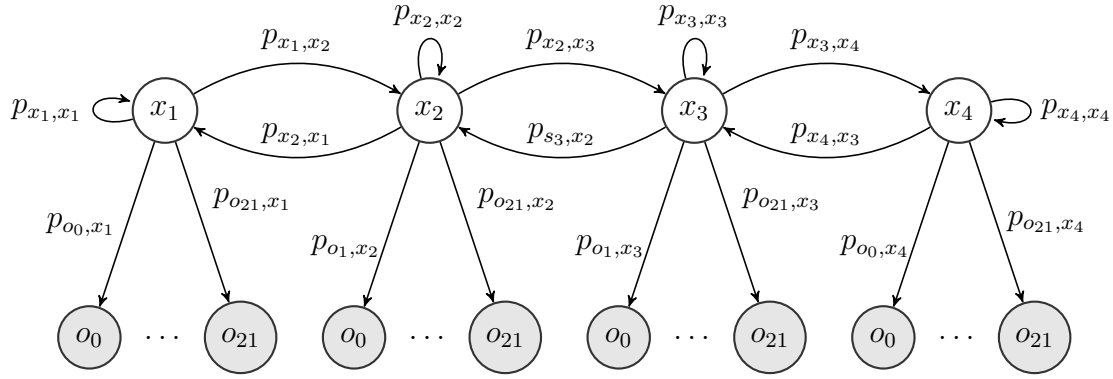


Figura 4.3: Esquema do modelo de *Markov* oculto utilizado

A parametrização do modelo HMM é realizada através da aplicação do algoritmo *Baum-Welch* no modelo da Figura 4.3 e as séries temporais de taxa de perda de pacotes. Segundo [17], o algoritmo *Baum-Welch* é um caso particular do algoritmo *Expectation-Maximization* (EM).

No algoritmo *Baum-Welch*, o método EM é realizado simultaneamente para estimar as probabilidades de trocas de estado ocultos e as probabilidades de emissões das observações num instante  $t$  qualquer da série. De acordo com o esquema da Figura 4.3, as probabilidades de transição de estados ocultos são os valores de  $p_{x_t, x_{t+1}} = \{p_{x_1, x_1}, p_{x_1, x_2}, p_{x_2, x_1}, p_{x_2, x_2}, p_{x_2, x_3}, \dots\}$ , enquanto as probabilidades de emissões de cada estado oculto no instante  $t$  são representados pelos valores  $p_{o_t, x_t} = \{p_{o_0, x_1}, \dots, p_{o_{21}, x_1}, p_{o_0, x_2}, \dots, p_{o_{21}, x_2}, p_{o_0, x_3}, \dots\}$ .

O algoritmo *Baum-Welch* usa como entrada series temporais de observações para calcular os parâmetros do HMM. Então, após a aplicação do algoritmo *Baum-Welch* utilizando as séries de taxa de perda de pacotes, a parametrização obtida para o modelo HMM foi a seguinte:

- Estado oculto  $x_1$ :

$$\text{Probabilidades de transição de estados} \begin{cases} p_{x_1, x_1} = 0,990976873434 \\ p_{x_1, x_2} = 0,00902312656604 \end{cases}$$

Probabilidades de emissão de símbolos:

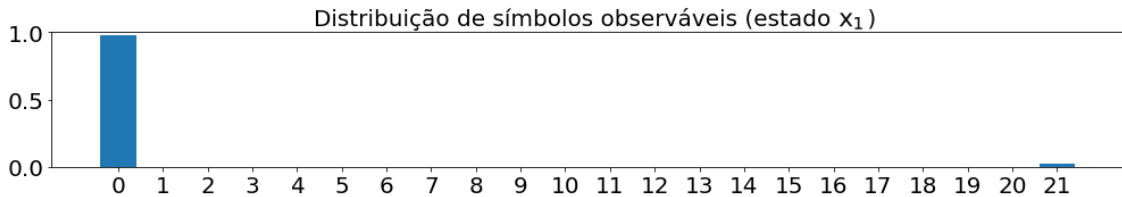


Figura 4.4: Distribuição da prob. de taxas de perda de pacotes no estado oculto  $x_1$

- Estado oculto  $x_2$ :

Probabilidades de transição de estados  $\begin{cases} p_{x_2,x_1} = 0,0151449439423 \\ p_{x_2,x_2} = 0,980538400218 \\ p_{x_2,x_3} = 0,00431665583963 \end{cases}$

Probabilidades de emissão de símbolos:

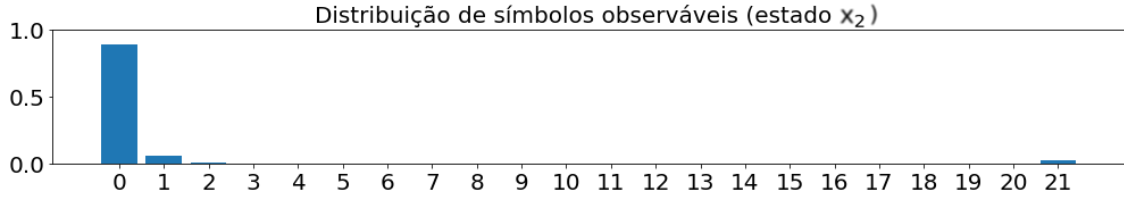


Figura 4.5: Distribuição da prob. de taxas de perda de pacotes no estado oculto  $x_2$

- Estado oculto  $x_3$ :

Probabilidades de transição de estados  $\begin{cases} p_{x_3,x_2} = 0,0209668007834 \\ p_{x_3,x_3} = 0,969657866357 \\ p_{x_3,x_4} = 0,00937533286004 \end{cases}$

Probabilidades de emissão de símbolos:

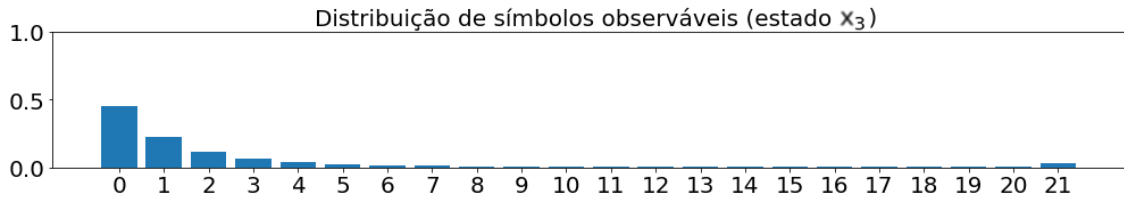


Figura 4.6: Distribuição da prob. de taxas de perda de pacotes no estado oculto  $x_3$

- Estado oculto  $x_4$ :

Probabilidades de transição de estados  $\begin{cases} p_{x_4,x_3} = 0,0396534402291 \\ p_{x_4,x_4} = 0,960346559771 \end{cases}$

Probabilidades de emissão de símbolos:

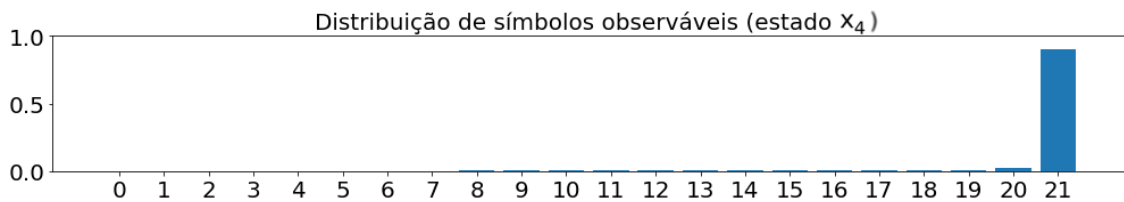


Figura 4.7: Distribuição da prob. de taxas de perda de pacotes no estado oculto  $x_4$

A partir do modelo HMM, será usado o algoritmo de *Viterbi* para inferir a sequência mais provável de estados ocultos a partir dos símbolos observados. Logo, com este algoritmo, pode-se inferir as condições de rede a cada minuto em cada série temporal de taxa de perda de pacotes. A Figura 4.8 ilustra um exemplo de uma série de taxa de perda de pacotes de um usuário em um dia e a respectiva sequência de estados ocultos que representam as condições de rede do usuário a cada minuto.

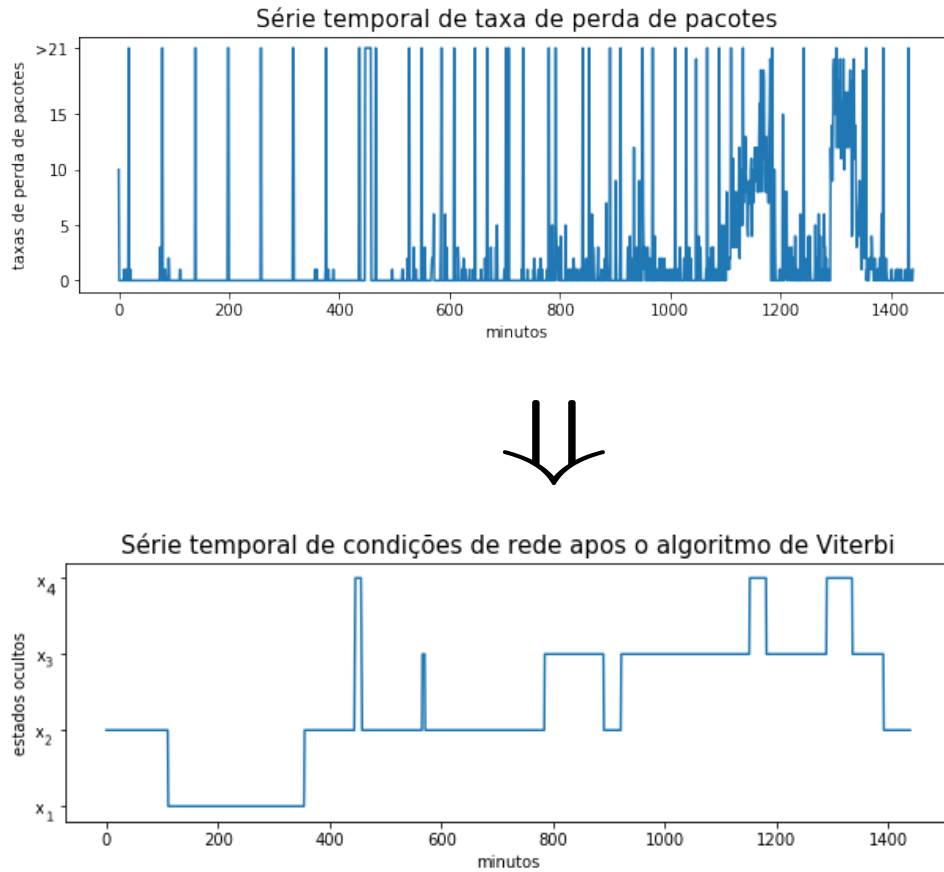


Figura 4.8: Série temporal de taxa de perda mapeada em estados ocultos do HMM

De acordo com as probabilidades de emissão de símbolos de cada estado oculto apresentadas nas Figuras 4.4, 4.5, 4.6 e 4.7 e com o exemplo da obtenção das séries de estados ocultos pela aplicação do algoritmo de *Viterbi* 4.8; percebe-se que o estado oculto  $x_3$  apresenta a distribuição com maior média de perda de pacotes, enquanto os estados ocultos  $x_1$  e  $x_2$  possuem altas probabilidades de gerar zero perda de pacote (observação  $o_0$ ). O estado oculto  $x_4$ , no entanto, tem maior probabilidade de gerar o símbolo  $o_{21}$  que representa *missing values*.

Em [7], para o objetivo de encontrar pontos de mudança nas séries, foi definido que um estado oculto representando alta perda de pacotes e outro representando indisponibilidade delimitam juntos os pontos de mudança. De maneira semelhante, para o objetivo dessa dissertação que engloba a previsão de condição de rede e

otimização do alarme de problemas na rede, também foi definido que os estados ocultos  $x_3$  e  $x_4$  são as condições de rede as quais o método busca prever e influenciar no acionamento do alarme, quando possível de acordo com o orçamento do ISP dado pelo número máximo de alarmes. Por isso, os estados ocultos  $x_3$  e  $x_4$  passam a ser representados por um único rótulo  $x_r$ , para facilitar a identificação dos estados ocultos críticos para o método. Enquanto isso, os demais estados ocultos ( $x_1$  e  $x_2$ ) passam a ser representados pelo rótulo  $x_b$ , para facilitar a identificação dos estados que não são críticos para o método.

A simplificação dos estados ocultos  $x_1$ ,  $x_2$ ,  $x_3$  e  $x_4$  para os rótulos  $x_r$  e  $x_b$  modificou todas as séries do *dataset* de séries de condições de rede para apenas dois níveis de condição. Na Figura 4.9, encontra-se um exemplo da obtenção de uma nova série de rótulos a partir da série de estados ocultos da Figura 4.8.

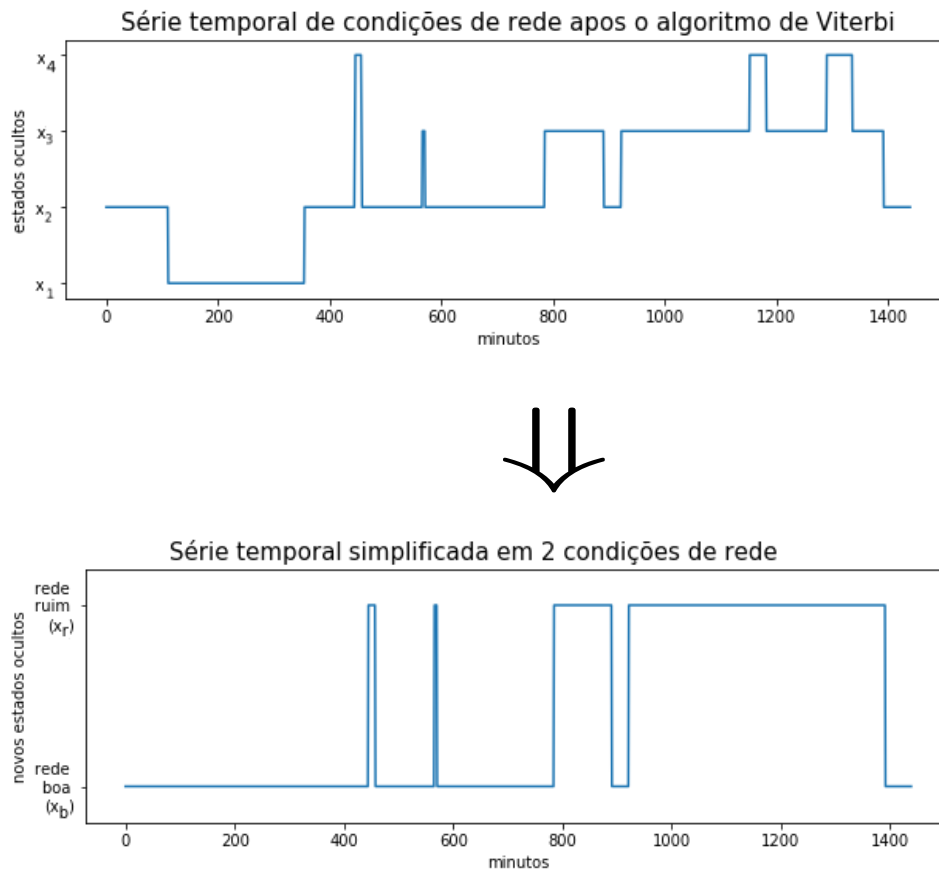


Figura 4.9: Simplificação da série de estados ocultos em 2 condições de rede

Com isso, foi necessário criar um novo HMM com apenas dois estados ocultos ( $x_r$  e  $x_b$ ). Tendo as novas séries temporais com apenas dois níveis de condição de rede, as novas probabilidades de transição de estados ocultos podem ser calculadas diretamente pelas frequências dessas novas transições.

Na nova HMM, as probabilidades de emissão de observações por estado oculto ( $x_r$  e  $x_b$ ) foram calculadas a partir das frequências dos símbolos observáveis nas séries

de taxa de perda de pacotes dado as amostras correspondentes nas novas séries com dois níveis de condição de rede.

A Figura 4.10 ilustra o novo HMM com apenas dois estados, rede boa ( $x_b$ ) e rede ruim ( $x_r$ ), e os novos parâmetros calculados.

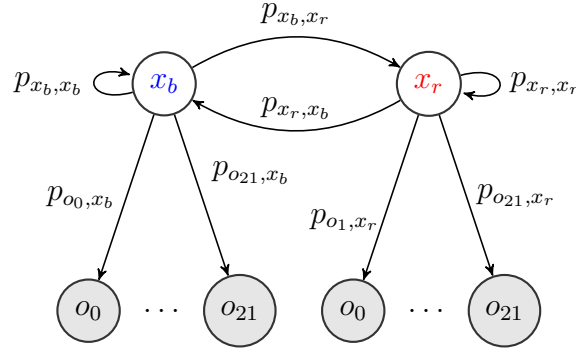


Figura 4.10: Nova cadeia de *Markov* oculta utilizada na metodologia.

- Probabilidades de transição de estados:

$$\begin{aligned}
 p_{x_b, x_b} &= 0,9978416721 & p_{x_b, x_r} &= 0,0021583279 \\
 p_{x_r, x_b} &= 0,0104834004 & p_{x_r, x_r} &= 0,9895165996
 \end{aligned}$$

- Probabilidades de emissão de símbolos, dado o estado oculto  $x_b$ :

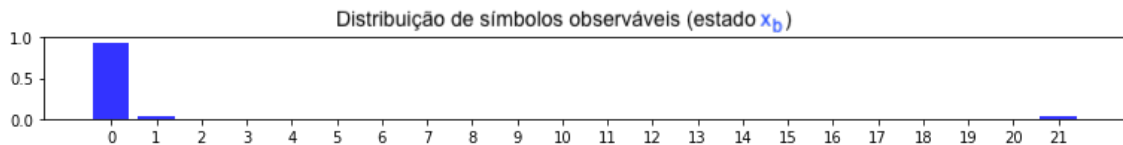


Figura 4.11: Distribuição da prob. de taxas de perda de pacotes no estado  $x_b$

- Probabilidades de emissão de símbolos, dado o estado oculto  $x_r$ :

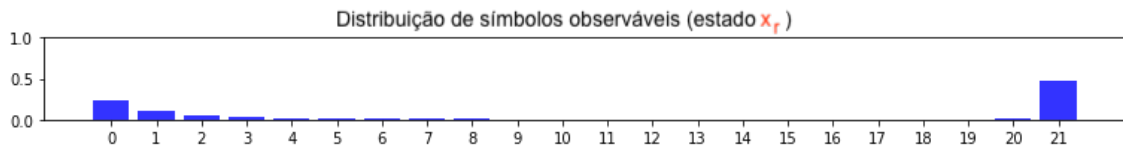


Figura 4.12: Distribuição da prob. de taxas de perda de pacotes no estado  $x_r$

Esta etapa teve o objetivo de obter, através de um HMM, séries temporais de condições de rede por minuto a partir de séries temporais de taxa de perda de pacotes. Essas novas séries temporais de condições de rede serão a informação principal de entrada para a terceira etapa do método que será descrita na próxima seção.



### 4.3 3ª etapa do método: MDP e algoritmo *Value Iteration*

A decisão de quando deve ser acionado um alarme de manutenção na rede pode ser tomada de diversas formas mais simples do que a proposta neste trabalho, com o algoritmo *Value Iteration*. A forma mais simples de decidir em acionar ou não um alarme para manutenção seria, por exemplo, utilizando os dados das séries temporais de taxa de perda de pacotes. Neste caso, o projetista poderia encontrar um valor a partir do qual a taxa de perda de pacotes causa uma piora na qualidade de serviço do cliente. Então, bastaria usar esse valor como limitante para esta decisão do alarme.

Outra forma simples de inferir quando deve haver uma manutenção da rede poderia ser utilizando, apenas, a informação da cadeia de *Markov* oculta gerada na etapa anterior. Uma aplicação comum para o algoritmo de *Viterbi* é a detecção de pontos de mudança em séries e, portanto, quando o padrão da série mudasse para um estado oculto significando rede ruim, ou seja, um estado com maior probabilidade de emitir taxas altas de perda de pacotes, o alarme de manutenção da rede seria acionado.

Em ambas as abordagens acima, não estão sendo considerados os limites de recursos diários para tais manutenções na rede. Portanto, assume-se que inúmeras manutenções poderiam ser acionadas e suspensas a qualquer instante da série. Com o algoritmo *Value Iteration* (VI) descrito nesta etapa do método, será inclusa, no modelo, a informação de quantas manutenções já aconteceram na série, até que esta contagem alcance um certo limite diário. Busca-se, ao final dessa etapa, a otimização desses recursos.

Então, para a descrição do modelo MDP  $\langle \mathcal{S}, \mathcal{A}, \mathbf{P}, \mathcal{R} \rangle$  que será utilizado nesta etapa, começamos com a definição das possíveis ações. Uma ação  $a^{(t)}$ , num instante  $t$  da série tal que  $1 \leq t \leq 1440$ , pertence ao conjunto  $\mathcal{A} = \{0, 1\}$ . Assume-se que cada dia começa sem acionamentos de alarmes ( $a^{(1)} = 0$ ) para manutenção dos nós da rede dos usuários. As ações  $a = 0$  e  $a = 1$  representam as seguintes decisões:

- $a = 0 \equiv$  manter a decisão do instante anterior quanto à manutenção:
  - manter o alarme desligado; ou
  - manter o alarme ligado.
- $a = 1 \equiv$  trocar a decisão do instante anterior:
  - ligar o alarme; ou
  - desligar o alarme.

Seguindo na descrição das características do modelo, tem-se a definição de três variáveis que compõem os estados desse MDP. Elas se tratam de variáveis que se

alteram, por sua vez, de acordo com outra variável discreta  $t$ , lembrando que  $t$  pode assumir apenas valores de números naturais entre os limites  $1 \leq t \leq 1440$ . Assim,  $t$  corresponde aos instantes de tempo, em minutos, de cada série temporal diária. Portanto, as definições das três variáveis de estado são as seguintes:

- $l^{(t)} \equiv$  o modo do alarme de manutenção em  $t$ , tal que  $l^{(t)} \in \{D, L\}$  e:
  - se  $l^{(t)} = \begin{cases} D \Rightarrow \text{alarme de manutenção está desligado em } t \\ L \Rightarrow \text{alarme de manutenção está ligado em } t \end{cases}$
- $n^{(t)} \equiv$  número de alarmes acionados na série até o instante  $t$ .
  - $0 \leq n^{(t)} \leq n_{max}$  ;
  - $n_{max} \equiv$  valor máximo de acionamentos por série (definido pelo projetista).
- $\tilde{v}^{(t)} \equiv [x(t-w+1), x(t-w+2), \dots, x(t-1), x(t)]$ ,  $\forall t \geq w$ 
  - $w \equiv$  tamanho do vetor  $\tilde{v}^{(t)}$  (estipulado pelo projetista);
  - condição de rede em  $t$  e nos instantes anteriores dentro de  $w$ .

A Figura 4.13 ilustra um exemplo de um vetor  $\tilde{v}^{(t)}$  obtido para uma série temporal de condições de rede.

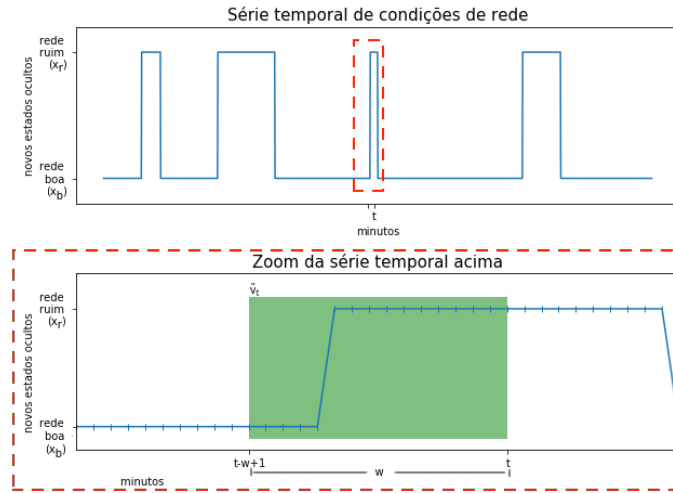


Figura 4.13: Exemplo de vetor  $\tilde{v}^{(t)}$  numa série temporal de condições de rede.

Cada estado do MDP num instante qualquer da série,  $s^{(t)} \in \mathcal{S}$ , é definido pelo vetor de variáveis de estados conforme apresentado abaixo.

$$s^{(t)} \leftarrow (l^{(t)}, n^{(t)}, \tilde{v}^{(t)}) \quad (4.1)$$

A partir do HMM obtido na segunda etapa da metodologia, o espaço de estados  $\mathcal{S}$  do MDP é definido da seguinte forma:

- Dois estados ocultos do HMM, ou seja,  $x^{(t)} \in \{x_b, x_r\}$  ;
- Limite de 1 alarme por dia,  $n_{max} = 1$  ;
- Vetor  $\tilde{v}^{(t)}$  de tamanho  $w$  .
- Logo, temos o conjunto:
 
$$\{v_1 = (x_b, \dots, x_b, x_b, x_b), v_2 = (x_b, \dots, x_b, x_b, x_r), \dots, v_m = (x_r, \dots, x_r, x_r, x_r)\}$$
 tal que:
  - qualquer  $\tilde{v}^{(t)} \in \{v_1, v_2, \dots, v_m\}$  ;
  - $m = 2^w$ , pois há somente dois valores possíveis para  $x^{(t)}$ .
- Adicionando os possíveis valores de  $l^{(t)}$  e  $n^{(t)}$ , temos o espaço de estados  $\mathcal{S}$  do modelo MDP:

$$\mathcal{S} = \{s_1, \dots, s_m, s_{m+1}, \dots, s_{2m}, \dots, s_{3m}, \dots, s_{4m}\}$$

$$\mathcal{S} = \{(D, 0, v_1), \dots, (D, 0, v_m), (L, 0, v_1), \dots, (L, 0, v_m), \dots, (D, 1, v_m), \dots, (L, 1, v_m)\}$$

Para exemplificar o espaço de estados do modelo MDP, descrito acima, suponha um cenário com a escolha de  $w = 3$

- Exemplo 1:
  - vetor  $\tilde{v}^{(t)}$  de tamanho  $w = 3$  .
  - Logo, o espaço de estados do modelo MDP é:

$$\mathcal{S} = \begin{Bmatrix} s_1, & s_2, & s_3, & s_4, & s_5, & s_6, & s_7, & s_8, & s_9, \\ s_{10}, & s_{11}, & s_{12}, & s_{13}, & s_{14}, & s_{15}, & s_{16}, & s_{17}, & s_{18}, \\ s_{19}, & s_{20}, & s_{21}, & s_{22}, & s_{23}, & s_{24}, & s_{25}, & s_{26}, & s_{27}, \\ s_{28}, & s_{29}, & s_{30}, & s_{31}, & s_{32} \end{Bmatrix}$$

$$\mathcal{S} = \left\{ \begin{array}{l} (D, 0, (x_b, x_b, x_b)), (D, 0, (x_b, x_b, x_r)), (D, 0, (x_b, x_r, x_b)), (D, 0, (x_b, x_r, x_r)), \\ (D, 0, (x_r, x_b, x_b)), (D, 0, (x_r, x_b, x_r)), (D, 0, (x_r, x_r, x_b)), (D, 0, (x_r, x_r, x_r)), \\ (L, 0, (x_b, x_b, x_b)), (L, 0, (x_b, x_b, x_r)), (L, 0, (x_b, x_r, x_b)), (L, 0, (x_b, x_r, x_r)), \\ (L, 0, (x_r, x_b, x_b)), (L, 0, (x_r, x_b, x_r)), (L, 0, (x_r, x_r, x_b)), (L, 0, (x_r, x_r, x_r)), \\ (D, 1, (x_b, x_b, x_b)), (D, 1, (x_b, x_b, x_r)), (D, 1, (x_b, x_r, x_b)), (D, 1, (x_b, x_r, x_r)), \\ (D, 1, (x_r, x_b, x_b)), (D, 1, (x_r, x_b, x_r)), (D, 1, (x_r, x_r, x_b)), (D, 1, (x_r, x_r, x_r)), \\ (L, 1, (x_b, x_b, x_b)), (L, 1, (x_b, x_b, x_r)), (L, 1, (x_b, x_r, x_b)), (L, 1, (x_b, x_r, x_r)), \\ (L, 1, (x_r, x_b, x_b)), (L, 1, (x_r, x_b, x_r)), (L, 1, (x_r, x_r, x_b)), (L, 1, (x_r, x_r, x_r)) \end{array} \right\}$$

Voltando à modelagem MDP, percebe-se que ainda é necessário encontrar os valores da matriz de probabilidades de transição entre os estados  $\mathbf{P}$ . Num instante qualquer  $t$ , a probabilidade de transição entre estados pertencente na matriz  $\mathbf{P}$  depende da ação em  $t$ . Por isso, define-se uma matriz  $\mathbf{P}_{a_0}$  referente a matriz de probabilidade de transição entre estados quando  $a^{(t)} = 0$  e outra matriz  $\mathbf{P}_{a_1}$  referente a matriz de probabilidade de transição entre estados quando  $a^{(t)} = 1$ , tais que  $\mathbf{P} = (\mathbf{P}_{a_0}, \mathbf{P}_{a_1})$ .

As mudanças dos valores das variáveis de estado  $l^{(t)}$  e  $n^{(t)}$  também são afetadas diretamente a partir da ação em  $t$ , a ação  $a^{(t)}$ . Porém, a variável representada pelo vetor  $\tilde{v}^{(t)}$  independe de  $a^{(t)}$ , ela depende somente das séries de condições de rede. Então, primeiramente, para montar as matrizes  $\mathbf{P}_{a_0}$  e  $\mathbf{P}_{a_1}$ , devemos criar uma matriz auxiliar com as probabilidades de todas as possíveis transições de  $\tilde{v}^{(t)} = \tilde{v}_i$  para  $\tilde{v}^{(t+1)} = \tilde{v}_j$ , chamaremos essa matriz de  $\mathbf{P}_v(\tilde{v}_i, \tilde{v}_j)$ . Então, os elementos da matriz  $\mathbf{P}_v$  são obtidos através da frequência de transições entre dois vetores adjacentes  $\tilde{v}_i$  e  $\tilde{v}_j$ .

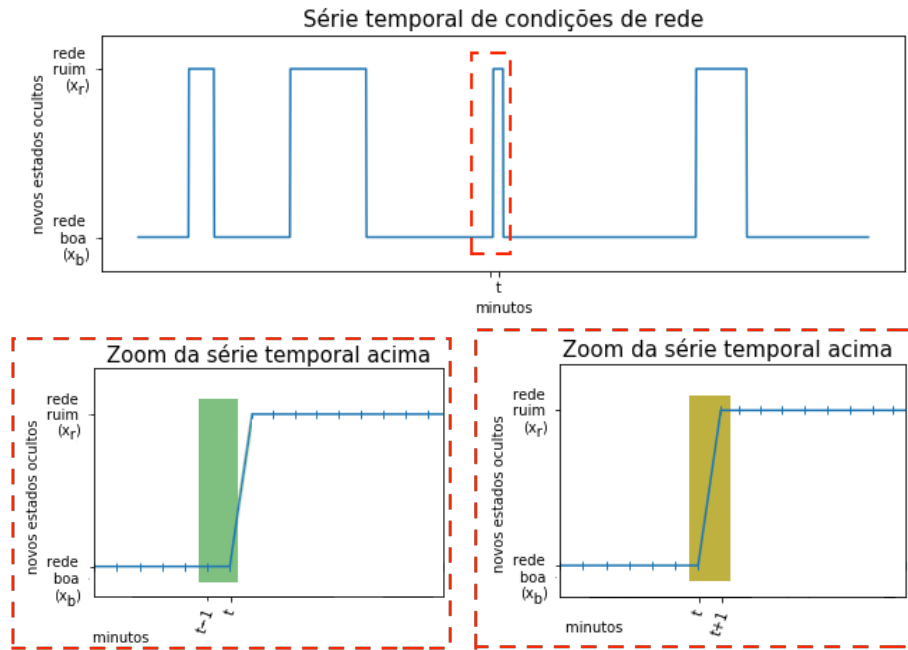


Figura 4.14: Exemplo de uma transição entre dois vetores adjacentes  $\tilde{v}_i$  e  $\tilde{v}_j$  numa série

Na Figura 4.14, por exemplo, está ilustrada a transição entre dois vetores adjacentes  $\tilde{v}_i$  e  $\tilde{v}_j$  em que  $w = 2$ ,  $\tilde{v}_i = (x_b, x_b)$  e  $\tilde{v}_j = (x_b, x_r)$ . Então, é calculada a frequência de todos os instantes  $t$ , na mesma série e em todas as outras séries de condições de rede, em que  $\tilde{v}^{(t)} = \tilde{v}_i$  e  $\tilde{v}^{(t+1)} = \tilde{v}_j$ . Com a frequência dessa transição de  $\tilde{v}_i$  para  $\tilde{v}_j$ , obtém-se o valor de probabilidade que preenche a posição  $(\tilde{v}_i, \tilde{v}_j)$  da matriz  $\mathbf{P}_v$ . Esse mesmo procedimento pode ser feito para calcular todas as outras posições de  $\mathbf{P}_v$ .

Abaixo, temos um exemplo do preenchimento de uma matriz  $\mathbf{P}_v$  para um *dataset* de séries de condições de rede,  $w = 3$  e  $n_{max} = 1$ .

$$\mathbf{P}_v = \begin{array}{c} \begin{array}{cccccccc} \begin{array}{c} (\mathbf{x}_b, \mathbf{x}_b, \mathbf{x}_b) \\ \parallel \\ \tilde{v}_j \end{array} & \begin{array}{c} (\mathbf{x}_b, \mathbf{x}_b, \mathbf{x}_r) \\ \parallel \\ \tilde{v}_j \end{array} & \begin{array}{c} (\mathbf{x}_b, \mathbf{x}_r, \mathbf{x}_b) \\ \parallel \\ \tilde{v}_j \end{array} & \begin{array}{c} (\mathbf{x}_b, \mathbf{x}_r, \mathbf{x}_r) \\ \parallel \\ \tilde{v}_j \end{array} & \begin{array}{c} (\mathbf{x}_r, \mathbf{x}_b, \mathbf{x}_b) \\ \parallel \\ \tilde{v}_j \end{array} & \begin{array}{c} (\mathbf{x}_r, \mathbf{x}_b, \mathbf{x}_r) \\ \parallel \\ \tilde{v}_j \end{array} & \begin{array}{c} (\mathbf{x}_r, \mathbf{x}_r, \mathbf{x}_b) \\ \parallel \\ \tilde{v}_j \end{array} & \begin{array}{c} (\mathbf{x}_r, \mathbf{x}_r, \mathbf{x}_r) \\ \parallel \\ \tilde{v}_j \end{array} \\ \left[ \begin{array}{cccccccc} 0,9985 & 0,0015 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0,5 & 0,5 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0,0208 & 0,9792 \end{array} \right] & \begin{array}{l} \tilde{v}_i = (\mathbf{x}_b, \mathbf{x}_b, \mathbf{x}_b) \\ \tilde{v}_i = (\mathbf{x}_b, \mathbf{x}_b, \mathbf{x}_r) \\ \tilde{v}_i = (\mathbf{x}_b, \mathbf{x}_r, \mathbf{x}_b) \\ \tilde{v}_i = (\mathbf{x}_b, \mathbf{x}_r, \mathbf{x}_r) \\ \tilde{v}_i = (\mathbf{x}_r, \mathbf{x}_b, \mathbf{x}_b) \\ \tilde{v}_i = (\mathbf{x}_r, \mathbf{x}_b, \mathbf{x}_r) \\ \tilde{v}_i = (\mathbf{x}_r, \mathbf{x}_r, \mathbf{x}_b) \\ \tilde{v}_i = (\mathbf{x}_r, \mathbf{x}_r, \mathbf{x}_r) \end{array} \end{array}$$

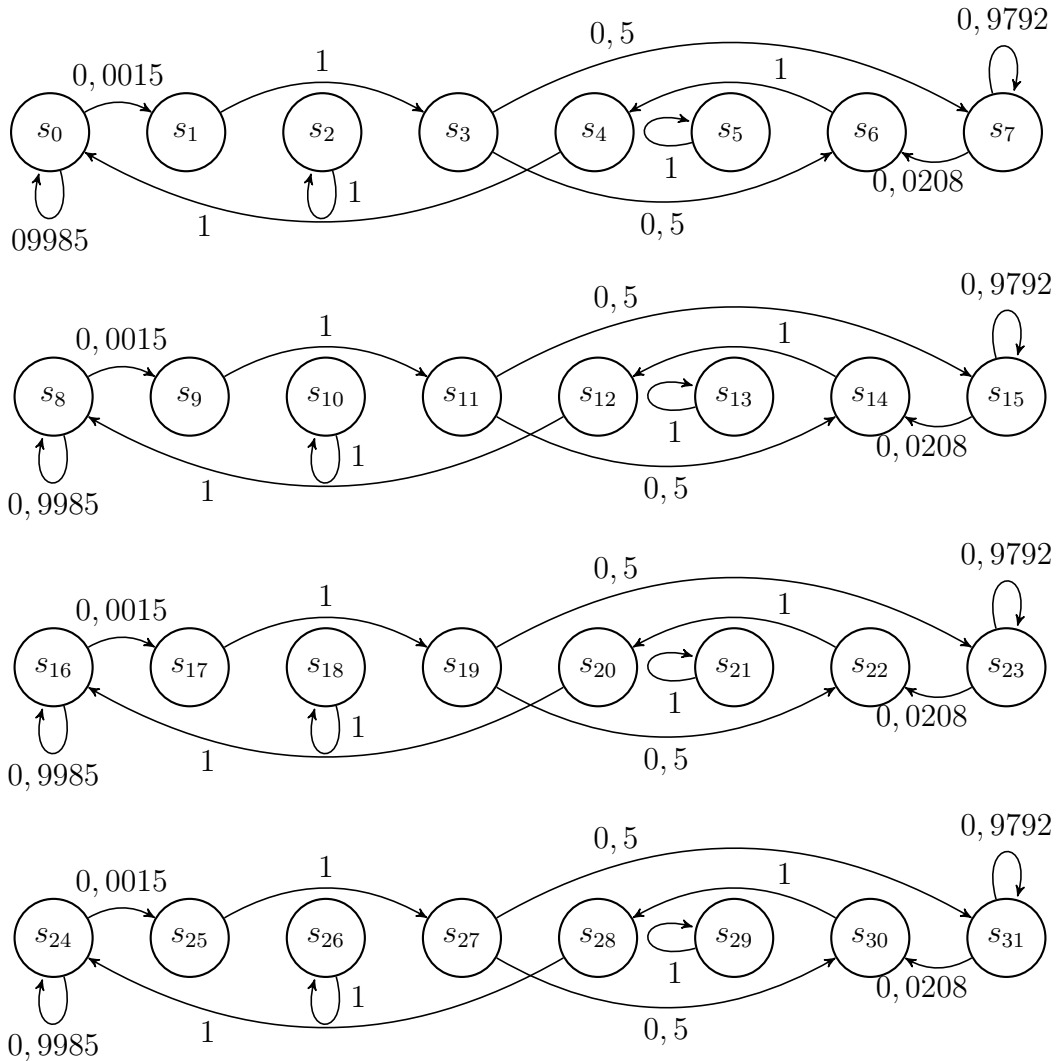
No exemplo acima, através da primeira linha da matriz  $\mathbf{P}_v$ , podemos ver que, numa transição qualquer de  $\tilde{v}_i$  para  $\tilde{v}_j$ , se  $\tilde{v}_i = (\mathbf{x}_b, \mathbf{x}_b, \mathbf{x}_b)$ , então, temos probabilidade de 0,9985 para  $\tilde{v}_j = (\mathbf{x}_b, \mathbf{x}_b, \mathbf{x}_b)$  e probabilidade de 0,0015 para  $\tilde{v}_j = (\mathbf{x}_b, \mathbf{x}_b, \mathbf{x}_r)$ .

Após o cálculo da matriz  $\mathbf{P}_v$ , basta adicionar as variáveis  $l^{(t)}$  e  $n^{(t)}$  para obter as matrizes  $\mathbf{P}_{a_0}$  e  $\mathbf{P}_{a_1}$ , de acordo com as influências das possíveis ações  $a^{(t)} \in \mathcal{A} = \{0, 1\}$ .

Seja, por exemplo, o número máximo de alarmes por série projetado para o valor  $n_{max} = 1$ . Sendo assim,  $\mathbf{P}_{a_0}(s, s')$  corresponde a matriz de probabilidade de troca do estado  $s^{(t)} = s$  para  $s^{(t+1)} = s'$  quando  $a^{(t)} = 0$ , tal que  $s, s' \in \mathcal{S}$ . Relembrando que se  $a^{(t)} = 0$ , os valores de  $l^{(t)}$  e  $n^{(t)}$  se mantêm respectivamente em  $l^{(t+1)}$  e  $n^{(t+1)}$ ; então, na matriz  $\mathbf{P}_{a_0}$ , as posições  $(s, s')$  que são dos seguintes formatos  $((D, 0, \tilde{v}_i), (D, 0, \tilde{v}_j))$ ,  $((D, 1, \tilde{v}_i), (D, 1, \tilde{v}_j))$ ,  $((L, 0, \tilde{v}_i), (L, 0, \tilde{v}_j))$  e  $((L, 1, \tilde{v}_i), (L, 1, \tilde{v}_j))$  possuem os mesmos valores das posições de  $(\tilde{v}_i, \tilde{v}_j)$  na matriz  $\mathbf{P}_v$ . Para as demais posições, que não satisfazem a condição da ação  $a = 0$ , a probabilidade é nula na matriz  $\mathbf{P}_{a_0}$ . Seja  $0_{8,8}$  uma matriz nula de tamanho  $8 \times 8$ , temos o seguinte preenchimento das posições de  $\mathbf{P}_{a_0}$ .

$$\mathbf{P}_{a_0} = \begin{matrix}
\begin{matrix} s' = (D, 0, \tilde{v}_j) \\ s' = (L, 0, \tilde{v}_j) \\ s' = (D, 1, \tilde{v}_j) \\ s' = (L, 1, \tilde{v}_j) \end{matrix} \\
\begin{bmatrix}
\mathbf{P}_v & 0_{8,8} & 0_{8,8} & 0_{8,8} \\
0_{8,8} & \mathbf{P}_v & 0_{8,8} & 0_{8,8} \\
0_{8,8} & 0_{8,8} & \mathbf{P}_v & 0_{8,8} \\
0_{8,8} & 0_{8,8} & 0_{8,8} & \mathbf{P}_v
\end{bmatrix}
\begin{matrix}
s = (D, 0, \tilde{v}_i) \\
s = (L, 0, \tilde{v}_i) \\
s = (D, 1, \tilde{v}_i) \\
s = (L, 1, \tilde{v}_i)
\end{matrix}
\end{matrix}$$

De acordo com  $\mathbf{P}_{a_0}$  do exemplo acima ( $w = 3$  e  $n_{max} = 1$ ), a cadeia de *Markov* abaixo pode ser usada para representar as probabilidades de troca de estados  $a = 0$ .

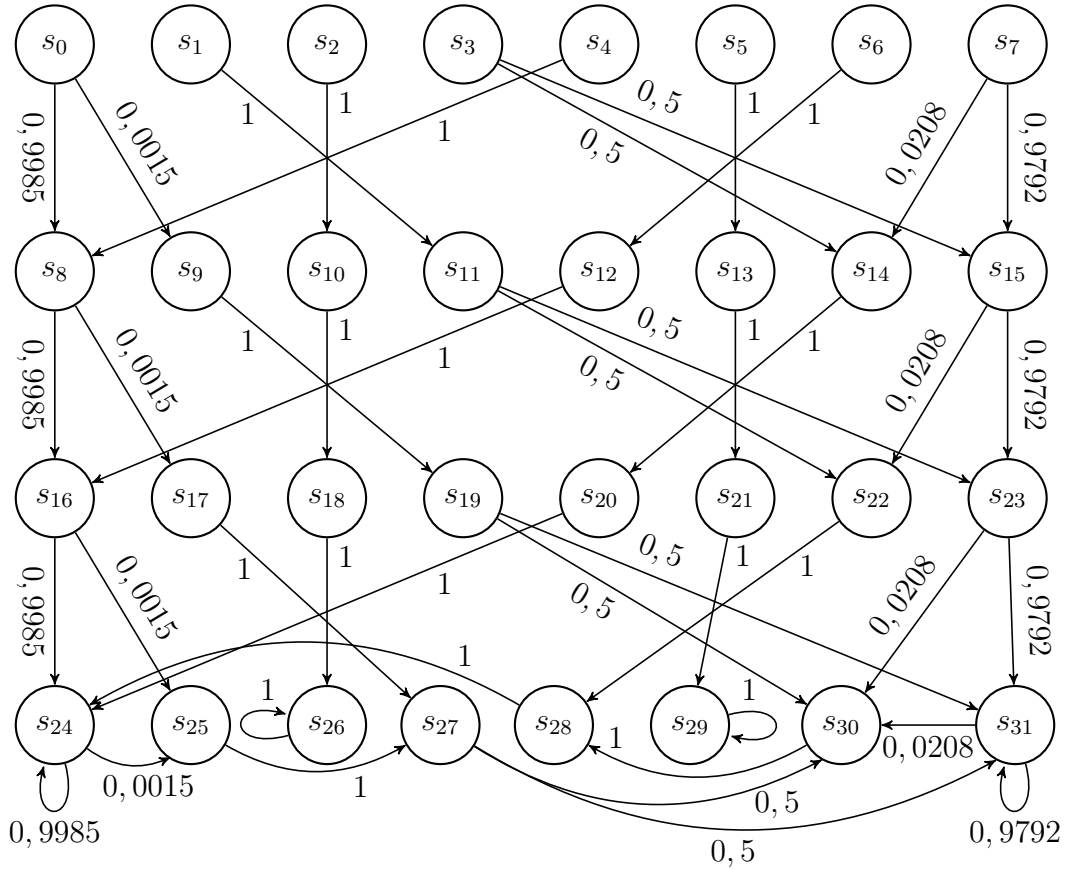


De maneira análoga, também é possível preencher os valores da matriz  $\mathbf{P}_{a_1}$ . Se  $a^{(t)} = 1$ , no mesmo exemplo com  $n_{max} = 1$ ; então os valores de  $(l^{(t)}, n^{(t)})$  se alteram para  $(l^{(t+1)}, n^{(t+1)})$  da seguinte forma: de  $(0, 0)$  para  $(1, 0)$ , de  $(1, 0)$  para  $(0, 1)$ , de

$(0, 1)$  para  $(1, 1)$  e de  $(1, 1)$  para  $(1, 1)$ . Logo, na matriz  $\mathbf{P}_{a_1}$ , as posições  $(s, s')$  que são dos formatos  $((D, 0, \tilde{v}_i), (L, 0, \tilde{v}_j))$ ,  $((L, 0, \tilde{v}_i), (D, 1, \tilde{v}_j))$ ,  $((D, 1, \tilde{v}_i), (L, 1, \tilde{v}_j))$  e  $((L, 1, \tilde{v}_i), (L, 1, \tilde{v}_j))$  possuem os mesmos valores do que em  $\mathbf{P}_v(\tilde{v}_i, \tilde{v}_j)$ . Para as demais posições, que não satisfazem a condição da ação  $a = 1$ , a probabilidade é nula em  $\mathbf{P}_{a_1}$ . Segue o preenchimento das posições de  $\mathbf{P}_{a_1}$ .

$$\mathbf{P}_{a_1} = \begin{matrix} & \begin{matrix} s' = (D, 0, \tilde{v}_j) \\ s' = (L, 0, \tilde{v}_j) \\ s' = (D, 1, \tilde{v}_j) \\ s' = (L, 1, \tilde{v}_j) \end{matrix} \\ \begin{matrix} s = (D, 0, \tilde{v}_i) \\ s = (L, 0, \tilde{v}_i) \\ s = (D, 1, \tilde{v}_i) \\ s = (L, 1, \tilde{v}_i) \end{matrix} & \begin{bmatrix} 0_{8,8} & \mathbf{P}_v & 0_{8,8} & 0_{8,8} \\ 0_{8,8} & 0_{8,8} & \mathbf{P}_v & 0_{8,8} \\ 0_{8,8} & 0_{8,8} & 0_{8,8} & \mathbf{P}_v \\ 0_{8,8} & 0_{8,8} & 0_{8,8} & \mathbf{P}_v \end{bmatrix} \end{matrix}$$

De acordo com  $\mathbf{P}_{a_1}$  do exemplo acima ( $w = 3$  e  $n_{max} = 1$ ), a cadeia de *Markov* abaixo pode ser usada para representar as probabilidades de troca de estados  $a = 1$ .



Para finalizar o modelo MDP, falta somente a definição da função de recompensas  $\mathcal{R}$  para cada transição de  $(s^{(t)} = s$  para  $(s^{(t+1)} = s'$  e de acordo com cada ação

$(a^{(t)} = a$ . Três possíveis valores de recompensa  $r_1$ ,  $r_2$  e  $r_3$  são escolhidos de acordo com a seguinte função  $\mathcal{R}(s, a, s')$ :

$$\mathcal{R}(s, a, s') = \begin{cases} r_1, & \text{se } \left\{ \begin{array}{l} (s^{(t)}, a^{(t)}, (D, n^{(t+1)} < n_{max}, (x^{(t-w+2)}, \dots, x_b)) \\ (s^{(t)}, a^{(t)}, (L, n^{(t+1)} < n_{max}, (x^{(t-w+2)}, \dots, x_r)) \end{array} \right\} \\ r_2, & \text{se } \left\{ \begin{array}{l} (s^{(t)}, a^{(t)}, (L, n^{(t+1)} < n_{max}, (x^{(t-w+2)}, \dots, x_b)) \\ (s^{(t)}, a^{(t)}, (D, n^{(t+1)} < n_{max}, (x^{(t-w+2)}, \dots, x_r)) \end{array} \right\} \\ r_3, & \text{se } (s^{(t)}, a^{(t)}, (1, n^{(t+1)} = n_{max}, \tilde{v}^{(t+1)})) \end{cases}$$

Analisando a expressão acima, verificamos que, se o número de alarmes ainda não foi excedido ( $n^{(t+1)} < n_{max}$ ), então, o sistema fornece uma recompensa  $r_1$  quando o agente desliga ou mantém desligado o alarme de manutenção ( $l^{(t+1)} = 0$ ) e a próxima condição da rede não é boa ( $x^{(t+1)} = x_b$ ). O mesmo acontece quando, nas mesmas condições, o agente liga ou mantém ligado o alarme de manutenção ( $l^{(t+1)} = 1$ ) e a próxima condição da rede é ruim ( $x^{(t+1)} = x_r$ ).

Seguindo nessa análise com o número de alarmes ainda não excedido ( $n^{(t+1)} < n_{max}$ ), o sistema fornece uma recompensa  $r_2$  quando o agente comete um erro de desligar ou manter desligado o alarme de manutenção ( $l^{(t+1)} = 0$ ), mas a próxima condição da rede é ruim ( $x^{(t+1)} = x_r$ ). A mesma recompensa é dada quando o agente liga ou mantém ligado o alarme de manutenção ( $l^{(t+1)} = 1$ ) indevidamente, pois a próxima condição da rede é boa ( $x^{(t+1)} = x_b$ ).

O terceiro valor de recompensa,  $r_3$ , é definido para o caso em que o limite de alarmes diários de manutenção da rede já alcançou o valor máximo ( $n^{(t+1)} = n_{max}$ ) e o sistema teve a ação de ligar ou manter o alarme ligado no instante seguinte ( $l^{(t+1)} = 1$ ), independentemente da próxima condição de rede ( $x^{(t+1)}$ ).

De acordo com a análise dos três valores de recompensa acima podemos perceber que elas precisam ser definidas de acordo com a seguinte regra:  $r_1 > r_2 > r_3$ . Sendo assim, os valores de  $r_1 = 1$ ,  $r_2 = 0,5$  e  $r_3 = 0$  foram determinados empiricamente através de engenharia reversa, ou seja, observando o desempenho do sistema. A escolha desses três valores de recompensa é uma tarefa muito importante para o desempenho do método porque, dependendo do valor dessas recompensas, o sistema *aprende* a economizar o acionamento de alarmes no caso de uma sequência curta de estados de condições ruins de rede para que este seja acionado no caso de uma sequência longa de estados de condições ruins de rede. Caso contrário, o sistema pode chegar facilmente ao limite de alarmes diários e conseqüentemente não acionar o alarme em uma situação onde deveria.

Uma vez concluída a descrição da modelagem MDP, o sistema está pronto para



a implementação do algoritmo *Value Iteration*. As entradas do algoritmo *Value Iteration* são as seguintes:

- Conjunto de estados  $\mathcal{S}$ ;
- Conjunto de ações  $\mathcal{A}$ ;
- A matriz de probabilidade de transição de estados  $\mathbf{P}$ ;
- A matriz de recompensa de transições de estados  $\mathcal{R}$ ;
- O fator de desconto  $\gamma$ .

Pode-se perceber que quase todas as entradas do algoritmo VI já estão definidas pela modelagem do MDP. O fator de desconto  $\gamma$ , porém, ainda precisa ser definido. Ele é um valor com os limites  $0 < \gamma < 1$ . A escolha desse parâmetro deve ser feita de acordo com a quantidade de passos futuros que se mantêm relevantes para uma ação num determinado momento. Isso significa que uma ação no início da série deve ter consequências nas escolhas de ações no fim da série. No caso de uma série temporal com 1440 pontos como nos conjuntos de dados desta metodologia, o fator de desconto deve ser da ordem de 1 para 1000 passos, portanto, o valor de  $\gamma = 0,999$  é uma boa escolha para esse parâmetro nesse sistema.

Para chegar nesse valor de  $\gamma = 0,999$ , foi feito um estudo com a aplicação do método em séries com dados sintéticos de condição de rede e variando  $\gamma$  entre 0 e 1. Foi constatado que, para  $0 < \gamma < 0,999$ , a recompensa instantânea de uma ação no início da série não contribuiu para uma ação no final da série pois, neste caso, as ações estavam separadas por mais de 1000 passos. Para  $0,999 \leq \gamma < 1$ , a recompensa instantânea de uma ação no início da série contribuiu para as ações ao longo da série; mas, com  $0,999 < \gamma < 1$ , o algoritmo demorou mais para convergir do que com  $\gamma = 0,999$ .

Finalmente, a saída deste algoritmo, que também significa o resultado dessa última etapa do método, é a política de ações ótima, representada pelo vetor  $\pi(s)$ . O vetor  $\pi(s)$  é obtido através da convergência de sucessivas iterações da Equação 3.6. A política ótima  $\pi(s)$  tem a informação da ação ótima para cada estado do MDP. Portanto,  $\pi(s)$  representa o aprendizado final do método.

No capítulo a seguir serão demonstrados experimentos realizados para exemplificar a metodologia apresentada neste capítulo.

# Capítulo 5

## Experimentos e análise de resultados

Neste capítulo, serão apresentados alguns experimentos utilizando a metodologia descrita no capítulo anterior. Tais experimentos foram realizados com *datasets* diferentes de treino para que seus resultados possam ser comparados entre si, além de suas respectivas avaliações de desempenho.

A seguir, são descritas primeiramente as métricas de avaliação do desempenho dos experimentos. Logo após, encontra-se a descrição de 2 experimentos utilizando medidas de taxa de perda de pacotes coletadas num ISP ao longo de 2 dias diferentes.

### 5.1 Métricas de avaliação de desempenho

Como em qualquer método de aprendizagem por reforço que encontra ações ótimas para determinados cenários; faz-se necessário, em cada experimento, comparar os resultados das decisões geradas pelo método com decisões coerentes que poderiam ser tomadas “manualmente” para resultar na melhor recompensa a longo prazo.

Relembrando que o objetivo da metodologia é a otimização de um alarme para sinalizar problemas na rede de um ISP considerando o seu orçamento. O método busca acionar o alarme somente quando o intervalo em que a rede permanece com condições ruins tem uma duração longa. Isso significa que se, por exemplo, o orçamento de um ISP for ajustado para apenas um acionamento do alarme por série ( $n_{max} = 1$ ) e se o método for treinado com apenas séries fictícias de condições de rede (tipo 1, Figura 5.1) em que há dois picos de condições ruins ( $x_r$ ), um com duração de 5 minutos e outro com duração de 100 minutos; então, o método deve aprender a acionar o alarme somente no maior pico da série.

Se, por exemplo, utilizando o mesmo orçamento do ISP ( $n_{max} = 1$ ), o método for treinado com outras séries fictícias de condições de rede (tipo 2, Figura 5.2) que possuam dois picos de condições ruins ( $x_r$ ), ambos com 100 minutos de duração; então, o método deve aprender a acionar o alarme logo no primeiro pico da série,

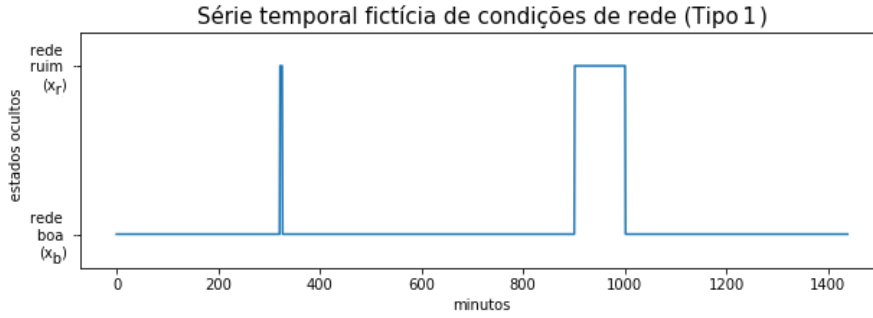


Figura 5.1: Exemplo de série temporal fictícia do tipo 1

uma vez que ele tem duração relevante para receber sinalização e o alarme, neste caso, ainda se encontra disponível para ser acionado.

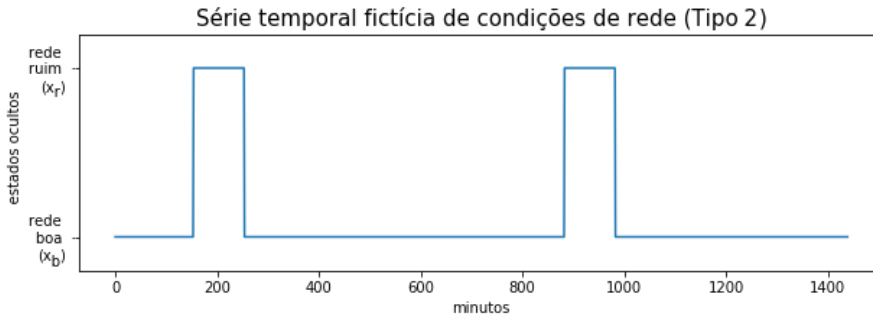


Figura 5.2: Exemplo de série temporal fictícia do tipo 2

Os experimentos com medições de taxas de perda de pacotes, que serão descritos na Seção 5.2, também possuem orçamento de ISP ajustado para um alarme por série ( $n_{max} = 1$ ).

Para a avaliação, adota-se a definição de que um “evento” ocorre na série toda vez que a condição de rede passa de  $x_b$  para  $x_r$  ou quando a série começa com a condição  $x_r$ . O fim de cada evento é marcado com a transição de  $x_r$  para  $x_b$  ou com o fim da série. Exemplificando, cada série das figuras 5.1 e 5.2 possuem 2 eventos.

Sendo assim, podemos preencher uma matriz de confusão para avaliar o desempenho do método em cada experimento, através da classificação de suas séries de teste segundo as seguintes definições:

- Verdadeiros positivos (ou TP, do inglês *true positives*):
  - série possui eventos com duração maior do que  $w$ : método aciona alarme em qualquer um dos eventos com duração maior do que  $w$ .
  - série possui eventos com duração menor do que  $w$ : método aciona alarme no evento de maior duração.
- Falsos positivos (ou FP, do inglês *false positives*):

- série possui eventos com duração maior do que  $w$ : método não aciona alarme em nenhum desses eventos.
- série possui eventos com duração menor do que  $w$ : método não aciona alarme no evento de maior duração.
- método mantém o alarme acionado quando o evento de série acaba
- Verdadeiros negativos (ou TN, do inglês *true negatives*):
  - método mantém desligado o alarme ao longo de séries sem eventos
- Falsos negativos (ou FN, do inglês *false negatives*):
  - método mantém desligado o alarme ao longo de séries com pelo menos um evento

De acordo com os valores encontrados para as definições acima, é possível calcular também os valores das seguintes medidas de desempenho do método: acurácia, precisão, *recall* e *F1-score*. A acurácia representa a porcentagem geral de acertos do método  $\left(\frac{TP + TN}{TP + FP + TN + FN}\right)$ . A precisão indica a porcentagem de acertos quando o método indicou pelo menos um acionamento do alarme na série  $\left(\frac{TP}{TP + FP}\right)$ . O *recall* ou a sensibilidade é a porcentagem de acertos quando a série necessitava um acionamento de alarme  $\left(\frac{TP}{TP + FN}\right)$ . O *F1-score* calcula a média harmônica entre a precisão e o *recall*, ou seja,  $\left(\frac{TP}{TP + \frac{FP + FN}{2}}\right)$ .

## 5.2 Experimentos

Os experimentos a seguir têm a finalidade de demonstrar como as diferenças nas características dos *datasets* resultam em políticas ótimas diferentes para as ações do alarme de manutenção do ISP. Cada experimento utiliza, na sua primeira etapa do método, um dia de coleta de séries temporais de taxa de perda de pacotes em roteadores de clientes de um ISP. O experimento 1 possui séries temporais de 2359 clientes, enquanto o experimento 2 conta com 2689 séries.

Como está descrito no capítulo 4, essas séries temporais de taxa de perda de pacotes são coletadas e utilizadas, como entrada, na segunda etapa do método. Nessa etapa, há uma parametrização de um HMM usando o algoritmo *Baum-Welch* de acordo com as séries de taxa de perda de pacotes coletadas na primeira etapa.

Ainda nesta segunda etapa, logo após o algoritmo *Baum-Welch*, é executado o algoritmo de *Viterbi*, utilizando como entrada novamente as séries de taxa de perda

de pacotes e o modelo HMM parametrizado, para inferir séries de condições de rede mais prováveis de terem gerado as séries de taxa de perda de pacotes.

Assim como mencionado anteriormente no capítulo de metodologia e segundo [7], o HMM parametrizado possui 4 estados ocultos representando 4 condições de rede. O algoritmo de *Viterbi* é aplicado e gera séries com 4 condições de rede. Posteriormente, para simplificação do método, as 4 condições de rede são resumidas em apenas 2 condições de rede: as condições de rede 1 e 2 foram consideradas “rede boa” e as condições 3 e 4 foram consideradas “rede ruim”. As Figuras 5.3 e 5.6 exemplificam duas séries onde as condições de rede foram agrupadas.

Portanto, ao fim dessa etapa, temos séries de condições de rede cujos pontos possuem apenas 2 opções de estados ocultos: condição boa de rede ( $x_b$ ) e condição ruim de rede ( $x_r$ ). Sendo assim, numa série qualquer de condição de rede  $\Phi = (X^{(1)}, X^{(2)}, \dots, X^{(\tau)})$ , o valor de um ponto qualquer  $X^{(t)} \in \{x_b, x_r\}$ ,  $1 \leq t \leq \tau$ .

Com as séries temporais de condições de rede resultantes da segunda etapa do método, podemos inicializar o procedimento da terceira etapa. Essa é a etapa de otimização de um alarme para sinalizar manutenção na rede de um ISP, de acordo com as condições de rede momentâneas e o orçamento diário do ISP. Esse procedimento é feito através da aplicação do algoritmo Value Iteration em um modelo MDP. Ao fim do treinamento com o *Value Iteration*, o sistema deve ligar ou manter ligado um alarme nos instantes da série em que houver previsão de uma longa permanência no nível de condição ruim de rede para os instantes futuros da série. O sistema deve desligar o alarme no instante em que perceber condições boas na série.

No modelo MDP, devemos escolher qual será o limite de alarmes por série temporal,  $n_{max}$ . No caso dos próximos experimentos, esse limite é unitário,  $n_{max} = 1$ ; então, o valor de  $n^{(t)}$  é um elemento do conjunto  $\{0, 1\}$ . Outra escolha necessária para o método, é a quantidade de pontos seguidos na série temporal de condições de rede que serão analisados a cada instante  $t$ . Essa quantidade significa o tamanho  $w$  do vetor  $v^{(t)}$  que compõe o estado  $s^{(t)}$ . Para ambos os seguintes experimentos, o valor escolhido para  $w$  é igual a 15. Portanto, num instante  $t$  de uma série qualquer, cada estado é dado por:  $s^{(t)} = (l^{(t)}, n^{(t)}, \underbrace{\tilde{v}^{(t)} = (x^{(t-14)}, x^{(t-13)}, \dots, x^{(t-2)}, x^{(t-1)}, x^{(t)})}_{w=15})$ .

Concluídas as escolhas dos parâmetros das variáveis de estado, falta a criação de  $\mathbf{P}$  e  $\mathcal{R}$ , respectivamente a matriz de probabilidade de transição entre estados dado uma ação e a função de recompensas dessas transições. Os valores de  $\mathbf{P}$  são preenchidos segundo a metodologia apresentada no capítulo anterior. A função  $\mathcal{R}$  também segue a regra apresentada no capítulo anterior, adicionando a seguinte informação:  $(r_1, r_2, r_3) = (1; 0, 5; 0)$ .

Com isso, está finalizada a modelagem MDP e próximo passo é a aplicação do algoritmo *Value Iteration*. O valor do fator de desconto  $\gamma = 0,999$  definido na

metodologia é utilizado em ambos os experimentos.

Estas foram as informações e definições comuns dos próximos dois experimentos. A seguir, estão descritas as particularidade de cada *dataset* e de cada resultado obtido, ou seja, cada política ótima resultante.

### 5.2.1 Experimento 1

O *dataset* deste experimento é formado por 2359 séries de medidas de roteadores domésticos de um ISP ao longo de um único dia de coleta. Todas as séries deste *dataset* seguem para os processamentos da segunda etapa do método. A Figura 5.3 abaixo ilustra a série temporal real, a série de condições de rede obtida após execução do algoritmo de *Viterbi* e a série com 4 condições de rede agrupadas em 2.

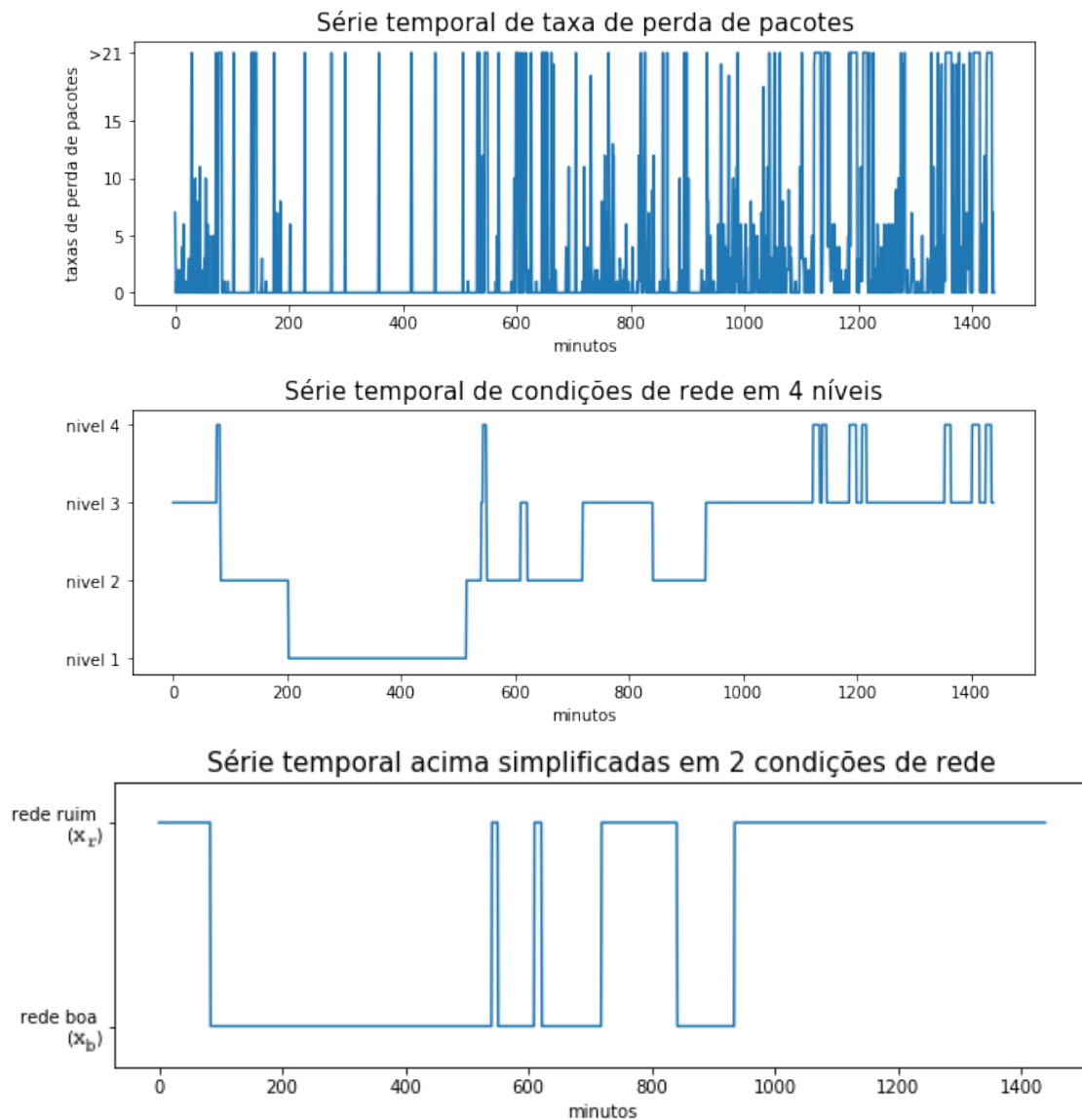


Figura 5.3: Exemplo de uma série temporal usada no experimento 1.

Após a segunda etapa do método, exemplificada acima, temos um novo *dataset* com séries temporais de condições de rede tais como a terceira série da figura 5.3.

A Figura 5.4 apresenta o histograma do número de eventos em cada série calculado para o *dataset* do experimento 1.

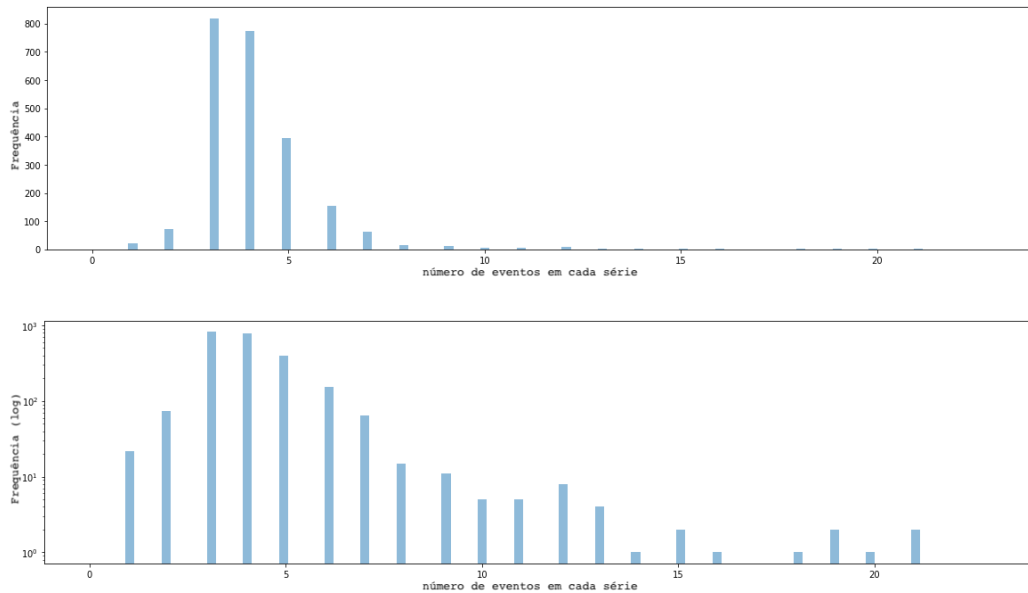


Figura 5.4: Histograma do número de eventos em cada série do *dataset* do experimento 1

Esses eventos, por sua vez, possuem durações de acordo com histograma da Figura 5.5.

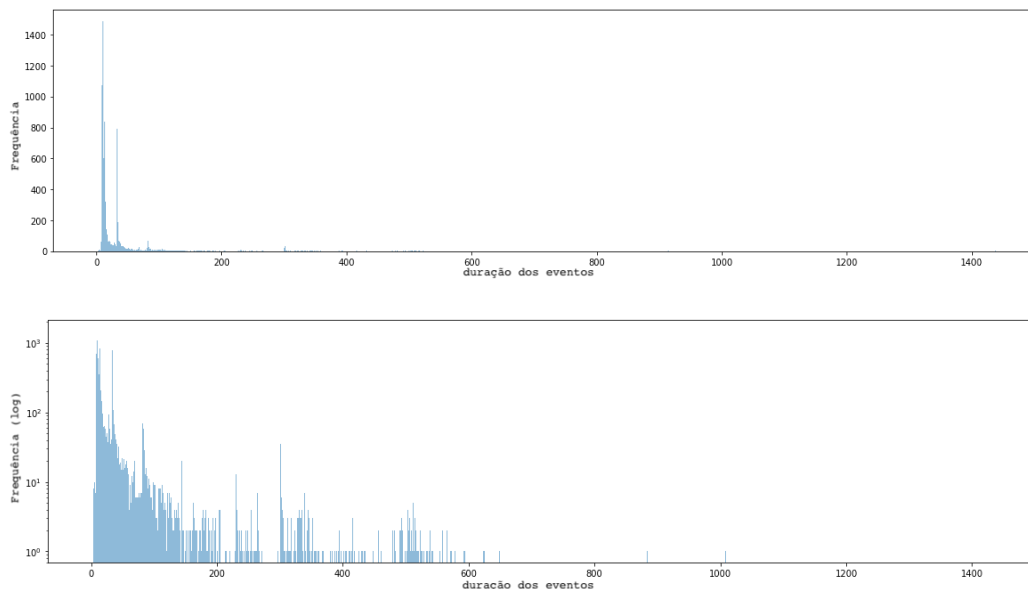


Figura 5.5: Histograma da duração dos eventos das séries do *dataset* do experimento 1

Foram separadas, aleatoriamente, 70% das séries deste *dataset* para treino do algoritmo *Value Iteration* (VI), de acordo com o que foi descrito na segunda etapa

do método do capítulo anterior. As 30% séries restantes foram reservadas para teste do alarme utilizando a política ótima resultante do treinamento com o algoritmo VI.

Então, ao fim da terceira etapa do método e considerando somente o *dataset* de treino, a política ótima encontrada a partir do algoritmo *Value Iteration* é a seguinte:

$$\begin{aligned}
 & \vdots \\
 s &= ( D, 0, (x_b, x_b, x_b, \dots, x_b, x_b, x_b, x_b, x_b) ) \rightarrow a = 0 \\
 s &= ( D, 0, (x_b, x_b, x_b, \dots, x_b, x_b, x_b, x_b, x_r) ) \rightarrow a = 0 \\
 s &= ( D, 0, (x_b, x_b, x_b, \dots, x_b, x_b, x_b, x_r, x_r) ) \rightarrow a = 0 \\
 s &= ( D, 0, (x_b, x_b, x_b, \dots, x_b, x_b, x_r, x_r, x_r) ) \rightarrow a = 0 \\
 s &= ( D, 0, (x_b, x_b, x_b, \dots, x_b, x_r, x_r, x_r, x_r) ) \rightarrow a = 0 \\
 & \vdots \\
 s &= ( D, 0, (x_b, x_r, x_r, \dots, x_r, x_r, x_r, x_r, x_r) ) \rightarrow a = 0 \\
 s &= ( D, 0, (x_r, x_r, x_r, \dots, x_r, x_r, x_r, x_r, x_r) ) \rightarrow a = 1 \\
 & \vdots
 \end{aligned}$$

O procedimento de teste é realizado começando com o alarme desligado em cada série do *dataset* de teste e decidindo sobre o alarme nos próximos instantes da série, segundo a política ótima resultante das séries treino. De acordo com as métricas de avaliação, ao final do procedimento de teste, cada série pode ser classificada como verdadeiro positivo (TP), falso positivo (FP), verdadeiro negativo (TN) ou falso negativo (FN). A Tabela 5.1 apresenta o número de séries em cada classificação.

Total de séries de teste	708
Verdadeiros positivos (TP)	565
Verdadeiros negativos (TN)	0
Falsos positivos (FP)	38
Falsos negativos (FN)	105

Tabela 5.1: Tabela ou matriz de confusão

A partir da Tabela 5.1, foram calculadas algumas medidas de desempenho do método. A Tabela 5.2 mostra os valores para acurácia, precisão, *recall* e *f1-score*.

Acurácia	0,79802
Precisão	0,93698
<i>Recall</i>	0,84328
<i>F1-score</i>	0,88767

Tabela 5.2: Medidas de desempenho do método

Portanto, como podemos observar na Tabela 5.2, o método apresentou um bom desempenho. O sistema aprendeu a política ótima para a decisão do alarme de manutenção, de acordo com as características do *dataset* e as informações de orçamento



diário do ISP.

## 5.2.2 Experimento 2

O *dataset* deste experimento é formado por 2689 séries de medidas de taxa de perda de pacotes. Assim como no experimento 1, essas séries são processadas na segunda etapa do método e geram um novo *dataset* de séries de condições de rede. A figura 5.6, a seguir, demonstra a série obtida após a segunda etapa do método.

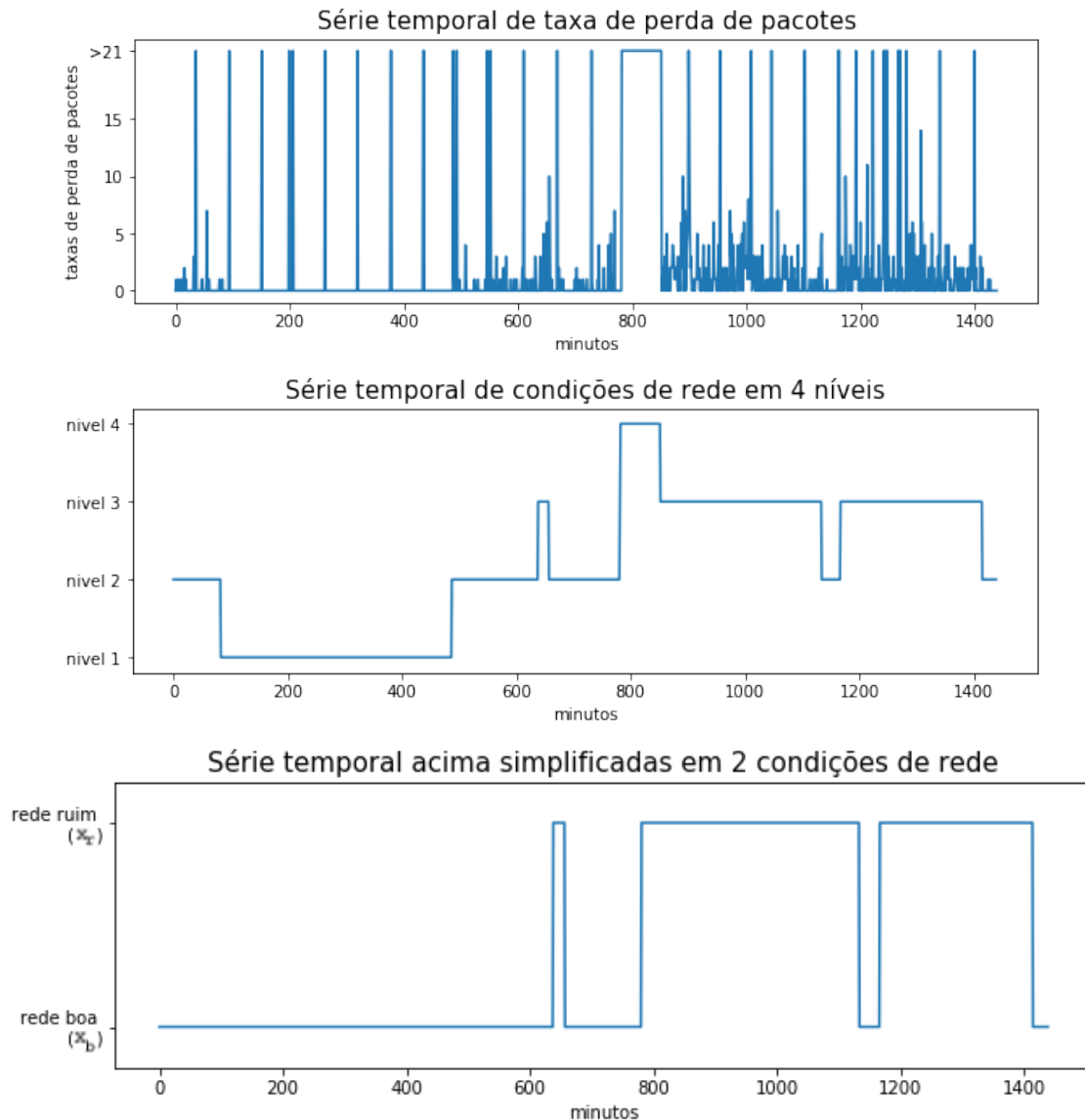


Figura 5.6: Exemplo de uma série temporal usada no experimento 2.

Ao final da segunda etapa, temos um novo *dataset* com séries temporais de condições de rede tais como a terceira série temporal exemplificada na Figura 5.6.

De acordo com a definição de “evento” numa série, descrito na primeira seção deste capítulo, a terceira série temporal na Figura 5.6 possui 3 eventos.

A Figura 5.7 apresenta o histograma do número de eventos em cada série temporal do *dataset* de condições de rede resultante da segunda etapa do método.

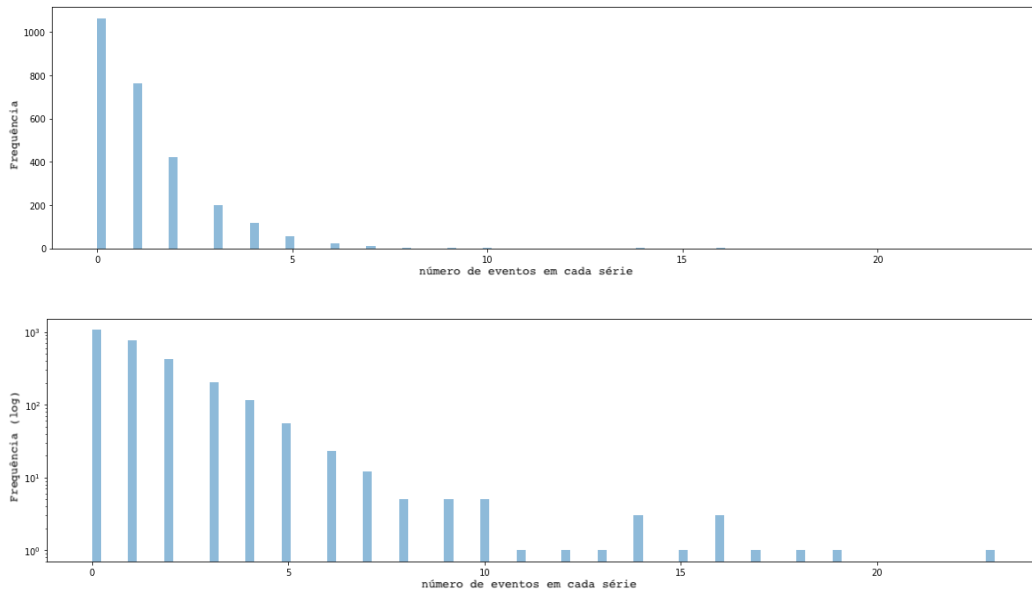


Figura 5.7: Histograma do número de eventos em cada série no *dataset* do experimento 2

Esses eventos têm durações de acordo com o histograma da Figura 5.8.

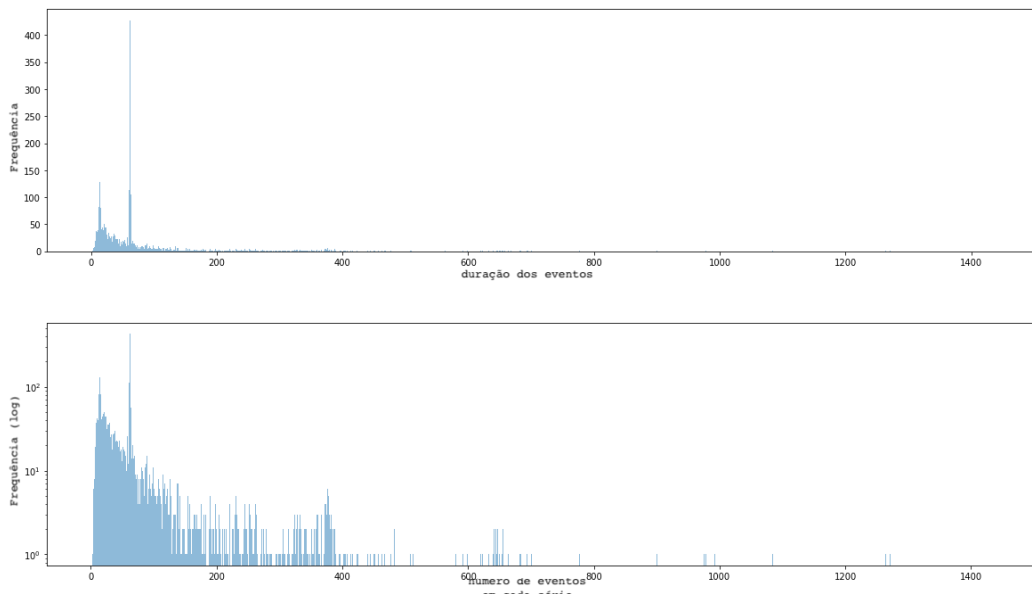


Figura 5.8: Histograma da duração dos eventos das séries do experimento 2.

Nesse *dataset* de séries de condições de rede, descrito acima, há uma divisão entre séries destinadas à realização do treinamento no algoritmo *Value Iteration* (VI) e séries reservadas para teste do alarme utilizando a política ótima resultante

do algoritmo VI. Essa divisão se deu aleatoriamente e da seguinte forma: 70% das séries deste *dataset* foi separado para treino e 30%, para o teste.

Logo, ao fim da terceira etapa do método e considerando somente o *dataset* de treino, a política ótima encontrada a partir do algoritmo *Value Iteration* é a seguinte:

$$\begin{aligned}
& \vdots \\
s &= ( D, 0, (x_b, x_b, x_b, \dots, x_b, x_b, x_b, x_b, x_b) ) \rightarrow a = 0 \\
s &= ( D, 0, (x_b, x_b, x_b, \dots, x_b, x_b, x_b, x_b, x_r) ) \rightarrow a = 1 \\
s &= ( D, 0, (x_b, x_b, x_b, \dots, x_b, x_b, x_b, x_r, x_r) ) \rightarrow a = 1 \\
s &= ( D, 0, (x_b, x_b, x_b, \dots, x_b, x_b, x_r, x_r, x_r) ) \rightarrow a = 1 \\
s &= ( D, 0, (x_b, x_b, x_b, \dots, x_b, x_r, x_r, x_r, x_r) ) \rightarrow a = 1 \\
& \vdots \\
s &= ( D, 0, (x_r, x_r, x_r, \dots, x_r, x_r, x_r, x_r, x_r) ) \rightarrow a = 1 \\
s &= ( D, 0, (x_r, x_r, x_r, \dots, x_r, x_r, x_r, x_r, x_r) ) \rightarrow a = 1 \\
& \vdots
\end{aligned}$$

O procedimento de teste é realizado começando com o alarme desligado em cada série de teste, ou seja,  $l^{(0)} = D$ , e decidindo sobre o alarme nos próximos instantes, segundo a política ótima resultante das séries treino.

Analisando o comportamento do alarme em todas as séries, elas podem ser classificadas como verdadeiro positivo (TP), verdadeiro negativo (TN), falso positivo (FP) ou falso negativo (FN), de acordo com as métricas de avaliação. A Tabela 5.3 ilustra o número de séries em cada classificação:

Total de séries de teste	807
Verdadeiros positivos (TP)	413
Verdadeiros negativos (TN)	324
Falsos positivos (FP)	70
Falsos negativos (FN)	0

Tabela 5.3: Tabela ou matriz de confusão

De acordo com os valores da Tabela 5.3, é possível calcular algumas medidas de desempenho do método. Os valores de acurácia, precisão, *recall* e *f1-score* são apresentados na Tabela 5.4.

Portanto, como podemos observar na Tabela 5.4, o método apresentou um bom desempenho. O sistema aprendeu a política ótima para a decisão do alarme de manutenção, de acordo com as características do *dataset*.

Acurácia	0,91326
Precisão	0,85507
<i>Recall</i>	0,99999
<i>F1-score</i>	0,92188

Tabela 5.4: Medidas de desempenho do método

### 5.3 Análise de Resultados

Apesar do objetivo do método ser o acionamento do alarme nos eventos da série com maiores durações, há duas limitações relacionadas aos parâmetros escolhidos para as variáveis de estado. A primeira limitação é que o número de alarmes por série não deve ultrapassar o limite diário estipulado ( $n \leq n_{max}$ ).

A segunda limitação dos parâmetros escolhidos é que eventos com durações maiores do que  $w$  serão considerados iguais a eventos com duração  $w$ , pois eles representarão, a cada passo, o mesmo vetor  $\tilde{v}$  e, conseqüentemente, o mesmo estado  $s$  para as decisões das ações. Sendo assim, tendo os parâmetros  $n_{max} = 1$  e  $w = 15$  para ambos os experimentos, podemos considerá-los nas seguintes análises.

O resultado de política ótima do experimento 1 se dá pelo fato de que geralmente há, no seu *dataset* de condições de rede, mais de um evento por série e muitos dos eventos são de duração curta, como podemos constatar na Figura 5.4 e na Figura 5.5. Então, neste caso, o método aprende a ser mais conservador em manter o alarme desligado até ele encontrar o primeiro vetor  $\tilde{v}^{(t)} = (x_r, \dots, x_r, x_r, x_r)$  da série. Desta forma, o sistema tem mais garantias de que, no instante  $t$ , há um evento longo, ou seja, maior que  $w$ .

No experimento 2, o resultado de política ótima decide por acionar o alarme ao encontrar o primeiro vetor  $\tilde{v}^{(t)} = (x_b, \dots, x_b, x_b, x_r)$  na série. A primeira explicação deste resultado é que, no *dataset* de condições de rede, geralmente há apenas um evento por série, como se pode conferir na Figura 5.7. Então, nessas séries com apenas um evento, o método deve acionar o alarme nos eventos únicos, se importando menos com a duração deles.

No histograma de durações de série do experimento 2 (Figura 5.8), também percebemos que a maioria dos eventos tem duração maior que  $w$ ; então, em séries que possuam mais de um evento de duração maior que  $w$ , o método deve acionar o alarme logo no primeiro evento encontrado.

Portanto, em ambos os exemplos de séries temporais citadas acima para o experimento 2, o método aprende a ser mais otimista e liga o alarme ao detectar o primeiro evento da série.

É possível conferir, através das Tabelas 5.2 e 5.4, que houve uma certa diferença entre as medidas de desempenho do método, correspondentes a cada experimento. Há alguns motivos para a obtenção dessa diferença. As características distintas de

cada *dataset* geram possibilidades também distintas de classificação das séries de teste após o método (TP,TN,FP ou FN).

No experimento 1, por exemplo, não há verdadeiros negativos (TN) de acordo com a Tabela 5.2, ou seja, não há nenhuma constatação de uma série com zero eventos em que o método não acione o alarme. Isso se deve ao fato de não haver séries com zero eventos (Figura 5.4) no *dataset* do experimento 1. Essa mesma situação não ocorre no experimento 2.

Outro exemplo de características de *dataset* interferindo nas medidas de desempenho, é o caso do experimento 2 apresentar zero falsos negativos (FN) de acordo com a Tabela 5.4. As características do *dataset* do experimento 2 geram uma política ótima que permite o acionamento do alarme ao encontrar uma condição ruim de rede ( $x_r$ ) pela primeira vez na série. Isso impossibilita a classificação de uma série em falso negativo (FN), pois sempre que uma série possuir um ou mais eventos, ela vai receber pelo menos um acionamento do alarme. Isso não ocorre com o resultado de política ótima do experimento 1.

Apesar dessas diferenças ocasionadas pelas características de cada *dataset*, o método, de modo geral, apresenta bom desempenho.

# Capítulo 6

## Conclusão

Este trabalho apresenta uma metodologia para otimização da decisão de quando realizar manutenção numa rede de um ISP. Ele se baseia na ideia de que um alarme de manutenção é acionado quando há uma mudança negativa no padrão de desempenho da rede, ou seja, as medidas de desempenho da rede passam de boas para ruins, e quando há a previsão de que essa mudança é duradoura.

O método possui três etapas. A primeira delas é a coleta de dados de desempenho da rede. Ela consiste da coleta periódica de medidas nos roteadores domésticos da rede do ISP. Ao fim desta etapa, temos um *dataset* com várias séries temporais de medidas de desempenho de rede.

Logo após seguimos para a etapa de modelagem das condições de rede. Através de uma análise estatística, define-se alguns padrões de condições de rede como estados ocultos de um HMM. Cada estado oculto possui uma distribuição de probabilidade em emitir observações que, neste caso, se tratam de possíveis valores de medições de desempenho da rede. Com o auxílio do algoritmo de *Viterbi*, as séries temporais de medidas de desempenho da rede se transformam em séries temporais de condições da rede. A finalidade dessa etapa é filtrar mudanças bruscas nas séries temporais de medidas de desempenho.

Na última etapa, que é a principal contribuição do trabalho, é usada uma técnica de *Reinforcement Learning* nas séries temporais de estados ocultos, para inferir o nível de qualidade da rede em instantes futuros e indicar o acionamento de um alarme de manutenção. Com isso, o gerenciador da rede pode otimizar as ações de manutenção preditiva na rede ou mudanças de modo de operação.

Esta metodologia apresentou bons resultados em testes simulando medidas de rede e obteve altas taxas de precisão, acurácia, *recall* e *f1-score*, em testes utilizando medidas reais de desempenho de rede. Isto significa que o algoritmo acionou o alarme de manutenção, na maioria dos casos, durante os pontos mais críticos da série que são os momentos com picos mais longos de condição ruim de rede ao longo de cada série.

Uma possível extensão desse trabalho seria o uso de outras medidas de desempenho da rede, como latência e vazão ou mais de uma dessas medidas ao mesmo tempo. A única medida de desempenho utilizada neste trabalho foi a taxa de perda de pacotes.

Poderíamos também adicionar a informação da topologia da rede do ISP às inferências de QoS da rede, de modo a correlacionar as séries de usuários próximos geograficamente e referentes ao mesmo dia.

Outra ideia para extensão do método seria a definição de múltiplos alarmes baseados no par condição de rede e aplicação, dado que as aplicações possuem diferentes requisitos de desempenho. Por exemplo, poderia ser definido o alarme  $A_1$  para a aplicação *streaming* de vídeo e condição de rede  $r_1$  e outro alarme  $A_2$  para a aplicação navegação na *web* e condição de rede  $r_2$ , onde  $r_2$  é uma condição de rede pior que  $r_1$ .

Com este trabalho, podemos otimizar recursos de manutenção em redes, algo muito importante principalmente para empresas de distribuição de serviço de Internet (ISP) de pequeno porte. Além de minimizar custos, a previsão contribui para uma melhor experiência para os usuários. Manutenções desnecessárias são tão indesejadas quanto longos problemas sem resolução, logo, este trabalho se assemelha a outros de *Reinforcement Learning* que buscam um equilíbrio para o sistema.

# Referências Bibliográficas

- [1] ASSESSORIA-TÉCNICA-ANATEL. “Relatório de acompanhamento do setor de telecomunicações – 3<sup>o</sup> Trimestre 2019”, *Anatel*, 2019. Disponível em: <<https://anatel.gov.br/dados/relatorios-de-acompanhamento/2019#R26>>. Acessado em: 03 fev. 2020.
- [2] ADEYEMI-EJEYE, A. O., ALRESHOODI, M., AL-JOBOURI, L., et al. “Impact of packet loss on 4K UHD video for portable devices”, *Multimedia Tools and Applications*, v. 78, n. 22, pp. 31733–31755, 2019. ISSN: 1573-7721. doi: 10.1007/s11042-019-07996-1. Disponível em: <<https://doi.org/10.1007/s11042-019-07996-1>>.
- [3] BI, Y., HAN, G., LIN, C., et al. “Effective Packet Loss Elimination in IP Mobility Support for Vehicular Networks”, *IEEE Network*, v. 34, n. 1, pp. 152–158, January 2020. ISSN: 1558-156X. doi: 10.1109/MNET.2019.1900093.
- [4] HUNG, F. H., WU, C. K., ZOU, Z., et al. “Packet error rate analysis in IoT for industrial air conditioning system”. In: *IECON 2017 - 43rd Annual Conference of the IEEE Industrial Electronics Society*, pp. 8367–8370, Oct 2017. doi: 10.1109/IECON.2017.8217469.
- [5] LIU, B., BRASS, P., DOUSSE, O., et al. “Mobility Improves Coverage of Sensor Networks”. In: *Proceedings of the 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing, MobiHoc '05*, p. 300–308, New York, NY, USA, 2005. Association for Computing Machinery. ISBN: 1595930043. doi: 10.1145/1062689.1062728. Disponível em: <<https://doi.org/10.1145/1062689.1062728>>.
- [6] SILVEIRA, F., DE SOUZA E SILVA, E. “Predicting Packet Loss Statistics with Hidden Markov Models”, *SIGMETRICS Perform. Eval. Rev.*, v. 35, n. 3, pp. 19–21, dez. 2007. ISSN: 0163-5999. doi: 10.1145/1328690.1328698. Disponível em: <<https://doi.org/10.1145/1328690.1328698>>.
- [7] SANTOS, G., MENDONÇA, G., DE SOUZA E SILVA, E., et al. “Análise não supervisionada para inferência de qualidade de experiência de usuá-



- rios residenciais”. In: *Anais do XXXVII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pp. 958–971, Porto Alegre, RS, Brasil, 2019. SBC. doi: 10.5753/sbrc.2019.7415. Disponível em: <<https://sol.sbc.org.br/index.php/sbrc/article/view/7415>>.
- [8] XIMENES, D., MENDONÇA, G., SANTOS, G. H. A., et al. “O Problema de Detecção e Localização de Eventos em Séries Temporais Aplicado a Redes de Computadores”. In: *17<sup>o</sup> Workshop em Desempenho de Sistemas Computacionais e de Comunicação (WPerformance 2018)*, v. 17, Porto Alegre, RS, Brasil, 2018. SBC. Disponível em: <<http://portaldeconteudo.sbc.org.br/index.php/wperformance/article/view/3323>>.
- [9] MAIDSTONE, R., HOCKING, T., RIGAILL, G., et al. “On optimal multiple changepoint algorithms for large data”, *Statistics and Computing*, v. 27, n. 2, pp. 519–533, Mar 2017. ISSN: 1573-1375. doi: 10.1007/s11222-016-9636-3. Disponível em: <<https://doi.org/10.1007/s11222-016-9636-3>>.
- [10] AMINIKHANGHAHI, S., COOK, D. J. “A survey of methods for time series change point detection”, *Knowledge and Information Systems*, v. 51, n. 2, pp. 339–367, May 2017. ISSN: 0219-3116. doi: 10.1007/s10115-016-0987-z. Disponível em: <<https://doi.org/10.1007/s10115-016-0987-z>>.
- [11] WANG, H., LI, J., YU, Q., et al. “Integrating recurrent neural networks and reinforcement learning for dynamic service composition”, *Future Generation Computer Systems*, v. 107, pp. 551 – 563, 2020. ISSN: 0167-739X. doi: <https://doi.org/10.1016/j.future.2020.02.030>. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0167739X19311240>>.
- [12] WANG, H., ZHOU, X., ZHOU, X., et al. “Adaptive Service Composition Based on Reinforcement Learning”. In: Maglio, P. P., Weske, M., Yang, J., et al. (Eds.), *Service-Oriented Computing*, pp. 92–107, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. ISBN: 978-3-642-17358-5.
- [13] AL-JAWAD, A., SHAH, P., GEMIKONAKLI, O., et al. “LearnQoS: A Learning Approach for Optimizing QoS Over Multimedia-Based SDNs”. In: *2018 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, pp. 1–6, 2018.

- [14] KIM, H., FEAMSTER, N. “Improving network management with software defined networking”, *IEEE Communications Magazine*, v. 51, n. 2, pp. 114–119, 2013.
- [15] RABINER, L., JUANG, B. “An introduction to hidden Markov models”, *IEEE ASSP Magazine*, v. 3, n. 1, pp. 4–16, Jan 1986. ISSN: 0740-7467. doi: 10.1109/MASSP.1986.1165342.
- [16] NORRIS, J. R. *Markov Chains*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge, Cambridgeshire, Reino Unido, Cambridge University Press, 1997. doi: 10.1017/CBO9780511810633.
- [17] RABINER, L. R. “A tutorial on hidden Markov models and selected applications in speech recognition”, *Proceedings of the IEEE*, v. 77, n. 2, pp. 257–286, 1989.
- [18] VITERBI, A. “Error bounds for convolutional codes and an asymptotically optimum decoding algorithm”, *IEEE Transactions on Information Theory*, v. 13, n. 2, pp. 260–269, April 1967. ISSN: 1557-9654. doi: 10.1109/TIT.1967.1054010.
- [19] SCHRITTWIESER, J., ANTONOGLU, I., HUBERT, T., et al. “Mastering Atari, Go, chess and shogi by planning with a learned model”, *Nature*, v. 588, n. 7839, pp. 604–609, Dec 2020. ISSN: 1476-4687. doi: 10.1038/s41586-020-03051-4. Disponível em: <<https://doi.org/10.1038/s41586-020-03051-4>>.
- [20] SILVER, D., HUBERT, T., SCHRITTWIESER, J., et al. “A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play”, *Science*, v. 362, n. 6419, pp. 1140–1144, 2018.
- [21] MNIH, V., KAVUKCUOGLU, K., SILVER, D., et al. “Playing atari with deep reinforcement learning”, *arXiv preprint arXiv:1312.5602*, 2013.
- [22] LIU, J., YANG, J., LIU, H., et al. “An improved ant colony algorithm for robot path planning”, *Soft Computing*, v. 21, n. 19, pp. 5829–5839, Oct 2017. ISSN: 1433-7479. doi: 10.1007/s00500-016-2161-7. Disponível em: <<https://doi.org/10.1007/s00500-016-2161-7>>.
- [23] KOLLAR, T., ROY, N. “Trajectory Optimization using Reinforcement Learning for Map Exploration”, *The International Journal of Robotics Research*, v. 27, n. 2, pp. 175–196, 2008. doi: 10.1177/0278364907087426. Disponível em: <<https://doi.org/10.1177/0278364907087426>>.

- [24] POLYDOROS, A. S., NALPANTIDIS, L. “Survey of Model-Based Reinforcement Learning: Applications on Robotics”, *Journal of Intelligent & Robotic Systems*, v. 86, n. 2, pp. 153–173, May 2017. ISSN: 1573-0409. doi: 10.1007/s10846-017-0468-y. Disponível em: <<https://doi.org/10.1007/s10846-017-0468-y>>.
- [25] KEMMER, L., VON KLEIST, H., DE ROCHEBOUËT, D., et al. “Reinforcement learning for supply chain optimization”, *European Workshop on Reinforcement Learning*, v. 14, n. 10, 2018.
- [26] PEIDRO, D., MULA, J., POLER, R., et al. “Quantitative models for supply chain planning under uncertainty: a review”, *The International Journal of Advanced Manufacturing Technology*, v. 43, n. 3, pp. 400–420, Jul 2009. ISSN: 1433-3015. doi: 10.1007/s00170-008-1715-y. Disponível em: <<https://doi.org/10.1007/s00170-008-1715-y>>.
- [27] SUTTON, R. S., BARTO, A. G. *Introduction to Reinforcement Learning*. 1st ed. Cambridge, MA, USA, MIT Press, 1998. ISBN: 0262193981.
- [28] SUTTON, R. S., BARTO, A. G. *Reinforcement learning: An introduction*. Cambridge, MA, USA, MIT press, 2018.
- [29] OTTERLO, M., WIERING, M. “Reinforcement Learning and Markov Decision Processes”, *Reinforcement Learning: State of the Art*, pp. 3–42, 01 2012. doi: 10.1007/978-3-642-27645-3\_1.
- [30] SUNDARESAN, S., DE DONATO, W., FEAMSTER, N., et al. “Broadband Internet Performance: A View from the Gateway”. In: *Proceedings of the ACM SIGCOMM 2011 Conference*, SIGCOMM ’11, p. 134–145, New York, NY, USA, 2011. Association for Computing Machinery. ISBN: 9781450307970. doi: 10.1145/2018436.2018452. Disponível em: <<https://doi.org/10.1145/2018436.2018452>>.
- [31] DA SILVA, A. P. C., VARELA, M., E SILVA, E. D. S., et al. “Quality assessment of interactive voice applications”, *Computer Networks*, v. 52, n. 6, pp. 1179–1192, 2008.
- [32] CHARONYKTAKIS, P., PLAKIA, M., TSAMARDINOS, I., et al. “On User-Centric Modular QoE Prediction for VoIP Based on Machine-Learning Algorithms”, *IEEE Transactions on Mobile Computing*, v. 15, n. 6, pp. 1443–1456, 2016.