UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE TECNOLOGIA E GEOCIÊNCIAS
DEPARTAMENTO DE ELETRÔNICA E SISTEMAS
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

HIGOR ÍTALO DOS SANTOS

ALGORITMOS NÃO SUPERVISIONADOS PARA A CONSTRUÇÃO DE AUTÔMATOS PROBABILÍSTICOS A PARTIR DE SÉRIES TEMPORAIS

Recife

2024

HIGOR ÍTALO DOS SANTOS

ALGORITMOS NÃO SUPERVISIONADOS PARA A CONSTRUÇÃO DE AUTÔMATOS PROBABILÍSTICOS A PARTIR DE SÉRIES TEMPORAIS

Tese apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Pernambuco como requisito parcial para obtenção do título de Doutor em Engenharia Elétrica.

Área de Concentração: Comunicações.

Orientador: Prof. Dr. Daniel Pedro Bezerra Chaves

Coorientador: Prof. Dr. Cecilio José Lins Pimentel

Recife

2024

.Catalogação de Publicação na Fonte. UFPE - Biblioteca Central

Santos, Higor Ítalo dos.
Algoritmos não supervisionados para a construção de autômatos probabilísticos a partir de séries temporais / Higor Ítalo dos Santos. - Recife, 2024.
88f.: il.

Tese (doutorado) - Universidade Federal de Pernambuco, Centro de Tecnologia e Geociências, Programa de Pós-Graduação em Engenharia Elétrica, 2024.
Orientação: Daniel Pedro Bezerra Chaves.
Coorientação: Cecilio José Lins Pimentel.
Inclui referências.

1. Engenharia Elétrica; 2. Autômatos probabilísticos de estados finitos; 3. Aprendizagem de máquina; 4. Minimização de grafos; 5. Modelagem não supervisionada; 6. Detecção de falhas. I. Chaves, Daniel Pedro Bezerra. II. Pimentel, Cecilio José Lins. III. Título.

UFPE-Biblioteca Central

HITOR ÍTALO DOS SANTOS

“ALGORITMOS NÃO SUPERVISIONADOS PARA A CONSTRUÇÃO DE AUTÔMATOS PROBABILÍSTICOS A PARTIR DE SÉRIES TEMPORAIS”

Tese apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Doutor em Engenharia Elétrica, na área de concentração em Comunicações.

Aprovada em: 11/09/2024.

BANCA EXAMINADORA

_____

Prof. Dr. Daniel Pedro Bezerra Chaves
(Orientador e Examinador Interno)
Universidade Federal de Pernambuco

_____

Prof. Dr. Juliano Bandeira Lima
(Examinador Interno)
Universidade Federal de Pernambuco

_____

Prof. Dr. Ricardo Menezes Campello de Souza
(Examinador Externo)
Universidade de Pernambuco

_____

Prof. Dr. Renato Mariz de Moraes
(Examinador Externo)
Universidade Federal de Pernambuco

_____

Prof. Dr. Vítor de Andrade Coutinho
(Examinador Externo)
Universidade Federal Rural de Pernambuco

A todos que de alguma forma contribuíram para que eu alcançasse horizontes inimagináveis e, a partir disso, ir mais além.

AGRADECIMENTOS

Primeiramente gostaria de agradecer a Deus pelo suporte espiritual, o qual me fornece ânimo e suporte em todos momentos de vida.

À Gabriela Chagas, maior incentivadora de minha pós-graduação, pelo apoio, paciência, compreensão e amor dedicados ao longo de anos de vida pessoal e acadêmica.

Aos meus pais, Inácia e Adilson, por terem buscado garantir a melhor educação possível para mim, mesmo em meio a tantas dificuldades, me permitindo alcançar o elevado nível em que me encontro neste momento.

À minha irmã Amanda, demais familiares e amigos pelo suporte material e sentimental proporcionados ao longo de minha vida, sem os quais não teria alcançado diversos objetivos.

Ao meu orientador Prof. Dr. Daniel Chaves e meu coorientador Prof. Dr. Cecilio Pimentel, por todos os anos dedicados a esta pesquisa, pela atenção, paciência, disponibilidade e tantos ensinamentos que me permitiram realizar este trabalho.

Aos colegas Andy e Robson, pela companhia e companheirismo nos anos finais do doutorado, tornando nossa rotina no ambiente de pesquisa mais agradável, compartilhando ideias, diálogos e um bom café.

Aos colegas de jornada pós-acadêmica: Diego Hamilton, Helder, Túlio, Antônio, Carlos, Matheus e tantos outros que acompanharam minha trajetória e de alguma forma contribuíram para meu progresso acadêmico.

A todos os professores que passaram por minha vida e que contribuíram com paciência, dedicação e ensinamentos para a minha formação.

A todos que desenvolvem pesquisas nos mais diversos campos da ciência, pela persistência em acreditar na sua importância, apesar de tão desvalorizada e até menosprezadas nos tempos atuais, sendo portanto os agentes responsáveis por levar o desenvolvimento tecnológico e intelectual da humanidade adiante.

À Universidade Federal de Pernambuco, por fornecer uma estrutura física e formação pós-acadêmica de qualidade.

A Fundação Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (Capes) pelo suporte financeiro recebido durante o tempo de realização deste trabalho.

RESUMO

Uma das metodologias de análise e modelagem de sistemas dinâmicos comumente estudadas envolve processos de simbolização de séries temporais associados a sistemas dinâmicos. Este método apresenta, dentre as principais vantagens, uma redução da complexidade dos dados ao capturar a dinâmica fundamental do sistema na forma de uma sequência simbólica, além de favorecer robustez a fatores externos, como ruídos. Uma técnica tradicionalmente adotada para análise destas sequências simbólicas consiste no emprego de Autômatos Probabilísticos de Estados Finitos (PFSA, *Probabilistic Finite State Automata*), uma abordagem estruturada capaz de representar a dinâmica discreta da sequência simbólica. Há diversas aplicações possíveis para esses PFSAs, dentre as quais a composição de algoritmos de controle e técnicas de detecção de falha. Este trabalho propõe dois novos algoritmos para determinação de PFSAs relacionados a um sistema dinâmico a partir de uma realização suficientemente longa de uma série simbólica. Empregando-se aprendizado de máquina e minimização de grafos, obtém-se PFSAs reduzidos e fiéis à dinâmica discreta proveniente do sistema original. Para validar os algoritmos propostos, sequências simbólicas associadas a sistemas dinâmicos simulados e experimentais são modeladas. Demonstra-se que os algoritmos propostos geram PFSAs com menor número de estados que outras propostas da literatura, sem comprometer os quantificadores de aderência entre as sequências modeladas e aquelas geradas pelos PFSAs. Adicionalmente, é explorado o uso do vetor de ocupação de estados de um PFSA como dados de treinamento para redes neurais, visando classificar diversos tipos de falhas em uma máquina elétrica rotativa, demonstrando a eficácia e aplicabilidade dessa abordagem nesse tipo de tarefa.

Palavras-chave: Aprendizagem de máquina. Autômatos probabilísticos de estados finitos. Minimização de grafos. Modelagem não supervisionada. Sistemas dinâmicos.

ABSTRACT

One of the commonly studied methodologies for the analysis and modeling of dynamic systems involves the symbolization processes of time series associated with dynamic systems. This method presents, among its main advantages, a reduction in data complexity by capturing the fundamental dynamics of the system in the form of a symbolic sequence, as well as increased robustness to external factors such as noise. A traditionally adopted technique for the analysis of these symbolic sequences consists of the use of Probabilistic Finite State Automata (PFSA), a structured approach capable of representing the discrete dynamics of the symbolic sequence. There are various possible applications for these PFSAs, including the composition of control algorithms and fault detection techniques. This work proposes two new algorithms for determining PFSAs related to a dynamic system from a sufficiently long realization of a symbolic series. By employing machine learning and graph minimization, reduced PFSAs faithful to the discrete dynamics of the original system are obtained. To validate the proposed algorithms, symbolic sequences associated with simulated and experimental dynamic systems are modeled. It is demonstrated that the proposed algorithms generate PFSAs with fewer states than other proposals in the literature, without compromising the quantifiers of adherence between the modeled sequences and those generated by the PFSAs. Additionally, the use of the state occupancy vector of a PFSA as training data for neural networks is explored, aiming to classify various types of faults in a rotating electrical machine, demonstrating the efficacy and applicability of this approach in such tasks.

Keywords: Machine learning. Probabilistic finite state automata. Graph minimization. Unsupervised modeling. Dynamical systems.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AUC	Area Under Curve
CT	Context Tree
DCGraM	D-Markov with Clustering and Graph Minimization
FPR	False Positive Rate
HMM	Hidden Markov Model
MaFaulDa	Machinery Fault Database
MFS	Machinery Fault Simulator
NDPFSA	Non-deterministic PFSA
PFSA	Probabilistic Finite State Automata
PS2ER	Preserved State-Splitting with Entropy Reduction
ROC	Receiver Operating Characteristic Curve
SDF	Symbolic Dynamic Filtering
S2ER	State-Splitting with Entropy Reduction
SVM	Support Vector Machine
TPR	True Positive Rate
VL-DCGraM	Variable Length D-Markov with Clustering and Graph Minimization
VLMC	Variable Length Markov Chain
WDCNN	Deep Convolutional Neural Networks with Wide First-layer Kernels

LISTA DE SÍMBOLOS

D	Profundidade de uma máquina D-Markov
S	Sequência simbólica gerada por um PFSA
ϵ	Sequência de comprimento nulo
L	Profundidade de uma árvore de contextos $\mathcal{CT}$
Σ	Alfabeto finito. Conjunto discreto de símbolos
Σ^n	Conjunto de sequências de símbolos de Σ de comprimento n
Σ^*	Conjunto de todas as sequências com símbolos de Σ de qualquer comprimento
σ	Símbolo pertencente ao alfabeto Σ
$\sum$	Operador matemático de somatório
G	Grafo discreto rotulado
$F(q)$	Contexto à direita de uma estado $q \in Q$ de uma grafo G
Q	Conjunto de estados de uma grafo G
q	Estado pertencente a conjunto de estados Q de um grafo G
δ	Função de transição de um grafo
$\bigcup$	Operador união de conjuntos
$\mathcal{P}$	Partição de um conjunto de estados Q
$\wedge$	Operador de interseção entre partições
$\mathcal{L}$	Linguagem gerada por um grafo
$\mathcal{M}_h$	Partição definida pela equivalência Moore de profundidade h
π	Função probabilidade de transição de estados de um PFSA, $\pi : Q \times \Sigma \rightarrow [0, 1]$
$\mathbf{P}$	Matriz de transição de probabilidade $Q \times Q$ de um PFSA
$\boldsymbol{\mu}$	Vetor estacionário da matriz de transição de probabilidade $\mathbf{P}$
$\mathbf{x}_n$	Ponto de dado associado a um *cluster*

k	Número de *clusters* do Algoritmo k-*means*
$\mathbf{m}^{(k)}$	Conjunto de vetores I-dimensionais correspondente aos centroides de um *cluster*
J	Função objetiva que relaciona a distância entre pontos de dados $\mathbf{x}_n$ e um centroide $\mathbf{m}^{(k)}$
r_{nk}	Variável de indicadores binários da função objetiva J
c	Número de *clusters* do algoritmo *fuzzy* c-*means*
$F_i(x_k)$	Grau de pertinência de um ponto de dado x_k a um *cluster* C_i
v_i	Centroide de um *cluster* C_i gerado via clusterização *fuzzy* c-*means*
$J_m(P)$	Índice de desempenho que relaciona a distância entre pontos de dado x_n e um centroide v_i
P_ω	Probabilidade associada a palavra ω que rotula um nó de uma estrutura em árvore
$P^{(t)}$	Partição de um conjunto de estados na clusterização *fuzzy* c-*means*
(G, π)	PFSA com grafo G e função probabilidade de transição π
$\mathcal{V}(q)$	*Morph* de um estado q de um PFSA (G, π), $\{\pi(q, \sigma), \forall \sigma \in \Sigma\}$
$\mathcal{V}(Q)$	Conjunto dos *morphs* de todos os estados $q \in Q$ de um PFSA (G, π)
CT	Árvore de contextos
$\mathcal{F}$	Conjunto de folhas de uma árvore de contextos CT
J_{pond}	Função objetiva que relaciona a distância entre a média ponderada de pontos de dado $\mathbf{x}_n$ e um centroide $\mathbf{m}^{(k)}$
l_{pert}	Limiar de pertinência para associação de um ponto de dado x_n a um *cluster* C_i
$f_i(x_k)$	Grau de pertinência de um ponto de dado x_n a um *subcluster* C_τ gerado via metodologia G-Moore do algoritmo NDPFSA
M_{pert}	Matriz de pertinências de pontos de dado x_n a *clusters* C_i
CD_σ	Conjunto de destino formado pelo rótulo de *clusters* que contém estados referentes a transição com símbolo σ a partir de um estado q
$M_{q_{\text{ND}}}$	Matriz de determinação de *morphs* de estados do modelo NDPFSA

q_{ND}	Estado do modelo NDPFSA
h	Taxa de entropia de um sistema dinâmico
h_ℓ	Taxa de entropia de ℓ-ésia ordem de um sistema dinâmico
$D_\ell(S_1 \| \| S_2)$	Divergência de Kullback-Leibler de ℓ-ésia ordem entre sequências S_1 e S_2
T	*Threshold* de ramificação de uma árvore de contextos CT
α	Valor real positivo adotado no cálculo da constante l_μ do algoritmo VL-DCGraM
M	Comprimento de palavra máximo adotado no algoritmo VL-DCGraM

SUMÁRIO

1 INTRODUÇÃO

Neste capítulo é feita uma introdução do presente trabalho, apresentando-se, na Seção 1.1, as motivações que servem de base para sua elaboração. Na Seção 1.2 é abordado o objetivo da pesquisa, com a descrição das etapas seguidas para o desenvolvimento da mesma. Na Seção 1.3 é apresentada a estrutura organizacional da tese.

1.1 MOTIVAÇÃO

A crescente complexidade e os custos de sistemas de engenharia nos últimos tempos tem demandado uma alta prioridade para características como segurança e confiabilidade em cenários adversos, como falhas de processos e anomalias de componentes. Como exemplos de tais sistemas, podemos citar redes e máquinas elétricas, equipamentos eletrônicos industriais, entre outros. Dada a importância desses sistemas, o desenvolvimento de métodos que viabilizem a detecção e identificação de possíveis causas de mau funcionamento, por vezes referenciadas na literatura como *anomalias* do sistema dinâmico associado, torna-se imprescindível para reduzir efeitos nocivos como: degradação do desempenho, diminuição da produtividade e riscos à segurança.

Técnicas que envolvem detecção de anomalias podem ser implementadas de diversas maneiras. Entre a variedade de abordagens existentes, duas categorias merecem destaque: métodos baseados em modelos e métodos baseados em dados. A primeira depende da disponibilidade do modelo do processo ou sistema considerado (GAO; CECATI; DING, 2015), (DING, 2008), obtido através de princípios físicos ou técnicas de identificação de sistema. Apesar de ter como vantagem a interpretação física, sua confiabilidade e eficiência computacional tendem a diminuir com o aumento da complexidade do sistema. Estes são usados em diversas aplicações, dentre as quais podemos citar um sistema de gerenciamento de bateria embarcado em tempo real (KIM et al., 2018) e um método para detecção *online* de anomalias na qualidade de energia em sistemas elétricos de potência (SAXENA et al., 2020).

O método baseado em dados tem se apresentado como um foco de pesquisa nos últimos anos (CHEN et al., 2017) e trata do problema de representação dos dados em um espaço reduzido, ou mais conveniente, mantendo um compromisso de confiabilidade e eficiência computacional a fim de rastrear a relação entrada-saída de sinais obtidos da leitura realizada por sensores confiáveis. Apresentando um amplo escopo de aplicações, pode ser caracterizado na maioria das vezes por viabilizar o aprendizado de padrões de comportamento dinâmico que gera os dados de interesse. Para tanto, são comumente aplicadas técnicas sob uma estrutura estatística, como análise de componentes principais (DUNTEMAN, 1989), mínimos quadrados parciais (BARKER; RAYENS, 2003), classificadores de padrões estatísticos e máquina de vetores de

suporte (LORENA; CARVALHO, 2007), entre outras. O problema a ser superado por essas técnicas é a necessidade de grande quantidade de dados de treinamento para captura das principais características do processo, fato atestado ao se lidar com séries temporais, onde o alto volume de dados acarreta em uma complexidade computacional equiparável. Apesar disso, a análise de séries temporais se firma como técnica de análise de dados relevante, tendo destaque quando aplicada a sistemas inerentemente não lineares (BRADLEY; KANTZ, 2015). Exemplos de aplicações de métodos orientados a dados incluem previsão de destino dinâmico para sistema de compartilhamento de carros sob demanda com base em sinal de GPS (WANG; ZHONG; MA, 2018), e estratégia de diagnóstico de falhas não supervisionada com base em agrupamento espectral (LIANG et al., 2019).

Durante o processo de medição de séries temporais, detalhes experimentais como taxa de amostragem, resposta dinâmica dos instrumentos e relação sinal-ruído podem afetar consideravelmente a confiabilidade dos resultados gerados. A adoção de técnicas de simbolização pode minimizar fatores degradantes, como o efeito do ruído (GRABEN, 2001). Essa técnica tem como principal característica a discretização das medidas das séries temporais em uma sequência simbólica em que os símbolos pertencem a um conjunto finito, de modo que o conteúdo de informação relevante contido nos dados temporais é mantido na sequência simbólica (LI et al., 2017). Através da compactação adequada do conjunto de dados de alta dimensão em arranjos de baixa dimensão, por meio da adoção da técnica denominada Filtragem Dinâmica Simbólica (SDF, *Symbolic Dynamic Filtering*), é possível alcançar ganhos adicionais por meio da captação de recursos de séries temporais (MUKHERJEE; RAY, 2014) e do comportamento dinâmico agregado (LI et al., 2017).

A análise baseada em dinâmica simbólica compreende um método de simbolização no qual uma série temporal é convertida em uma série simbólica (LIND; MARCUS, 1995). A etapa inicial consiste em particionar o intervalo de valores amostrais da série temporal em segmentos, de tal forma que cada amostra é associada a um único segmento. A simbolização é obtida mapeando-se cada segmento a um símbolo distinto. A sequência simbólica obtida é inspecionada em duas escalas de tempo: uma de tempo rápida, na qual assume-se que o processo tem dinâmica estatisticamente estacionária, e outra de tempo lenta, em que o processo pode apresentar dinâmica não estacionária. Nesse segundo caso, há possibilidade de se relacionar o processo com uma evolução gradual das propriedades estatísticas da dinâmica, devido à evolução gradual de anomalias (MUKHERJEE; RAY, 2014). A etapa seguinte da análise envolve a identificação de sequências de comprimento finito que apresentam padrões dinâmicos subjacentes significativos (LI; RAY, 2017). Estas sequências constituem blocos de construção para a elaboração de modelos dinâmicos a partir da sequência simbólica (GRABEN, 2001), utilizados por algoritmos para detecção de eventos (MUKHERJEE; RAY, 2014). Tais modelos devem ser analiticamente simples e precisos, fornecendo descrições estatísticas da sequência simbólica. Assim, tais modelos se apresentam como uma ferramenta computacionalmente atraente para representar o comportamento de um sistema, encontrando com isso aplicações em simulação

(BANDYOPADHYAY; BHATTACHARYA, 2014), análise de desempenho (PIMENTEL; ALAJAJI, 2009) e detecção de falhas (RAY, 2004), (SAXENA et al., 2020), (GHALYAN; RAY, 2021).

Autômatos probabilísticos de estado finito (PFSA, de *Probabilistic Finite State Automata*) são encontrados em uma variedade de áreas relacionadas a reconhecimento de padrões ou em campos aos quais o reconhecimento de padrões está vinculado, como linguística computacional, aprendizado de máquina, análise de séries temporais (VIDAL et al., 2005a). Os PFSA são frequentemente construídos a partir de cadeias de símbolos que, por sua vez, são geradas por particionamento de séries temporais de sinais de sensores (MUKHERJEE; RAY, 2014). Modelos baseados em PFSAs apresentam uma construção simples e que favorecem uma implementação computacional eficiente, se tornando assim uma ferramenta eficiente para captar a estrutura causal de um comportamento dinâmico observado (VIDAL et al., 2005a), (VIDAL et al., 2005b). Um exemplo de PFSA é a máquina D-Markov (RAY, 2004), caracterizada por um processo Markoviano no qual as estatísticas de todas as sequências de comprimento D são utilizadas para formação de seus estados; isso determina uma ordem finita igual a D que, à medida que se eleva, permite a tal máquina reproduzir o comportamento original do sistema de forma mais fidedigna, porém à custa de um aumento exponencial do seu número de estados. Exemplos de aplicação desses modelos são encontrados na modelagem do processo de ruído em sistemas de comunicação sem fio (PIMENTEL; ALAJAJI, 2009), (ALTINEL; KURT, 2017), e detecção de anomalias (RAY, 2004), (MUKHERJEE; RAY, 2014), (REN; YE; LI, 2017) (PENG et al., 2020), (GHALYAN; RAY, 2021).

A temática de classificação de padrões e detecção de anomalias tem sido foco de diversas pesquisas (PENG et al., 2020),(GHALYAN; RAY, 2021), (BHATTACHARYA; RAY, 2022). Dois casos típicos em que técnicas de detecção são amplamente aplicadas são aqueles em que (i) mudanças severas na série temporal ocorrem como consequência de variações abruptas do sistema dinâmico modelado, e (ii) ocorre divergência entre as estatísticas da série temporal observada e da série temporal utilizada para obter o modelo, caso em que o problema pode ser tratado como um teste de hipótese padrão (por exemplo, binário). No entanto, em muitas aplicações, essas mudanças ocorrem devido à degradação gradual no sistema dinâmico subjacente. Além disso, de acordo com a realidade prática, a distribuição de probabilidade após a ocorrência da mudança é tipicamente desconhecida. Nesse contexto, a fim de lidar com incertezas devido à imprecisão na modelagem de sistemas, autômatos não determinísticos, como Cadeias de Markov Ocultas (HMM, *Hidden Markov Model*) (HAJEK, 2015), surgem como uma extensão sólida dos autômatos clássicos e da teoria da linguagem formal, uma vez que o caráter não determinístico da estrutura algébrica de transição de estados desses modelos pode favorecer a captura de incertezas atreladas aos estados de um sistema (VIDAL et al., 2005b). Essa característica propicia uma expansão da classe de sistemas representáveis e, com isso, tem-se uma motivação para elaboração de modelos não determinísticos, aumentando-se o conjunto de estruturas de modelagem e o escopo de aplicações.

A abordagem de filtragem por dinâmica simbólica para detecção prévia de anomalias em sistemas dinâmicos complexos (RAY, 2004) concilia elementos de dinâmica simbólica, autômatos de estados finitos e ferramentas de reconhecimento de padrões para descrever quantitativamente o comportamento dinâmico de séries temporais. O emprego de recursos computacionais permite a construção de PFSAs a partir de sequências simbólicas. Um desvio do comportamento nominal é refletido no padrão estatístico da máquina de estados finitos que serve como uma indicação de anomalia do sistema. Logo, PFSAs representam a informação contida na série temporal simbólica por meio de um conjunto de estados interconectados e, a partir disso, constituem-se como um mecanismo eficiente para extração de características. Sua eficácia pode ser verificada por comparação com técnicas existentes, como redes neurais artificiais, análise de componentes principais, entre outras (BHATTACHARYA; RAY, 2022).

1.2 OBJETIVOS

O objetivo do presente trabalho é propor algoritmos para a construção de PFSAs, a partir das estatísticas de uma sequência simbólica observada S, utilizando duas metodologias. A primeira compreende um método de agrupamento de dados associado a uma técnica de aprendizado de máquina, o algoritmo *fuzzy c-means* (MIYAMOTO et al., 2008), e técnicas baseadas em minimização de grafos (BERSTEL et al., 2010) da teoria de autômatos (como Moore e generalizações) para obter modelos PFSA compactos e precisos, subdividida nas seguintes etapas:

- Construção de uma máquina D-Markov de memória D;
- Formação de agrupamentos (*clusters*) *fuzzy* de estados, pertencendo à mesma classe com probabilidade de transição (*morph*) semelhante, criando uma partição inicial $\mathcal{P}$ dos estados;
- Aplicação de um algoritmo baseado em minimização de grafo contendo estados não determinísticos para refinar a partição inicial por divisão de estados, a fim de melhorar a precisão do modelo resultante.

O algoritmo de agrupamento, ou clusterização, é uma forma eficiente de combinar estados com estatísticas de primeira ordem similares em classes de equivalência. Contudo, estados da mesma classe podem apresentar estatísticas de ordem superior distintas. A adoção de um algoritmo de minimização de grafo permite a identificação dessas distinções e fracionar a respectiva classe em subclasses, de forma a preservar a estatística de ordem superior dos estados em uma mesma classe.

A segunda metodologia proposta visa a elaboração de PFSAs via construção de uma estrutura em árvore baseada em redução de entropia, a qual busca assegurar a utilização de suas folhas no conjunto de estados do modelo gerado, e com isso reproduzir as estatísticas da

sequência modelada de forma análoga ou superior a outros métodos e com menor número de estados, tendo as seguintes etapas:

- Construção de estrutura em árvore regida por uma função custo baseada em redução de entropia;
- Análise das folhas obtidas a fim de garantir a presença destas no conjunto de estados do PFSA a ser gerado.

Em metodologias que empregam estruturas em árvore, restrições associadas às transições de estado podem fazer com que algumas folhas não sejam utilizadas como estados do PFSA, afetando potencialmente a qualidade do modelo resultante. Garantir que cada folha gerada pela árvore componha um estado do PFSA, evitando assim a sua exclusão durante a construção do modelo, serve como estratégia para resolver esta limitação.

São apresentados exemplos de modelagem para ilustrar que os algoritmos propostos produzem PFSAs com menor número de estados quando comparado ao de algumas máquinas encontradas na literatura, a saber, D-Markov, D-Markov com Clusterização e Minimização de Grafos (DCGraM, *D-Markov with Clustering and Graph Minimization*) e D-Markov de Comprimento Variável com Clusterização e Minimização de Grafos (VL-DCGraM, *Variable Length D-Markov with Clustering and Graph Minimization*), descritas no Capítulo 2, apresentando quantificadores de desempenho semelhantes. Ao variar o número de *clusters* e a quantidade de iterações no algoritmo de minimização de grafo, ou o tamanho do conjunto de folhas da árvore, é estabelecido um compromisso entre a cardinalidade do conjunto de estados e o desempenho alcançado.

1.3 ESTRUTURA DA TESE

Este trabalho está organizado em sete capítulos. No Capítulo 2 é revisado o embasamento teórico discutindo sequências discretas, PFSA, algoritmos de agrupamento e minimização de grafos; além disso, são mostrados os algoritmos D-Markov, DCGraM, VL-DCGraM e S2ER encontrados na literatura. O Capítulo 3 contém os detalhes de desenvolvimento do algoritmo *Preserved* S2ER (PS2ER). O Capítulo 4 aborda os detalhes de desenvolvimento do algoritmo para construção do modelo PFSA não determinístico (NDPFSA, *Non-deterministic PFSA*). No Capítulo 5 são apresentados alguns sistemas dinâmicos e descreve-se a análise de desempenho entre os algoritmos de modelagem apresentados quanto à geração de um PFSA. O Capítulo 6 trata da capacidade dos modelos quanto à identificação de séries anômalas a partir dos dados de ocupação de estados de suas respectivas máquinas. Finalmente, no Capítulo 7, conclusões e considerações para trabalhos futuros são apresentados.

2 PRELIMINARES SOBRE GRAFOS, AUTÔMATO PROBABILÍSTICO DE ESTADOS FINITOS E CLUSTERIZAÇÃO

Neste capítulo, são revisados conceitos de grafos e PFSA (LIND; MARCUS, 1995) (VIDAL et al., 2005a) que são necessários nos capítulos subsequentes. O conceito de minimização de grafos é apresentado e um algoritmo convencional para essa finalidade, Moore, é descrito. O algoritmo de clusterização *k-means* e algumas variações são apresentadas, constituindo uma etapa essencial para uma das propostas sugerida. Finalmente, quatro algoritmos já conhecidos para geração de PFSAs a partir de séries simbólicas são apresentados, a saber, D-Markov, DCGraM (FRANCH, 2017) ,VL-DCGraM (SANTOS; CHAVES; PIMENTEL, 2024) e S2ER (MUKHERJEE; RAY, 2014).

2.1 SEQUÊNCIAS DE SÍMBOLOS DISCRETOS

Introduzindo conceitos preliminares de linguagens formais (HOPCROFT, 2008), denomina-se como palavra uma sequência finita u de símbolos em um alfabeto Σ, sendo seu comprimento denotado por $|u|$. Uma sequência com comprimento igual a 0 é definida como a palavra vazia ε. O conjunto Σ^n representa todas as possíveis palavras de comprimento n, formadas por símbolos de Σ; e o conjunto formado por todas as sequências possíveis com símbolos de Σ, incluindo a sequência vazia ε, é dado por $\Sigma^* = \bigcup_{n=0}^{\infty} \Sigma^n$.

A concatenação de duas palavras u e $v \in \Sigma^*$ forma uma sequência uv. Como exemplo, dado um alfabeto binário $\Sigma = \{0, 1\}$, a concatenação de $u = 0011$ e $v = 011$ resulta em $uv = 0011011$. Vê-se que o comprimento de uv é igual à soma dos comprimentos de u e v, ou seja, $|uv| = |u| + |v|$. Para o conjunto Σ^*, o fechamento e a associatividade são satisfeitos com relação à concatenação, de modo que $u(vw) = (uv)w = uvw$; já a comutatividade não é satisfeita, pois uv e vu não são necessariamente iguais. Sendo a palavra vazia ε um elemento neutro para a concatenação, isto é, $\varepsilon u = u\varepsilon = u$, tem-se que Σ^* com a operação de concatenação determina um monoide, visto que possui um elemento identidade e constitui um conjunto com uma operação associativa (LALLEMENT, 1979).

Denomina-se sufixo da palavra $w \in \Sigma^*$ a palavra $v \in \Sigma^*$, $|w| > |v|$, para a qual w pode ser expressa como a concatenação uv para algum $u \in \Sigma^*$. De forma similar, u é denominada prefixo de w.

2.2 GRAFOS

No decorrer desta tese, o termo grafo é utilizado para denotar um gráfico direcionado rotulado conforme a Definição 2.1.

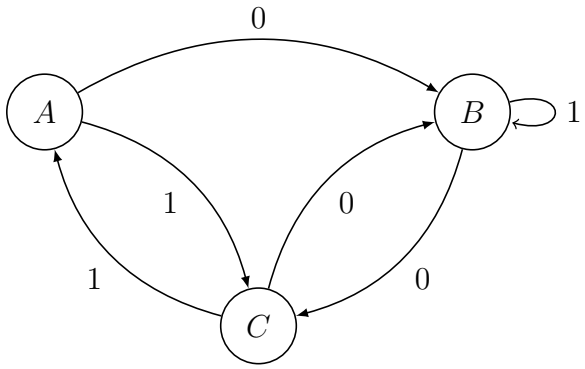
Definição 2.1 (Grafo). Um grafo *G* sobre o alfabeto Σ corresponde a uma tupla (Q, Σ, δ), na qual:

- Q representa um conjunto finito de estados com cardinalidade $|Q|$;
- Σ representa um alfabeto finito com cardinalidade $|\Sigma|$;
- δ representa a função de transição de estado $Q \times \Sigma \to Q$.

Uma forma de representar um estado $q \in Q$ é por meio de um ponto ou círculo e, se $\exists\, \delta(q, \sigma) = q'$, $q, q' \in Q$ e $\sigma \in \Sigma$, essa transição pode ser visualizada com uma seta direcionada partindo de q para q' rotulada com σ. Na Figura 1 é apresentado um exemplo de um grafo contendo três estados sobre um alfabeto binário, em que observa-se uma transição do estado *A* para o estado *B* com o símbolo 0, ou seja, $\delta(A, 0) = B$.

A função de transição pode ser ampliada a fim de aceitar não apenas símbolos isolados, mas palavras inteiras. Considerando $\omega \in \Sigma^n$, em que $\omega = \sigma_1\sigma_2 \cdots \sigma_n$ com $\sigma_j \in \Sigma$ para j de 1 a n, e dados os estados $q_0, q_1, \cdots, q_n \in Q$, define-se a função $\delta^*(q_0, \omega) = q_n$ se $\delta(q_0, \sigma_1) = q_1, \delta(q_1, \sigma_2) = q_2, \cdots, \delta(q_{n-1}, \sigma_n) = q_n$. Dados dois estados $q_1, q_2 \in Q$, se existe uma palavra $\omega \in \Sigma^*$ tal que $\delta^*(q_1, \omega) = q_2$, diz-se haver um caminho entre q_1 e q_2 e que ω é gerada por G. Na Figura 1, o caminho que parte do estado *A* e passa pelos estados *C*, *B*, *B*, *C*, *B*, *C* forma a palavra $\omega = 101000$, ou seja, $\delta^*(A, 101000) = C$.

Figura 1 – Um grafo de três estados com *Q* = {A, B, C} e $\Sigma = \{0, 1\}$.



Fonte: O autor.

Definição 2.2 (Contexto à Direita). Em um grafo G, o conjunto de todas as palavras formadas ao percorrer os caminhos iniciados no estado q e que terminam em algum estado de Q é definido como o contexto à direita de $q \in Q$:

$$F(q) = \{\omega \in \Sigma^* \mid \delta^*(q, \omega) \in Q\}. \tag{2.1}$$

Definição 2.3 (Linguagem de um grafo). Denomina-se linguagem de um grafo *G* o conjunto $\mathcal{L} \subset \Sigma^*$ formado pela união de contextos à direta de cada estado $q \in Q$:

$$\mathcal{L} = \bigcup_{q \in Q} F(q). \tag{2.2}$$

2.3 MINIMIZAÇÃO DE GRAFOS

Na atual seção aborda-se o tema da minimização de grafos, apresentando o algoritmo de Moore como um exemplo de algoritmo utilizado para a minimização de grafos determinísticos. Além disso, conforme discutido em (FRANCH, 2017), essa classe de algoritmos é aplicada para identificação de palavras com contextos à direita que apresentam estatísticas pouco similares.

2.3.1 Noções Preliminares

Para uma linguagem L, é possível existirem dois grafos $G_1 = (Q_1, \Sigma, \delta_1)$ e $G_2 = (Q_2, \Sigma, \delta_2)$ com $|Q_1| \neq |Q_2|$ capazes de gerá-la. Isso é justificado pelo fato de que uma linguagem L pode ser representada por diversos grafos diferentes. Dentre esses grafos, é preferível escolher aquele com o menor número de estados, pois isso favorece a diminuição da complexidade computacional dos algoritmos que utilizam essas representações, bem como da memória necessária para armazenar essas estruturas.

Definição 2.4 (Grafo Mínimo). Para uma determinada linguagem L, existe um grafo mínimo $G_{min} = (Q, \Sigma, \delta)$ capaz de gerá-la. Este grafo mínimo é caracterizado pela propriedade de que, para quaisquer $q_1, q_2 \in Q$, com $q_1 \neq q_2$, temos $F(q_1) \neq F(q_2)$.

Se um grafo $G_1 = (Q_1, \Sigma, \delta_1)$ possui dois estados distintos q_1 e q_2 que apresentam mesmo contexto à direita, é possível obter um novo grafo $G_2 = (Q_2, \Sigma, \delta_2)$, que gera a mesma linguagem, fundindo esses estados. Isto é, G_2 substitui os estados q_1 e q_2 por um estado q', de modo que, se algum estado e em G_1 possui uma transição com rótulo $\sigma \in \Sigma$ para q_1 ou q_2, então o estado e em G_2 terá uma transição com o mesmo rótulo σ para q'. Em outras palavras, para qualquer $e \in Q_1$ com transição $\delta_1(e, \sigma) = q_1$ ou $\delta_1(e, \sigma) = q_2$, com $\sigma \in \Sigma$, o estado e em Q_2 terá a transição $\delta_2(e, \sigma) = q'$. As transições a partir do estado q' serão definidas como $\delta_2(q', \sigma) = \delta_1(q_1, \sigma) = \delta_1(q_2, \sigma)$, para todo $\sigma \in \Sigma$. As linguagens geradas por G_1 e G_2 são equivalentes (HOPCROFT, 2008).

O processo de minimização de grafos envolve o particionamento do conjunto de estados Q de um grafo utilizando a relação de equivalência de Nerode (BERSTEL et al., 2010), a qual é descrita pela relação

$$p, q \in Q, p \equiv q \Leftrightarrow F(p) = F(q), \tag{2.3}$$

e a subsequente construção do grafo mínimo, cujos estados correspondem aos representantes das partições obtidas.

Definição 2.5 (Partições e Relações de Equivalência). Dado um conjunto E, uma partição de E é uma coleção $\mathcal{P}$ de subconjuntos não vazios e disjuntos P de E, tal que a união de todos os subconjuntos P em $\mathcal{P}$ é igual a E, ou seja, $\bigcup_{P \in \mathcal{P}} P = E$. O índice da partição refere-se ao

número de elementos que a compõem. A partição $\mathcal{P}$ estabelece uma relação de equivalência sobre E, e o conjunto de todas as classes de equivalência de uma relação de equivalência em E constitui uma partição do conjunto (HOPCROFT, 2008).

Definição 2.6 (Relação entre grafo mínimo e equivalência de Nerode). Um grafo é considerado mínimo se, e somente se, a partição gerada pela relação de equivalência de Nerode for a partição identidade.

A fim de apresentar o algoritmo de minimização, é necessário introduzir alguns conceitos fundamentais. Considerando um grafo G, dois algoritmos conhecidos na literatura para obtenção de um grafo mínimo são os algoritmos de Moore e de Hopcroft (BERSTEL et al., 2010). O algoritmo de Moore será descrito nesta seção, porém algumas definições são apresentadas previamente antes de detalhá-lo.

Um subconjunto F de E é dito ser saturado por uma partição $\mathcal{P}$, de E, quando corresponde à união de classes de $\mathcal{P}$. Dada uma outra partição $\mathcal{Q}$ de E, diz-se que Q é um refinamento de $\mathcal{P}$ (ou que $\mathcal{P}$ é menos refinada que $\mathcal{Q}$) se cada classe de $\mathcal{Q}$ está contida em alguma classe de $\mathcal{P}$, o que pode ser expresso como $Q \leq \mathcal{P}$. O índice de $\mathcal{Q}$ é maior ou igual ao índice de $\mathcal{P}$.

Dadas as partições $\mathcal{P}$ e $\mathcal{Q}$ de E, $\mathcal{U} = \mathcal{P} \wedge \mathcal{Q}$ representa a partição menos refinada que ainda refina tanto $\mathcal{P}$ quanto $\mathcal{Q}$. Os elementos de $\mathcal{U}$ são os conjuntos não vazios $P \cap Q$, em que $P \in \mathcal{P}$ e $Q \in \mathcal{Q}$. Esta notação pode ser estendida para múltiplos conjuntos, como $\mathcal{U} = \mathcal{P}_1 \wedge \mathcal{P}_2 \wedge \ldots \wedge \mathcal{P}_n = \bigwedge_{j=1}^{n} \mathcal{P}_j$.

Dado $F \subseteq E$, uma partição $\mathcal{P}$ de E induz uma partição $\mathcal{P}'$ de F por interseção. $\mathcal{P}'$ é formada pelos conjuntos $P \cap F$, para $P \in \mathcal{P}$. Se $\mathcal{P}$ e $\mathcal{Q}$ são respectivamente partições de E e $\mathcal{Q} \leq \mathcal{P}$, então as restrições $\mathcal{P}'$ e $\mathcal{Q}'$ a F também satisfazem $\mathcal{Q}' \leq \mathcal{P}'$.

Considerando um conjunto $P \subset Q$ e um símbolo $\sigma \in \Sigma$, denota-se o conjunto de estados $q \in Q$ tal que $\delta(q, \sigma) \in P$ por $\sigma^{-1}P$. Dados $P, R \subset Q$ e $\sigma \in \Sigma$, a partição $(P, \sigma)|R$ é composta por dois subconjuntos possivelmente não vazios, dados por

$$R \cap \sigma^{-1}P = \{r \in R \,\big|\, \delta(r, \sigma) \in P\} \tag{2.4}$$

e

$$R \backslash \sigma^{-1}P = \{r \in R \,\big|\, \delta(r, \sigma) \notin P\}. \tag{2.5}$$

Considere a extensão da função de transição sobre $P(Q) \times \Sigma$, em que $P(Q)$ representa o conjunto das partes de Q, definida como $\delta(A, \sigma) = B$, sendo $B = \{q \in Q \,|\, \exists\, p \in A, \delta(p, \sigma) = q\}$. O par (P, σ) é denominado divisor (*splitter*). Observe que $(P, \sigma)|R = R$ se $\delta(R, \sigma) \subset P$ ou $\delta(R, \sigma) \cap P = \emptyset$; e que $(P, \sigma)|R$ essencialmente se divide em duas classes não vazias se tanto $\delta(R, \sigma) \cap P \neq \emptyset$ quanto $\delta(R, \sigma) \cap P^c \neq \emptyset$, ou, de forma equivalente, se $\delta(R, \sigma) \not\subset P$ e $\delta(R, \sigma) \not\subset P^c$. Para o caso em que $(P, \sigma)|R$ é composto por duas classes, diz-se que (P, σ) divide R. Esta notação pode ser estendida para sequências, utilizando $\omega \in \Sigma^*$ em vez de $\sigma \in \Sigma$.

Proposição 2.1. *A partição $\mathcal{P}$ que corresponde à equivalência de Nerode é a partição menos refinada para qual nenhum divisor (P, σ), com $P \in \mathcal{P}$ e $\sigma \in \Sigma$, divide qualquer classe em $\mathcal{P}$, isto é, $(P, \sigma)|R = R$ para todos $P, R \in \mathcal{P}$ e $\sigma \in \Sigma$.*

2.3.2 Algoritmo Moore

Um algoritmo amplamente reconhecido na literatura para a minimização de grafos é o algoritmo de Moore (MOORE, 1956). Esse método baseia-se em uma abordagem de refinamento iterativo, que parte de uma partição inicial dos elementos de um conjunto, definida com base em um critério de equivalência. A partição é refinada ao identificar elementos de uma mesma classe de equivalência que transitam, com o mesmo rótulo, para classes de equivalência distintas. Esse processo de refinamento é repetido até que todas as transições, com o mesmo rótulo, de elementos de uma mesma classe de equivalência levem a uma única classe de equivalência. O Algoritmo 1 representa um esboço deste processo, identificando o grafo mínimo que gera a linguagem produzida pelo grafo G, a partir da partição inicial $\mathcal{P}$.

Algoritmo 1 Moore

1: **Entradas:** $G, \mathcal{P}$
2: **Saída:** partição $\mathcal{P}$ refinada
3: **repita**
4: $\quad \mathcal{P}' \leftarrow \mathcal{P}$
5: $\quad$ **para todo** $\sigma \in \Sigma$ **faça**
6: $\quad\quad \mathcal{P}_\sigma \leftarrow \bigwedge_{P \in \mathcal{P}} (P, \sigma)|Q$
7: $\quad \mathcal{P} \leftarrow \mathcal{P} \wedge \bigwedge_{\sigma \in \Sigma} \mathcal{P}_\sigma$
8: **até que** $\mathcal{P} = \mathcal{P}'$

Seja um grafo $G = (Q, \Sigma, \delta)$ e uma partição inicial de Q, $\mathcal{P} = \{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$. O conjunto $L_q^{(h)}$ é definido como

$$L_q^{(h)}(G) = \left\{ w \in \Sigma^* \;\middle|\; |w| \leq h, \delta^*(q, w) \in \mathcal{P}_j \right\}, \tag{2.6}$$

em que $\mathcal{P}_j \in \mathcal{P}$ contém todas as palavras de comprimento, no máximo, h capazes de serem geradas partindo-se de $q \in Q$ e alcançando um estado na classe de equivalência $\mathcal{P}_j$. Define-se a equivalência Moore de ordem h, denotada por $\equiv_h$, como

$$p \equiv_h q \Leftrightarrow L_p^{(h)}(G) = L_q^{(h)}(G), \tag{2.7}$$

a qual estabelece que dois estados são considerados equivalentes se gerarem as mesmas palavras de comprimento, no máximo, h que atingem um estado em $\mathcal{P}_j$. Quando a partição gerada pela relação $\equiv_h$ coincide com a obtida pela aplicação da relação de equivalência de Nerode $\equiv$ sobre $\mathcal{P}_j$, chega-se à profundidade do algoritmo de Moore. Esta profundidade é definida como o menor número inteiro h tal que $\equiv_h$ é igual a $\equiv h + 1$. Isso resulta em um algoritmo que aplica sucessivamente a relação de equivalência de Moore até que a partição não possa mais ser refinada.

Proposição 2.2. *Para dois estados $p, q \in Q$ e $h \geq 0$,*

$$p \equiv_{h+1} q \iff p \equiv_h q \text{ e } \delta(p,\sigma) \equiv_h \delta(q,\sigma),\ \forall \sigma \in \Sigma. \tag{2.8}$$

Adotando-se esta formulação e definindo $\mathcal{M}_h$ como a partição definida pela equivalência Moore de profundidade h, a proposição a seguir é válida.

Proposição 2.3. *Para $h \geq 0$,*

$$\mathcal{M}_{h+1} = \mathcal{M}_h \wedge \bigwedge_{\sigma \in \Sigma} \bigwedge_{P \in \mathcal{M}_h} (P,\sigma)|Q. \tag{2.9}$$

Esse cálculo implica que, para cada símbolo $\sigma \in \Sigma$ e para cada classe de equivalência $P \in \mathcal{M}_h$, sendo $\mathcal{M}_h$ a partição da iteração anterior, um divisor (P,σ) é criado e aplicado ao conjunto original de estados Q. Isso gerará partições que indicam quais estados de Q conduzem a estados em P com o símbolo σ e quais não conduzem. Dessa forma, obtém-se a partição mais refinada $\bigwedge_{P \in \mathcal{M}_h} (P,\sigma)|Q$, pois refinará $\mathcal{M}_h$ em conjuntos de estados que conduzem às mesmas classes com transições rotuladas por σ. Em seguida, a partição mais refinada $\bigwedge_{\sigma \in \Sigma} \bigwedge_{P \in \mathcal{M}_h} (P,\sigma)|Q$ é obtida entre essas classes de equivalência, dividindo Q em classes que conduzem à mesma classe em $\mathcal{M}_h$ para cada símbolo distinto de Σ. Isso efetivamente calcula o segundo membro $\delta(p,\sigma) \equiv_h \delta(q,\sigma),\ \forall \sigma \in \Sigma$ em (2.8). Finalmente, em (2.9), a partição mais refinada resultante dessa última etapa e de $\mathcal{M}_h$ é obtida, resultando no incremento das classes de equivalência de Moore.

O cálculo em (2.9) é executado no Algoritmo 1,ç no qual a iteração refina a partição atual até que não se identifique alterações entre $\mathcal{M}_h$ e $\mathcal{M}_{h+1}$, indicando que a equivalência Nerode é alcançada. Para constituição de um grafo, a partição inicial é criada pela junção de estados em Q que apresentam transições com rótulos iguais.

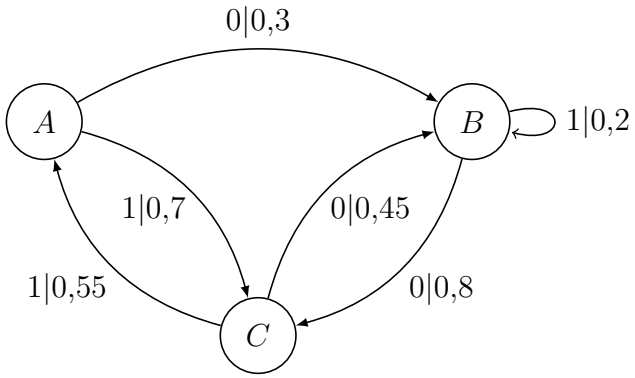
2.4 AUTÔMATO PROBABILÍSTICO DE ESTADOS FINITOS

Definição 2.7 (**Autômato Probabilístico de Estados Finitos**)**.** Um autômato probabilístico de estados finitos (PFSA, *Probabilistic Finite State Automata*) é caracterizado por um grafo G e uma função π definida sobre seus ramos, representado pelo par (G,π). A função $\pi : Q \times \Sigma \to [0,1]$ é tal que, para qualquer $q \in Q$, $\sum_{\sigma \in \Sigma} \pi(q,\sigma) = 1$, estabelecendo assim uma distribuição de probabilidade associada a cada estado de G.

Definição 2.8 (**Morph**)**.** Denomina-se *morph* de um estado $q \in Q$ a distribuição de probabilidade $\mathcal{V}(q) = \{\pi(q,\sigma); \forall \sigma \in \Sigma\}$.

Ao representar-se graficamente um PFSA, cada transição partindo de um estado q é associada a um rótulo que consiste em um símbolo $\sigma \in \Sigma$ e uma probabilidade $\pi(q, \sigma)$, indicando a probabilidade de transitar a partir de q com o símbolo σ. Um exemplo de PFSA é ilustrado na Figura 2, considerando $Q = \{A, B, C\}$ e $\Sigma = \{0, 1\}$, que observa-se tratar do mesmo grafo da Figura 1, porém com probabilidades associadas às suas transições para formar o PFSA.

Figura 2 – PFSA com grafo correspondente ao da Figura 1.



Fonte: O autor.

Dado um PFSA (G, π), existe uma probabilidade associada a cada palavra $\omega \in \Sigma^*$ que pode ser formada a partir de cada estado de G. Na Figura 2, partindo do estado A, é possível obter-se a palavra $\omega = 1010001$ (uma vez que $\delta^*(A, \omega) = A$) ao transitar através dos estados C, B, B, C, B, C e A e concatenar os rótulos do caminho para cada uma das transições entre estados. Realizando o produto das probabilidades em cada transição de saída, temos que $\Pr(\omega|A) = 0{,}7 \times 0{,}45 \times 0{,}2 \times 0{,}8 \times 0{,}45 \times 0{,}8 \times 0{,}55 = 0{,}0099792$.

Seja **P** a matriz de transição de probabilidades $|Q| \times |Q|$ de um PFSA, na qual sua entrada (i,j) representa a probabilidade de transição do estado q_i para o estado q_j, para $q_i, q_j \in Q$. O vetor estacionário de **P**, denotado por $\boldsymbol{\mu} = [\mu_1, \mu_2, \ldots, \mu_{|Q|}]$, é o autovetor associado ao autovalor 1 e satisfaz a relação $\boldsymbol{\mu}\mathbf{P} = \boldsymbol{\mu}$. A matriz de transição de probabilidades e o vetor estacionário do PFSA da Figura 2 são dados por

$$\mathbf{P} = \begin{bmatrix} 0 & 0{,}3 & 0{,}7 \\ 0 & 0{,}2 & 0{,}8 \\ 0{,}55 & 0{,}45 & 0 \end{bmatrix} \tag{2.10}$$

$$\boldsymbol{\mu} = \begin{bmatrix} 0{,}2372 & 0{,}3315 & 0{,}4313 \end{bmatrix}. \tag{2.11}$$

Definem-se duas matrizes **P**(0) e **P**(1), onde **P**(0) + **P**(1) = **P**, tal que o elemento (i, j) da matriz **P**(σ), $\sigma \in \{0, 1\}$, é a probabilidade condicional de transitar do estado i para o estado j e emitir o símbolo σ. Para o PFSA da Figura 2, as matrizes **P**(0) e **P(1)** são:

$$\mathbf{P}(0) = \begin{bmatrix} 0 & 0{,}3 & 0 \\ 0 & 0 & 0{,}8 \\ 0 & 0{,}45 & 0 \end{bmatrix} \tag{2.12}$$

$$\mathbf{P}(1) = \begin{bmatrix} 0 & 0 & 0{,}7 \\ 0 & 0{,}2 & 0 \\ 0{,}55 & 0 & 0 \end{bmatrix}. \tag{2.13}$$

2.5 CLUSTERIZAÇÃO

Na atual seção é descrito um algoritmo relacionado a uma técnica de aprendizado de máquina conhecida como clusterização, sendo utilizado para formar a partição inicial a ser empregada em um algoritmo de minimização de grafos.

O termo clusterização refere-se a métodos em que os elementos de um conjunto de N objetos são agrupados em K grupos, denominados *clusters*, que apresentam algum grau de similaridade (MACKAY, 2003). Neste trabalho, a clusterização é aplicada para realizar o agrupamento de estados que apresentam *morphs* semelhantes.

2.5.1 *k-means*

O Algoritmo *k-means*, também conhecido como Algoritmo de Lloyd (KANUNGO et al., 2002), visa particionar um conjunto de dados em k *clusters*, sendo k um valor predefinido. Um *cluster* pode ser concebido como um agrupamento de pontos de dados dispostos à pequenas distâncias (definidas segundo uma determinada métrica) entre si em comparação com as distâncias para pontos fora do *cluster*. Esta ideia é formalizada ao se introduzir inicialmente um conjunto de vetores I-dimensionais $\mathbf{m}^{(k)}$, com $k = 1, \ldots, K$. Cada vetor $\mathbf{m}^{(k)}$ está associado ao k-ésimo agrupamento e pode ser interpretado como o centro do agrupamento, também conhecido como centróide.

Para realização do processo de clusterização, almeja-se um mecanismo para o agrupamento de pontos de dados, bem como um conjunto de vetores $\mathbf{m}^{(k)}$, tal que a soma dos quadrados das distâncias de cada ponto ao vetor mais próximo $\mathbf{m}^{(k)}$ seja minimizada. Para tanto, a cada ponto de dados $\mathbf{x}_n$ é associada uma variável de indicadores binários $r_{nk} \in 0, 1$, em que $k = 1, \ldots, K$ e $n = 1, \ldots, N$, determinando-se a qual dos K grupos o ponto de dados $\mathbf{x}_n$ está associado. Dessa forma, se $\mathbf{x}_n$ estiver associado ao agrupamento k, então $r_{nk} = 1$ e $r_{nj} = 0$ para $j \neq k$. A partir disso, pode-se definir uma função objetiva, por vezes denominada medida de distorção, expressa por

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \|\mathbf{x}_n - \mathbf{m}^{(k)}\|^2 \tag{2.14}$$

que representa a soma dos quadrados das distâncias de cada ponto de dados para o vetor atribuído $\mathbf{m}^{(k)}$. O objetivo é determinar valores para r_{nk} e $\mathbf{m}^{(k)}$ que minimizem J. Isso pode ser alcançado por meio de um procedimento iterativo, no qual cada iteração compreende duas etapas sucessivas de otimização em relação a r_{nk} e a $\mathbf{m}^{(k)}$. Inicialmente, escolhem-se valores iniciais para $\mathbf{m}^{(k)}$. Em seguida, na primeira fase, minimiza-se J em relação a r_{nk} enquanto $\mathbf{m}^{(k)}$ permanece fixo. Na segunda fase, minimiza-se J em relação a $\mathbf{m}^{(k)}$, mantendo r_{nk} fixo. Este processo de otimização em dois estágios é repetido iterativamente até que um critério de convergência seja alcançado.

Primeiramente, aborda-se a determinação de r_{nk}. Como a função J em (2.14) é uma função linear de r_{nk}, essa otimização pode ser conduzida para fornecer uma solução fechada. Os termos que envolvem n distintos são independentes, permitindo a otimização isolada para cada n. Isso é alcançado atribuindo $r_{nk} = 1$ para qualquer valor de k que minimize $\|\mathbf{x}_n - \mathbf{m}^{(k)}\|^2$, enquanto $r_{nk} = 0$ para os demais valores de k. Em outras palavras, cada ponto de dados é atribuído ao centro do agrupamento mais próximo. Isso pode ser formalizado como:

$$r_{nk} = \begin{cases} 1, & \text{se } k = \arg\min_j \|\mathbf{x}_n - \mathbf{m}^{(j)}\|^2 \\ 0, & \text{caso contrário.} \end{cases} \tag{2.15}$$

Prossegue-se com a otimização de $\mathbf{m}^{(k)}$ com r_{nk} mantido fixo. A função objetiva J é uma função quadrática de $\mathbf{m}^{(k)}$ e pode ser minimizada igualando a zero sua derivada em relação a $\mathbf{m}^{(k)}$, ou seja,

$$2\sum_{n=1}^{N} r_{nk}(\mathbf{x}_n - \mathbf{m}^{(k)}) = 0, \tag{2.16}$$

o que resulta em

$$\mathbf{m}^{(k)} = \frac{\sum_n r_{nk}\mathbf{x}_n}{\sum_n r_{nk}}. \tag{2.17}$$

O denominador nesta expressão corresponde ao número de pontos atribuídos ao *cluster* k. Assim, essa expressão implica que $\mathbf{m}^{(k)}$ é igual à média de todos os pontos $\mathbf{x}_n$ atribuídos ao *cluster* k.

As duas fases de realocação de pontos de dados para *clusters* e recálculo dos centroides são executadas sucessivamente até que não ocorram mais alterações nas atribuições (ou até que um número máximo de iterações seja atingido). Como cada fase reduz o valor de J, a convergência do algoritmo é assegurada.

O algoritmo completo é apresentado no Algoritmo 2 , seguido de uma explicação de cada etapa.

Etapa 1: Inicialização

Uma prática comum para a inicialização das médias $\mathbf{m}^{(k)}$ é atribuir valores arbitrários de $\mathbb{R}^I$ para cada média. Embora existam outras heurísticas que podem ser empregadas-(HAMERLY;

Algoritmo 2 *k-means*

Entradas: x_n, k
Saídas: K *clusters* contendo todos os pontos de dados x_n
Etapa 1: Inicialização:
para $k \in 1, \ldots, K$ **faça**
    $\mathbf{m}^{(k)} \leftarrow$ *random*()
Etapa 2: Atribuição
para $n = 1, \ldots, N$ **faça**
    para $k = 1, \ldots, K$ **faça**
        se $k = \arg\min_j \parallel \mathbf{x}_n - \mathbf{m}^{(j)} \parallel^2$ **então**
            $r_{nk} \leftarrow 1$
        senão
            $r_{nk} \leftarrow 0$
Etapa 3: Atualização
para $k = 1, \ldots, K$ **faça**
    $R^{(k)} \leftarrow \sum_n^n r_{nk}$
    $\mathbf{m}^{(k)} = \frac{\sum_n r_{nk}\mathbf{x}_n}{R^{(k)}}$
Etapa 4: Repetição
se Atribuições modificadas **então**
    Repetir etapas 2 e 3
senão
    retorne $\{r_{nk}, \forall n \in 1, \ldots, N\}$

ELKAN, 2002), a inicialização aleatória frequentemente é adequada para a maioria das aplicações.

Etapa 2: Atribuição

Para cada $\mathbf{x}_n$ para $n = 1, \ldots, N$ e $k = 1, \ldots, K$, o valor de r_{nk} é calculado conforme a equação (2.15) para ser empregado na etapa de atualização.

Etapa 3: Atualização

Nesta etapa, os valores das médias são atualizados, assumindo a posição média dos pontos contidos em cada *cluster*. Para um determinado grupo k, $\mathbf{m}^{(k)}$ é atualizado com $\frac{\sum_n r_{nk}x_n}{R(k)}$, em que $\sum_n r_{nk}x_n$ representa apenas os pontos de dados no agrupamento k, visto que r_{nk} é zero para qualquer ponto de dados externo ao *cluster*, e $R^{(k)}$ é atualizado para $\sum_n r_{nk}$, que corresponde ao total de pontos de dados no *cluster* k.

Etapa 4: Repetição

As etapas 2 e 3 são repetidas até que os agrupamentos antes e após a iteração sejam iguais, indicando que as médias atingiram o centroide de cada agrupamento e que a ação está encerrada.

2.5.2 *fuzzy c-means*

A clusterização *fuzzy c-means*, proposta por Bezdek (BEZDEK, 1981), compreende um algoritmo de agrupamento não hierárquico cujo objetivo é fornecer uma classificação *fuzzy* de um conjunto de elementos em c *clusters*, sendo c um número pré-determinado.

No contexto das partições geradas por abordagens de clusterização, denomina-se clusterização *hard* aquela em que cada elemento pertence a um e apenas um *cluster*, caracterizando-os como conjuntos disjuntos, como é o caso da clusterização k-*means*. Em contrapartida, na clusterização *fuzzy* permite-se a associação de cada elemento com todos os *clusters* através de uma função de pertinência (ZADEH, 1996). Esse conceito de conjuntos *fuzzy* é vantajoso à representação de cenários em que um determinado elemento possui algum nível de semelhança com os diversos agrupamentos existentes, permitindo que um algoritmo associe parcialmente cada ponto a mais de um *cluster*.

O método de clusterização *fuzzy c-means* pode ser descrito da seguinte maneira: dado um conjunto de elementos $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, em que x_k é comumente um vetor de características $x_k = [x_{k_1}, x_{k_2}, \ldots, x_{k_p}] \in \mathbb{R}^p$, sendo $k \in \{1, 2, \ldots, n\}$, e $\mathbb{R}^p$ o espaço p-dimensional, busca-se encontrar uma pseudopartição *fuzzy* que forneça uma representação conveniente dos dados. Essa pseudopartição de X corresponde a uma família de c subconjuntos *fuzzy* $P = \{F_1, F_2, \ldots, F_c\}$ que satisfaça as Equações 2.18 e 2.19:

$$\sum_{i=1}^{c} F_i\left(x_k\right) = 1, \tag{2.18}$$

$$0 < \sum_{k=1}^{n} F_i\left(x_k\right) < n, \tag{2.19}$$

nas quais $F_i\left(x_k\right)$ é denominado valor ou grau de pertinência de um elemento x_k em relação ao *cluster* i. Assim, de acordo com (2.18), para todo $k \in \{1, 2, \ldots, n\}$, sendo n o total de elementos de X, o somatório dos valores de pertinência de um elemento relativos a cada um dos subconjuntos deve ser unitário; e de acordo com (2.19), para todo $i \in \{1, 2, \ldots, c\}$, em que c indica o número de agrupamentos, o somatório dos valores de pertinência de todos os elementos em um subgrupo deve ser menor que o total de elementos contidos no universo de X.

A partir dos graus de pertinência $F_i(x_k)$ e do conjunto de centroides v_i, o desenvolvimento de um algoritmo iterativo para o método *fuzzy c-means* baseia-se na minimização de um índice de desempenho $J_m(P)$, definido por

$$J_m(P) = \sum_{k=1}^{n} \sum_{i=1}^{c} \left[F_i(x_k)\right]^m \left\|x_k - v_i\right\|^2, \tag{2.20}$$

que denota a soma das distâncias ponderadas de cada ponto de dado x_k ao centroide v_i, por meio do cálculo dos valores $F_i(x_k)$ e v_i que minimizem esse índice. A variável m deve ter valor real maior que 1 e representa um índice associado a distância admitida entre pontos de

dados e o centroide calculado na iteração atual, de forma que quanto maior o valor de m, maior se torna a associação de um elemento a cada uma das pseudopartições. Seu valor é escolhido convenientemente para o problema analisado, não havendo base teórica para determinação de um valor ótimo de m.

A eficiência do algoritmo depende de diversos fatores, tais como o número de *clusters* adotado, a alocação inicial dos centroides, a métrica de distância utilizada, o critério de parada do algoritmo e até mesmo as propriedades geométricas dos dados. Para a obtenção de agrupamentos adequados, é necessária a execução extensiva de testes, verificando-se variações nos principais parâmetros mencionados.

As etapas do algoritmo são descritas a seguir.

Etapa 1: Inicialização

Seja $P^{(t)}$ uma pseudopartição definida na iteração t. Como etapa inicial do algoritmo, para a iteração inicial $t = 0$, define-se a pseudopartição $P^{(0)} = \{F_1, F_2, \ldots, F_c\}$ especificando os valores de pertinência iniciais F_i dos n elementos de X às c classes dessa pseudopartição, sendo esses valores informados ou, mais comumente, gerados de forma aleatória.

Etapa 2: Atribuição

Para $t = 0$, atribuem-se c centroides $v_1^{(0)}, v_2^{(0)}, \ldots, v_c^{(0)}$ iniciais (fornecidos diretamente ou estipulados aleatoriamente). Para $t > 0$, calculam-se os centros de cada *cluster* $v_1^{(t)}, v_2^{(t)}, \ldots, v_c^{(t)}$, com relação a $P^{(t)}$ e ao valor fornecido da variável m, por meio da equação

$$v_i^{(t)} = \frac{\sum_{k=1}^{n} [F_i(x_k)]^m x_k}{\sum_{k=1}^{n} [F_i(x_k)]^m}. \tag{2.21}$$

Os centroides em $t = 0$ também podem ser calculados de acordo com (2.21) ao invés de serem fornecidos (YONAMINE et al., 2002); contudo deve-se evitar a ocorrência de centros de *clusters* idênticos para classes diferentes, pois tal inicialização pode levar à reincidência desses valores durante as iterações posteriores do algoritmo, afetando negativamente os resultados obtidos.

O vetor v_i calculado em (2.21) determina o centro do *cluster* C_i, correspondendo a uma média ponderada dos elementos que o compõem, onde o peso de x_k é a potência de ordem m do grau de pertinência desse ao conjunto *fuzzy* C_i.

Etapa 3: Atualização

Para cada $x_k \in X$ e para todo $i \in \{1, 2, \ldots, c\}$, atualiza-se $P^{(t)}$ para $P^{(t+1)}$ calculando-se o valor de pertinência de x_k a C_i por (2.22):

$$F_i^{(t+1)}(x_k) = \left[\sum_{j=1}^{c} \left(\frac{\left\| x_k - v_i^{(t)} \right\|^2}{\left\| x_k - v_j^{(t)} \right\|^2} \right)^{\left(\frac{1}{m-1}\right)} \right]^{-1} \tag{2.22}$$

na qual $||\cdot||$ representa uma norma assumida para o produto interno no espaço R^p e $\left\| x_k - v_i^{(t)} \right\|^2$ representa a distância entre x_k e v_i.

Verifica-se que (2.22) relaciona a distância de cada elemento x_k ao centroide v_i considerado na iteração atual com a distância desse mesmo elemento aos centroides v_j dos demais agrupamentos, gerando um valor de pertinência de $F_i(x_k)$ ao *cluster* C_i que apresenta proporcionalidade a seus valores de pertinência com relação aos demais *clusters*.

Para o caso em que $\left\| x_k - v_i^{(t)} \right\|^2 = 0$ para algum $i \in I \subseteq \{1, 2, \ldots, c\}$ (ou seja, x_k coincide com o centro de *cluster* v_i), define-se $F_i^{(t+1)}(x_k)$ como um número real não negativo que satisfaz à equação

$$\sum_{i \in I} F_i^{(t+1)}(x_k) = 1 \tag{2.23}$$

e adota-se $F_i^{(t+1)}(x_k) = 0$ para todo $i \in \{1, 2, \ldots, c\} - I$. A possibilidade de ocorrência desse tipo de situação torna a atribuição feita em (2.23) necessária, uma vez que um elemento pode ter sido associado a outros *clusters* com pertinência maior que zero se C_i não for o primeiro *cluster* selecionado pelo algoritmo, o que demonstra a influência da ordem de seleção das pseudopartições.

Etapa 4: Repetição

Como etapa final, compara-se $P^{(t)}$ e $P^{(t+1)}$ através da seguinte condição de parada: se $|P^{(t)} - P^{(t+1)}| < \varepsilon$, sendo ε um valor definido, o algoritmo é encerrado; caso contrário, faz-se o incremento $t = t + 1$ e retorna-se ao Passo 2.

Nessa etapa, o termo $|P^{(t)} - P^{(t+1)}|$ corresponde à distância entre as pseudopartições anterior e atual, a qual é dada por

$$|P^{(t)} - P^{(t+1)}| = \max_{i,k} \left| F_i^{(t+1)}(x_k) - F_i^{(t)}(x_k) \right|. \tag{2.24}$$

A Equação (2.24) permite verificar se a máxima diferença entre o grau de pertinência de qualquer elemento x_k em um dado agrupamento C_i na iteração atual e na iteração passada é menor que o erro definido ε, verificação feita para todos os agrupamentos. Em caso positivo, o algoritmo é concluído e os valores de pertinências atuais de todos os elementos em todas as classes são retornados como resultado. Percebe-se que quanto menor o valor de ε maior o número de passos necessários para o encerramento do algoritmo, implicando em pseudopartições mais refinadas obtidas ao final do processo.

O algoritmo *fuzzy c-means* é o representado no Algoritmo 3.

Algoritmo 3 *Fuzzy c-means*

Entradas: X, c
Saídas: c *clusters* contendo todos os pontos de dados X
Etapa 1: Inicialização:
$P^{(t)} \leftarrow P^{(0)}$
Etapa 2: Atribuição
para $t = 1, \ldots, T$ **faça**
    para $i = 1, \ldots, c$ **faça**
        $v_i^{(t)} = \frac{\sum_{k=1}^{n} [F_i(x_k))]^m x_k}{\sum_{k=1}^{n} [F_i(x_k))]^m}$
Etapa 3: Atualização
para $i = 1, \ldots, c$ **faça**
    $F_i^{(t+1)}(x_k) = \left[\sum_{j=1}^{c} \left(\frac{\left\| x_k - v_i^{(t)} \right\|^2}{\left\| x_k - v_j^{(t)} \right\|^2} \right)^{\left(\frac{1}{m-1}\right)} \right]^{-1}$
Etapa 4: Repetição
se $|P^{(t)} - P^{(t+1)}| \geq \varepsilon$ **então**
    Retornar a Etapas 2
senão
    retorne $P^{(t)}$

2.6 ESTRUTURA EM ÁRVORE

Denomina-se como árvore uma estrutura composta por estados rotulados, denominados nós, que são gerados recorrentemente a partir de um nó raiz. Cada nó da árvore é rotulado com palavras sobre o alfabeto Σ, sendo o rótulo do nó raiz a palavra nula ε. De cada nó da árvore partem $|\Sigma|$ ramos cada um rotulado com um elemento de Σ. O rótulo de um nó é formado pela concatenação dos rótulos dos ramos que partem de ε até esse nó. O conjunto de nós alcançados pelos ramos que emanam de um nó rotulado como ω é referido como os nós filhos de ω, cada um rotulado como $\sigma\omega$, para $\sigma \in \Sigma$, sendo ω o nó pai correspondente. Cada nó, exceto a raiz, tem exatamente um nó pai. Os nós terminais da árvore (ou seja, aqueles que não possuem nós filhos), são denominados folhas. Para cada nó ω da árvore atribui-se a probabilidade P_ω e a função $\pi : \omega \times \Sigma \rightarrow [0, 1]$, a qual satisfaz $\sum_{\sigma \in \Sigma} \pi(\omega, \sigma) = 1$. O conjunto de folhas de uma árvore é definido por $\Gamma = \{w_1, w_2, \cdots, w_m\}$. Tanto P_ω como π são estimados a partir da sequência simbólica S modelada sobre o alfabeto Σ.

Árvores de Contexto

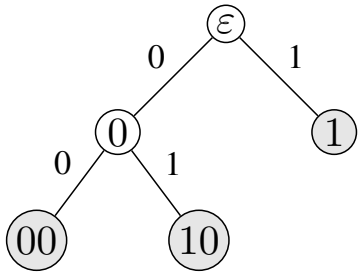
Há classes de processos caracterizados por apresentar memória constante, como observado nas máquinas D-Markov (abordadas na seção seguinte), tendo memória de tamanho fixo e igual a D e cuja geração envolvem a especificação de $|\Sigma|^D(|\Sigma| - 1)$ parâmetros. No entanto, existem classes de processos de ordem D (memória máxima) que são satisfeitos para palavras com comprimento menor que D, para algumas sequências anteriores $x_{n-D}...x_{n-1}$, podendo assim ser representados por PFSAs mais compactos. Um processo é chamado de cadeia de

Markov de comprimento variável (VLMC, *Variable Lenght Markov Chain*) de ordem D quando sua memória varia dependendo dos símbolos anteriores até o comprimento D (BABICH; KELLY; LOMBARDI, 1999; BÜHLMANN, 2000). Um conceito importante para caracterizar um VLMC de ordem D é a função de contexto de ordem D, $c : \Sigma^D \to C$. Ela garante que para todo $w \in \Sigma^D$ existe $s \in C$ que satisfaz $Pr(\sigma|w) = Pr(\sigma|s)$, para todo $\sigma \in \Sigma$. Assim, $c(w) = s$. Observe que C é um conjunto mínimo, portanto, $c()$ está bem definido. O contexto refere-se à sequência de símbolos anteriores que influencia a ocorrência do símbolo atual.

A determinação do conjunto C pode ser feita com uma árvore de contexto (CT, de *Context Tree*). A Figura 3 exemplifica essa construção para $\Sigma = \{0, 1\}$, $D = 2$. Cada nó tem exatamente um predecessor e $|\Sigma|$ sucessores (exceto as folhas que não apresentam sucessores, e o nó raiz com o rótulo ϵ que não apresenta predecessor). Nesta figura, as folhas são representadas por um círculo sombreado. Com exceção do nó raiz, cada nó é rotulado com a palavra formada pelo rótulo de seu predecessor concatenado à direita com o símbolo que rotula essa transição. O rótulo de uma folha corresponde a um contexto de um VLMC. Assim, $C = \{00, 10, 1\}$.

A Figura 3 ilustra uma árvore binária enraizada com 3 folhas (nós sombreados). Um caminho do nó ε até o nó rotulado como ω percorre nós rotulados com os sufixos de ω.

Figura 3 – Representação de uma árvore binária com $\Sigma = \{0,1\}$. O conjunto de folhas da árvore é $\Gamma = \{00, 10, 1\}$.



Fonte: O autor.

2.7 ALGORITMOS PARA CONSTRUÇÃO DE PFSAS

Na atual seção, quatro algoritmos que constroem um PFSA a partir de uma sequência S de comprimento N sobre um alfabeto Σ são apresentados: Máquinas D-Markov, DCGraM, VL-DCGraM e S2ER.

2.7.1 Máquinas D-Markov

Uma máquina D-Markov (MUKHERJEE; RAY, 2014) é um Autômato Probabilístico de Estados Finitos que produz símbolos com dependência limitada ao histórico de no máximo D símbolos anteriores, em que D representa a profundidade da máquina. Este autômato gera um processo Markoviano $\{s_n\}$ de ordem D tal que

$$\Pr(s_n| \dots s_{n-D} \dots s_{n-1}) = \Pr(s_n|s_{n-D} \dots s_{n-1}). \tag{2.25}$$

Para o desenvolvimento de uma máquina D-Markov, inicialmente são consideradas todas as possíveis sequências sobre Σ de comprimento D como estados de Q, resultando em um total de $|\Sigma|^D$ estados. Para cada $\sigma \in \Sigma$, a transição com rótulo σ a partir do estado q é determinada da seguinte forma: Seja q o estado rotulado como $q = \sigma_1\sigma_2 \ldots \sigma_D$ com $\sigma_n \in \Sigma$ para $n = 1, 2, \ldots, D$. Cria-se uma transição de q para $q' = \sigma_2 \ldots \sigma_D\tau$, ou seja, $\delta(q, \tau) = q'$, com probabilidade:

$$\Pr(\tau \mid q) = \frac{\Pr(q\tau)}{\Pr(q)}, \tag{2.26}$$

em que $\Pr(q)$ e $\Pr(q\tau)$ correspondem às probabilidades das sequências $\sigma_1\sigma_2 \ldots \sigma_D$ e $\sigma_2\sigma_3 \ldots \sigma_D\tau$, respectivamente.

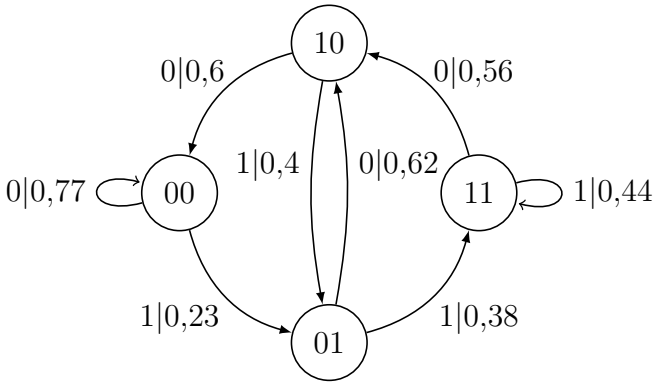
Por exemplo, considere uma sequência binária S com as probabilidades de palavras de comprimento $\ell \leq 3$ mostradas na Tabela 1 . Para formação de uma máquina 2-Markov, os estados são 00, 01, 10 e 11. Utilizando (2.26), a máquina D-Markov representada na Figura 4 é gerada.

Tabela 1 – Probabilidades de palavras de comprimento até $\ell = 3$ obtidas a partir de uma sequência binária *S*.

$\ell = 1$	Prob.	$\ell = 2$	Prob.	$\ell = 3$	Prob.
0	0,7	00	0,49	000	0,17
1	0,3	01	0,24	001	0,12
		10	0,25	010	0,12
		11	0,02	011	0,11
				100	0,12
				101	0,12
				110	0,11
				111	0,13

Fonte: O autor.

Figura 4 – Uma máquina D-Markov com D = 2.



Fonte: O autor.

2.7.2 DCGraM

O Algoritmo DCGraM (FRANCH et al., 2020) representa uma evolução dos modelos D-Markov, combinando técnicas de agrupamento de estados e minimização de grafos para gerar uma partição inicial e refiná-la recursivamente.

Para aplicar o algoritmo de clusterização, os *morphs* necessitam ser representados como pontos em um espaço de dimensão $|\Sigma|$. Para um dado estado q, seu *morph* é representado como um ponto $(\Pr(\sigma_1|q), \Pr(\sigma_2|q), \ldots, \Pr(\sigma_{|\Sigma|}|q))$ para $\sigma_1, \sigma_2, \ldots, \sigma_{|\Sigma|} \in \Sigma$ no espaço $|\Sigma|$-dimensional. Dado que os *morphs* são distribuições de probabilidade, $\sum_{\sigma\in\Sigma} \Pr(\sigma|q) = 1, \forall q \in Q$. Com essas restrições, os pontos são representados em um espaço $(|\Sigma| - 1)$-dimensional. Para criar de maneira eficiente uma partição inicial $\mathcal{P}$ dos estados de uma máquina D-Markov (G_D, π_D), um algoritmo de clusterização é utilizado para agrupar estados com *morphs* semelhantes.

A ideia principal do DCGraM é aplicar uma técnica de minimização de grafos, iniciando com a partição $\mathcal{P}$ determinada pelo algoritmo de clusterização, resultando em um grafo reduzido que mantém um comportamento estatístico similar ao modelo D-Markov. Os estados são agrupados com base em seus *morphs*, que possuem probabilidades semelhantes de gerar um símbolo, embora a probabilidade de gerar sequências mais longas possa variar. Este critério é usado para dividir esses *clusters* por meio do algoritmo de minimização de grafos. Uma vez que o algoritmo de minimização de grafos conclui o refinamento da partição inicial, os estados dentro do mesmo *cluster* passam a gerar sequências com probabilidades semelhantes.

O algoritmo DCGraM está descrito no Algoritmo 4, e suas etapas são detalhadas a seguir.

Etapa 1: Criação de Máquina D-Markov

Inicialmente um modelo D-Markov (G_D, π_D) é criado para um D específico, a partir de uma sequência de entrada S.

Etapa 2: Clusterização de estados

Indicando por $\mathcal{V}(q)$ o *morph* do estado q temos, por extensão, que $\mathcal{V}(Q)$ representa o conjunto de *morphs* dos estados em Q. O algoritmo k-*means* aplicado no DCGraM utiliza a métrica Euclidiana $\| x - m(k) \|^2$ para estimar a distância entre os elementos do conjunto de estados e os centroides calculados no k-*means*.

O algoritmo opera inicialmente atribuindo valores iniciais aos centroides $\mathbf{m}^{(k)}$. Em seguida, cada ponto de dados x é associado a um *cluster* j para o qual a distância $\|x - \mathbf{m}^{(j)}\|^2$ seja mínima, para $j \in \{1, \ldots, K\}$. Os valores de $\mathbf{m}^{(k)}$ são então atualizados para a média dos pontos de dados atribuídos a cada *cluster*. Essas etapas de atribuição e atualização são reexecutadas até que nenhuma atribuição seja alterada após uma iteração.

Os estados de G_D são agrupados utilizando a função auxiliar *kmeans*$(K, Q, \mathcal{V}(Q))$. Isso aplica o algoritmo k-*means* em $\mathcal{V}(Q)$ para criar K grupos e particionar o conjunto de estados Q de G_D conforme os rótulos desses grupos. Em outras palavras, os estados que recebem o mesmo rótulo pelo k-*means* são agrupados em uma mesma classe de equivalência. Essa abordagem resulta na partição inicial $\mathcal{P}$ do conjunto de estados de G_D e é utilizada como entrada para um algoritmo de minimização de grafos.

Etapa 3: Aplicação do algoritmo de minimização de grafo

Um algoritmo de minimização de grafos (por exemplo o Moore) é aplicado às classes de equivalência na partição $\mathcal{P}$, a fim de gerar a partição refinada final G_o, na qual cada estado em cada classe de equivalência possui o mesmo contexto à direita e gera sequências com probabilidades similares.

Uma vez obtido G_o, π_o é calculado utilizando a função *averageMorph*. As classes de equivalência de G_o são então transformadas em estados, cujos *morphs* são a média dos *morphs* dos estados pertencentes à mesma classe de equivalência. Por fim, o PFSA reduzido (G_o, π_o) é retornado como a saída do DCGraM.

Algoritmo 4 DCGraM

Entradas: S, D, K
Saída: PFSA (G_o, π_o)
 # Etapa 1: Criação de Máquina D-Markov
$(G_D, \pi_D) \leftarrow$ *dmarkov*(S, D)
 # Etapa 2: Clusterização de estados
$\mathcal{P} \leftarrow$ *kmeans*$(K, Q, \mathcal{V}(Q))$
 # Etapa 3: Aplicação do algoritmo de minimização de grafo
$G_o \leftarrow$ *graphMinimization*$(G_D, \mathcal{P})$
$\pi_o \leftarrow$ *averageMorphs*(G_o)
retorne (G_o, π_o)

2.7.3 VL-DCGraM

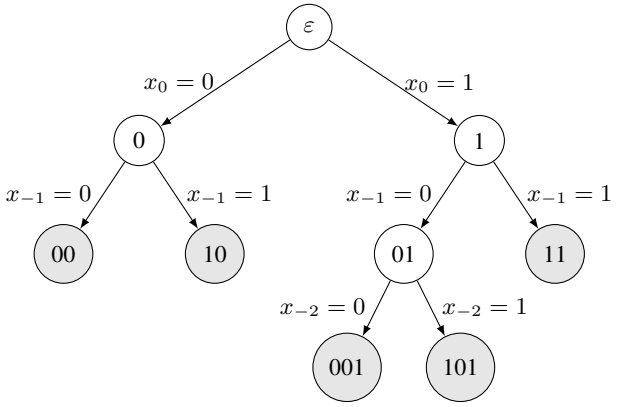
O uso de um D-Markov como PFSA inicial para o algoritmo DCGraM pode ser impraticável para sistemas dinâmicos com grandes requisitos de memória, uma vez que o número de estados do D-Markov cresce exponencialmente com D. Neste contexto, o algortimo VL-DCGraM (SANTOS; CHAVES; PIMENTEL, 2024) surge como uma alternativa para contornar esta limitação. A principal distinção entre VL-DCGraM e DCGraM é a substituição do D-Markov original por uma cadeia de Markov de comprimento variável (VLMC - *Variable length Markov Chain*). O objetivo é construir um PFSA cujos estados são rotulados com palavras de comprimento variável e que apresentam comportamento estatístico equivalente ou melhor que o DCGraM. Assim como o DCGraM, o VL-DCGraM está estruturado em três etapas, diferenciando-se do anterior nas duas primeiras: 1) Construção de um VLMC a partir

da sequência discreta disponível; 2) Aplicação de k-*means* com média ponderada para geração dos centróides; 3) Aplicação de um algoritmo de minimização de estado. Na geração de uma VLMC (Q, Σ, δ, π), na primeira etapa, o algoritmo compreende a implementação de três blocos, descritos à seguir:

- **Bloco I**: A etapa inicial do algoritmo VL-DCGraM consiste na construção do conjunto de estados $Q = \mathcal{C}$ a partir das folhas de uma árvore de contextos CT (BABICH; KELLY; LOMBARDI, 1999), na qual novos nós são gerados recorrentemente utilizando uma travessia em pré-ordem (WEISS, 1998) da árvore, começando pelo nó raiz ε, até que todos os sucessores de um dado nó apresentem o mesmo *morph* ou, na prática, apresentem *morphs* similares, adotando a distância Euclidiana menor que um certo limiar T como critério de similaridade entre *morphs*. Neste caso, esse nó é especificado como uma folha da CT. Os parâmetros necessários para determinar a CT são sua profundidade L e T.

  Denotando o *morph* de um estado q por $\mathcal{V}(q)$ ou $V(w)$, em que w é a *string* que rotula q, temos que a cada novo nó visitado, se a desigualdade $\|\mathcal{V}(\sigma_i w) - \mathcal{V}(\sigma_j w)\| > T$ ($\|\cdot\|$ denota a distância Euclidiana entre dois vetores) for verificada para algum par $\sigma_i, \sigma_j \in \Sigma$, e a profundidade máxima da árvore L não tiver sido alcançada, então $|\Sigma|$ nós sucessores são criados, um para cada elemento em Σ; caso contrário, o nó q é inserido no conjunto $\mathcal{C}$. Este processo é repetido até que não seja mais possível estender os nós da CT. Podemos inferir pela Figura 5 que as palavras 000 e 100 têm o mesmo *morph*, e devido a isso, 00 é uma folha da CT.

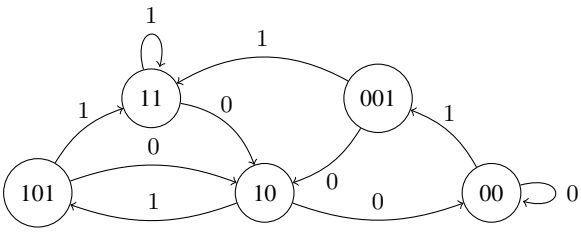
Figura 5 – Exemplo de árvore de contextos com $\Sigma = \{0, 1\}$ e $L = 3$.



Fonte: O autor

- **Bloco II**: Uma vez especificado o conjunto de estados $Q = \mathcal{C}$, o passo seguinte é o calculo da função de transição $\delta(q, \sigma)$ para cada $q \in Q$ e $\sigma \in \Sigma$, sendo definida como o estado em Q com um rótulo igual ao sufixo máximo de $w\sigma$, onde w é o rótulo de q; caso esse estado não exista, um novo estado v é incluído em Q com o rótulo $w\sigma$. O grafo na Figura 6 é construído a partir da CT na Figura 5. Os *morphs* de cada estado são calculados conforme especificado em (2.8).

Figura 6 – Grafo obtido a partir da CT na Figura 5. $D = 3$.



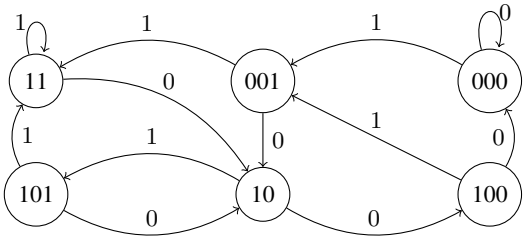
Fonte: O autor.

- **Bloco III**: O PFSA gerado pelo Bloco II passa por um processo de divisão de estados para evitar desvios estatísticos causados pela alta probabilidade de visitação em certos estados. Essa probabilidade pode variar em ordens de magnitude significativas. Um limiar é definido para reger a divisão de estados, sendo proporcional à média do logaritmo da probabilidade estacionária dos estados, dado por

$$l_\mu = 10^{\alpha \sum_{i=1}^{N} \mu_i \log_{10} \mu_i} = 10^{-\alpha h} \tag{2.27}$$

em que $\mu = (\mu_1, \cdots, \mu_{|Q|})$ é o vetor de probabilidade estacionária do PFSA (este vetor especifica, em estado estacionário, a probabilidade de visitar cada estado), α é um parâmetro positivo, e $h = -\sum_{i=1}^{N} \mu_i \log_{10} \mu_i$ é a entropia de μ. Um comprimento máximo para um rótulo de estado, denotado por M, $M \geq L$, também é especificado nesta etapa. Sempre que um estado q_i com rótulo w (com $|w| < M$) tem probabilidade μ_i maior ou igual a l_μ, este estado é substituído no conjunto Q por estados com rótulos $\sigma_i w, \sigma_i \in \Sigma$; se $|w| = M$, então q_i permanece no conjunto Q. A função de transição é calculada para esses novos estados. Este processo é repetido até que a probabilidade de visitar um estado se torne menor que l_μ para todos os estados com rótulos menores que M em comprimento. Para exemplificar esta etapa, suponha que a probabilidade estacionária do estado rotulado com $w = 00$ seja maior que o limiar estabelecido l_μ. Então, este estado é dividido com a inclusão dos estados 000 e 100, conforme mostrado na Figura 7.

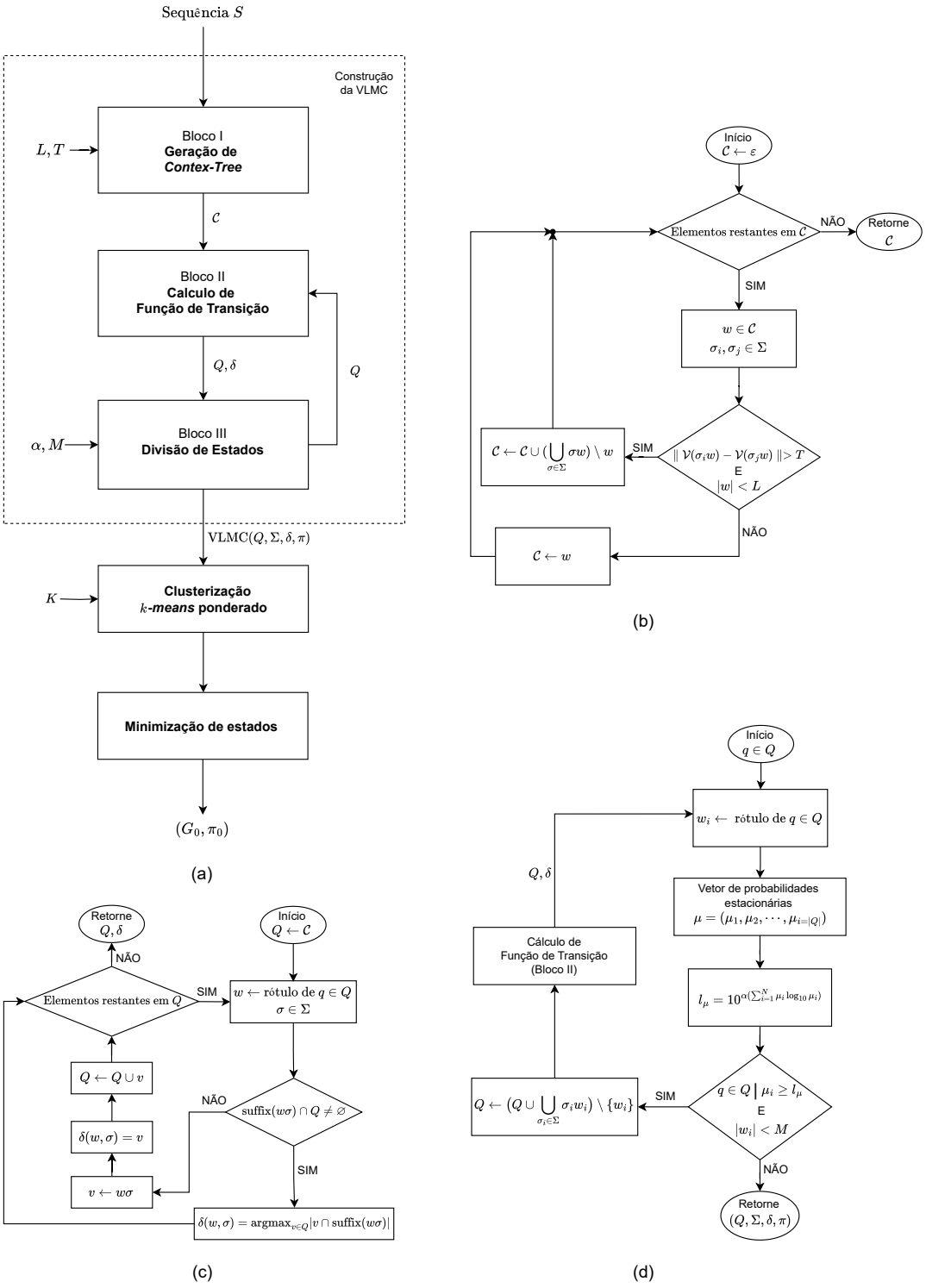
Figura 7 – Grafo obtido após divisão do estado de rótulo 00 no grafo da Figura 6.



Fonte: O autor.

Ao final desta etapa, uma VLMC (Q, Σ, δ, π) é construído e é o PFSA inicial para as próximas etapas. A Figura 8 mostra o fluxograma do algoritmo VL-DCGraM (para um valor fixo dos parâmetros do modelo). Os detalhes dos Blocos I, II, III são mostrados nas Figuras 8(b)-(d). Particularmente, na Figura 8(c), a função $\mathbf{suffix}(\cdot)$ retorna o conjunto de sufixos para o seu argumento. A saída do algoritmo VL-DCGraM gera o PFSA (G_o, π_o).

Figura 8 – Diagramas representativos das etapas do algoritmo VL-DCGraM: Diagrama de blocos do algoritmo VL-DCGraM (a), Fluxograma referente ao Bloco I (b), Fluxograma referente ao Bloco II (c) e Fluxograma referente ao Bloco III (d).



Fonte: O autor.

2.7.4 Algoritmo *State-Splitting by Entropy Reduction*

Conforme apresentado na Seção 2.7.1, em máquinas D-Markov cada um de seus estados é suficientemente descrito por um bloco de símbolos de comprimento finito D, de tal modo que símbolos anteriores aos últimos D símbolos observados não afetam os símbolos subsequentes. Do ponto de vista da modelagem de uma cadeia de símbolos, alguns blocos simbólicos podem apresentar conteúdo de informação mais significativo que outros, o que os torna interessantes para determinação de um conjunto de classes compostas por coleções de sequências de símbolos de comprimentos variados. Uma estratégia para obtenção desses conjuntos é a adotada pelo algoritmo descrito em (MUKHERJEE; RAY, 2014), aqui denominado de *State-Splitting by Entropy Reduction* (S2ER). Nesse algoritmo, parte-se do conjunto $Q = \Sigma$ para D = 1 e realizam-se divisões sucessivas de estados, priorizando-se as que resultam em uma maior redução da taxa de entropia.

De maneira geral, a divisão de estados realizada pelo algoritmo aumenta o número de elementos a fim de obter maior precisão na representação do conteúdo da informação na série temporal. Isto é realizado dividindo-se os estados que efetivamente reduzem a taxa de entropia, concentrando-se assim nos estados críticos (ou seja, aqueles que carregam mais informações).

Com base nas especificações na Seção 2.6, o algoritmo S2ER atua iterativamente estendendo a folha que mais reduz a expressão

$$H(\Gamma) = -\sum_{\omega\in\Gamma} P_\omega \sum_{\sigma\in\Sigma} \pi(\omega,\sigma)\log\pi(\omega,\sigma), \tag{2.28}$$

a qual define-se como a entropia do conjunto de folhas da árvore. Seja Γ^i o conjunto obtido de Γ pela substituição de ω_i pelos correspondentes nós filhos, ou seja, $\Gamma^i = \Gamma\backslash\{\omega_i\} \cup \{\sigma\omega_i : \sigma \in \Sigma\}$ para $\omega_i \in \Gamma$. No algoritmo S2ER, ω_i será substituído por seus filhos na árvore se $H(\Gamma^i) < H(\Gamma^j)$ para qualquer outro $\omega_j \in \Gamma$. Portanto, concluída essa iteração, teremos uma árvore expandida cujo conjunto de folhas é Γ^i.

A máxima redução da taxa de entropia é o critério que rege a seleção do estado a ser dividido pelo algoritmo S2ER. O critério de parada é alcançado quando o limiar mínimo de redução de entropia, η_{split}, para a realização de uma divisão é atingido, ou quando se alcança o número máximo estipulado, N_{max}, de folhas da árvore. As ações implementadas no algoritmo S2ER são exibidas no Algoritmo 5.

Como será abordado nos Capítulos 5 e 6, o S2ER apresenta valores iniciais melhores nos quantificadores de desempenho quando comparado com os PFSAs descritos anteriormente, gerados a partir dos dados de sistemas dinâmicos fornecidos. No entanto, com o aumento no número de estados do PFSA, há uma inflexão na evolução de quantificadores, acarretando em sua saturação. Esse efeito provém da construção da função de transição δ, que não incorpora todos os elementos em Γ na determinação do conjunto de estados Q do PFSA.

Algoritmo 5 S2ER

Entradas: Sequencia simbólica $s_1 s_2 s_3 \cdots$, $s_i \in \Sigma$, η_{split}, N_{max}
Saída: PFSA (Q, Σ, δ, π)
Inicialização: Cria uma máquina 1-Markov $\tilde{Q} \coloneqq \Sigma$
repita
    $Q \coloneqq \tilde{Q}$
    $\tilde{Q} = \arg\min_{Q'} H(\Sigma|Q')$, em que $Q' = (Q \setminus \{q\}) \cup (\sigma q : \sigma \in \Sigma)$ e $q \in Q$
até que $|\tilde{Q}| > N_{\text{max}}$ ou $H(\Sigma|Q) - H(\Sigma|\tilde{Q}) < \eta_{\text{split}}$
para $q \in \tilde{Q}$ e $\sigma \in \Sigma$ **faça**
    se $\delta(q, \sigma)$ não é única **então**
        $\tilde{Q} \coloneqq (\tilde{Q} \setminus \{q\}) \cup (\sigma q : \sigma \in \Sigma)$
retorne $\{Q, \Sigma, \delta, \pi\}$

3 ALGORITMO *PRESERVED STATE-SPLITTING BY ENTROPY REDUCTION*

Neste capitulo é apresentado um algoritmo para geração de PFSAs por redução de entropia visando sanar carências identificadas no algoritmo S2ER.

3.1 ALGORITMO PS2ER

Buscando abordar as limitações encontradas no S2ER, propõe-se neste trabalho o algoritmo denominado *Preserved* S2ER, ou PS2ER, que gera um PFSA essencial (ou seja, que não apresenta estados inalcançáveis), garantindo que cada elemento de Γ constitua um estado do PFSA. Dessa forma, as folhas obtidas pela redução da entropia não são removidas ao se tornarem estados isolados, como pode acontecer com o S2ER. Como resultado, as estatísticas das sequências geradas pelo PFSA reproduzem aquelas da sequência original com maior precisão, resultando em uma melhoria consistente nos quantificadores de desempenho do PFSA em comparação com os demais métodos propostos.

O algoritmo PS2ER baseia-se na construção de uma árvore de contexto (CT) (BÜHLMANN, 2000; BABICH; KELLY; LOMBARDI, 1999) em que o conjunto de folhas Γ determina os contextos de um sistema. Contextos são definidos como as menores sequências de símbolos passados que influenciam a ocorrência de símbolos atuais. Dado que o comprimento máximo do contexto é D, estes são caracterizados por uma função de contexto de ordem D, $c : \Sigma^D \to \Gamma$, que garante que para cada $w \in \Sigma^D$, exista um sufixo de comprimento mínimo, $s \in \Gamma$, que satisfaça $\Pr(\sigma|w) = \Pr(\sigma|s)$, para todo $\sigma \in \Sigma$. Portanto, $c(w) = s$. Pode-se demonstrar pela minimalidade de Γ que $c(\cdot)$ é bem definido. Na construção da CT, novos nós são gerados recursivamente, a partir de um nó raiz, até que as folhas geradas correspondam aos contextos.

A construção do PFSA por meio do algoritmo PS2ER pode ser dividida em três etapas:

- Etapa 1 (Extensão de nós): Determina-se Γ^i, para cada $\omega_i \in \Gamma$, com $|\omega_i| < l_{\max}$, lembrando que Γ^i é obtido de Γ pela substituição de ω_i pelos correspondentes nós filhos σw_i, $\sigma \in \Sigma$. Mantém-se a extensão do nó que gera Γ^i, desde que $H(\Gamma^i) < H(\Gamma^j)$ para qualquer outro $\omega_j \in \Gamma$ e que $\Delta H = H(\Gamma) - H(\Gamma^i) > \eta$, sendo η um limiar associado à redução de entropia durante as extensões de nós (critério de similaridade para obtenção do contexto).

- Etapa 2 (Determinação dos sufixos de um nó folha): Para cada $w_i \in \Gamma$ e $\sigma \in \Sigma$, a função $\text{suffix}_\Gamma(\omega_i\sigma)$ retorna, caso exista, o sufixo de $\omega_i\sigma$ em Γ (deve-se notar que nenhuma palavra em Γ possui sufixo próprio em Γ, o que garante unicidade a essa função). Se não existir, isto é, $\text{suffix}_\Gamma(\omega_i\sigma) = \emptyset$, a folha de rótulo ω_i é estendida, substituindo-a por seus nós filhos em Γ. A função de transição $\delta(\omega_i, \sigma)$ é definida como sendo $\text{suffix}_\Gamma(\omega_i\sigma)$, para $\omega_i \in \Gamma$ e $\sigma \in \Sigma$. Este processo deve ser iterado até que a extensão adicional da árvore não seja mais

possível, considerando um limite máximo para o número de folhas em Γ, definido pelo parâmetro $N_{\max}$.

- Etapa 3 (Construção do PFSA): Uma vez atingido um critério de parada, adota-se $Q = \Gamma$ como o conjunto de estados do PFSA. Para cada $q \in Q$ e $\sigma \in \Sigma$, as funções de transição $\delta(q, \sigma)$ são obtidas na Etapa 2. Assumindo que ω é o rótulo de q, herdado da correspondência de q com as folhas em Γ, então $\delta(q, \sigma)$ é o estado cujo rótulo é o sufixo de $\omega\sigma$ em Γ. O *morph* de q, $\pi(q, \sigma)$, é obtido diretamente da sequência simbólica S. O PFSA é então retornado como (Q, Σ, δ, π).

O algoritmo inicia com $\Gamma = \{\varepsilon\}$ e itera através das Etapas 1 e 2 até que algum critério de parada seja satisfeito. O parâmetro $l_{\max}$ limita a profundidade máxima da árvore. Considera-se que as probabilidades usadas para o cálculo de $H(\Gamma)$ em (2.28) são conhecidas para $0 \leq |w| \leq l_{\max}$. O parâmetro $N_{\max}$ limita o número de estados no PFSA, impactando diretamente a complexidade do algoritmo. Observa-se que na Etapa 1, se a condição $\Delta H < \eta$ for satisfeita para a folha w_i que produz o maior decréscimo de entropia, então essa condição também é satisfeita para os outros elementos em Γ. Desta forma, este conjunto é constituído exclusivamente de contextos e não é mais estendido de acordo com os critérios adotados para identificação de contextos. Isto permite a construção do PFSA na Etapa 3. É importante enfatizar que a Etapa 2 garante que o PFSA construído na Etapa 3 seja um PFSA essencial. A implementação das etapas do algoritmo PS2ER é apresentada no Algoritmo 6.

A fim de ilustrar o processo de extensões da árvore na Figura 11, realizado pelo Algoritmo 6, seja $\Gamma = \{00, 010, 110, 01, 11\}$ o conjunto de folhas da árvore em determinada iteração do algoritmo. Supondo os valores de $H(\Gamma^i)$ para cada $w_i \in \Gamma$: $H(\Gamma^1) = 0{,}505, H(\Gamma^2) = 0{,}500, H(\Gamma^3) = 0{,}506, H(\Gamma^4) = 0{,}508, H(\Gamma^5) = 0{,}509$; sendo $\arg\min_i H(\Gamma^i) = 2$, conclui-se que $w_2 = 010$ deve ser estendida, o que leva ao novo conjunto de folhas $\Gamma = \{00, 0010, 1010, 110, 01, 11\}$. Em seguida, de acordo com a Etapa 2, verifica-se para cada elemento w_i de Γ a existência de uma folha no próprio Γ cujo rótulo é sufixo de $w_i\sigma$, $\sigma \in \Sigma$. Observa-se que a extensão da folha 01 com símbolo $\sigma = 0$ forma a sequência 010, que não possui sufixos em Γ. Isso indica a necessidade de realizar um nova extensão na árvore (folhas 001 e 101 partindo da folha pontilhada 01 na Figura 11), resultando em $\Gamma = \{00, 0010, 1010, 110, 001, 101, 11\}$, que são as folhas na Figura 11. Esse conjunto satisfaz a verificação de sufixos na Etapa 2, o que acarreta na conclusão desta etapa. A Figura 12 ilustra a construção do PFSA a partir do cálculo de $\delta(q, \sigma)$ para todo $q \in Q = \Gamma$ e $\sigma \in \Sigma$ (os morphs de cada estado não são mostrados nesta figura). Um fluxograma indicativo das etapas executadas pelo algoritmo P2SER é mostrado na Figura 9.

Vale notar que, ao se construir um PFSA usando o conjunto $\Gamma = \{00, 0010, 1010, 110, 01, 11\}$ sem considerar a Etapa 2 (essencialmente construindo-o via S2ER), obtem-se o PFSA mostrado na Figura 10. Nesse caso, não há transições de entrada para os estados 1010

Algoritmo 6 PS2ER

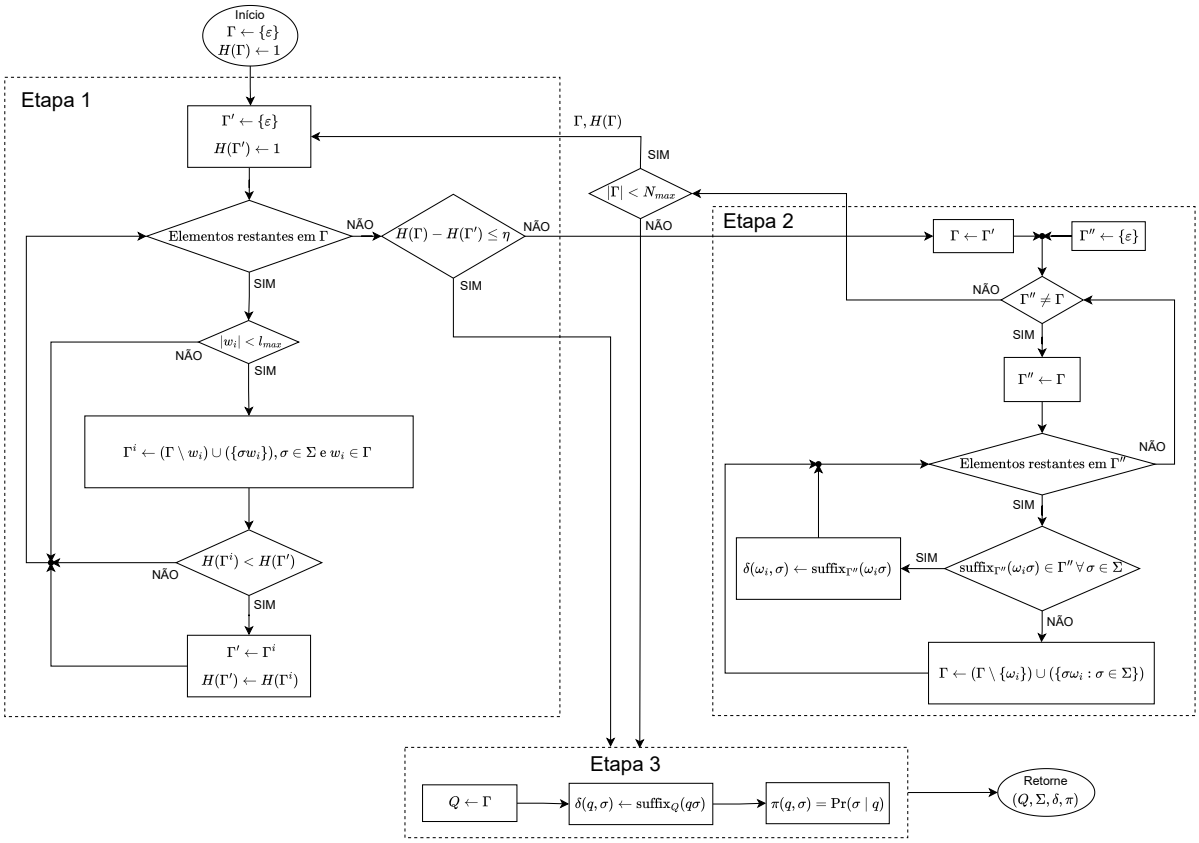
Entradas: $N_{\max}$, $l_{\max}$, η, P_ω, π
Saída: PFSA (Q, Σ, δ, π)
Inicialização: $\Gamma \leftarrow \{\varepsilon\}, H(\Gamma) = 1$
repita
    # Etapa 1: Extensão de nós
    $\Gamma' \leftarrow \{\varepsilon\}$
    $H(\Gamma') = 1$
    para $w_i \in \Gamma$ **faça**
        se $|w_i| < l_{\max}$ **então**
            $\Gamma^i \leftarrow (\Gamma \setminus \{w_i\}) \cup (\{\sigma w_i : \sigma \in \Sigma\})$
            se $H(\Gamma^i) < H(\Gamma')$ **então**
                $\Gamma' \leftarrow \Gamma^i$
                $H(\Gamma') \leftarrow H(\Gamma^i)$
    se $H(\Gamma) - H(\Gamma') \leq \eta$ **então**
        Seguir para a **Etapa 3**
    senão
        # Etapa 2: Identificação de máximos sufixos
        $\Gamma \leftarrow \Gamma'$
        $\Gamma'' \leftarrow \{\varepsilon\}$
        enquanto $\Gamma'' \neq \Gamma$ **faça**
            $\Gamma'' \leftarrow \Gamma$
            para $\omega_i \in \Gamma''$ **faça**
                se $\text{suffix}_{\Gamma''}(\omega_i\sigma) \in \Gamma'' \, \forall \, \sigma \in \Sigma$ **então**
                    $\delta(\omega_i, \sigma) \leftarrow \text{suffix}_{\Gamma''}(\omega_i\sigma)$
                senão
                    $\Gamma \leftarrow (\Gamma \setminus \{\omega_i\}) \cup (\{\sigma\omega_i : \sigma \in \Sigma\})$
até que $|\Gamma| \geq N_{\max}$
 # Etapa 3: Construção do PFSA
$Q \leftarrow \Gamma$
para $q \in Q$ **faça**
    para $\sigma \in \Sigma$ **faça**
        $\delta(q, \sigma) \leftarrow \text{suffix}_Q(q\sigma)$ # Calculado a linha 24
        $\pi(q, \sigma) = \Pr(\sigma \mid q)$ # Fornecido como dado de entrada
retorne $\{Q, \Sigma, \delta, \pi\}$

e 0010 (estes são estados inalcançáveis) e podem ser removidos, o que geralmente leva a uma perda de desempenho em comparação com o PFSA apresentado na Figura 12. Comparações de desempenho entre o PS2ER e demais modelos apresentados nesse trabalho são realizadas nos Capitulos 5 e 6.

3.1.1 Complexidade do Algoritmo PS2ER

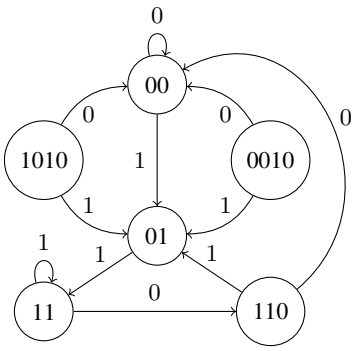
A complexidade do algoritmo PS2ER pode ser estimada por meio da análise de cada uma de suas etapas separadamente. A complexidade da Etapa 1 é dada como a ordem do número de vezes que a entropia $H(\Gamma_i)$ é calculada para extensões de folhas. Assim, dada a geração de

Figura 9 – Fluxograma do algoritmo PS2ER.



Fonte: O autor.

Figura 10 – Função de transição de estados obtida a partir do modelo S2ER.



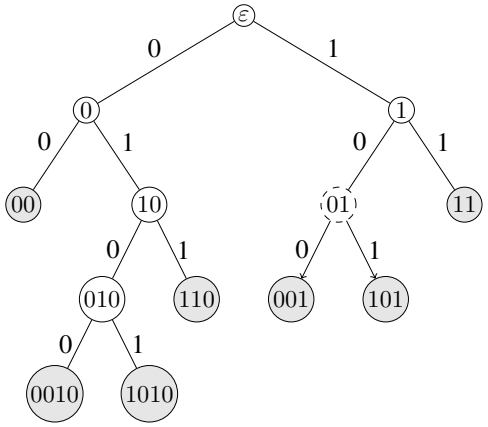
Fonte: O autor.

um máximo de $N_{\max}$ folhas, cada folha é associada a uma sequência sobre o alfabeto Σ, onde o comprimento médio dessas folhas é $N = \log_{|\Sigma|}(N_{\max})$. O número total de nós na CT corresponde ao número de sequências de comprimento máximo N, e é dado pela soma dos termos de uma progressão geométrica até a ordem N

$$\frac{|\Sigma|^{\log_{|\Sigma|}(N_{\max}+1)} - 1}{|\Sigma| - 1}. \tag{3.1}$$

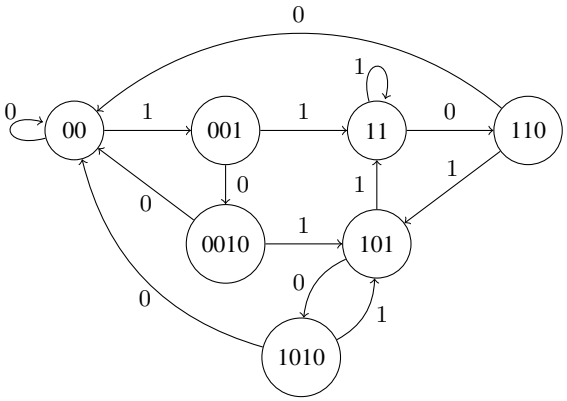
Assim, a complexidade associada a esta etapa é $\mathcal{O}(|\Sigma|^{\log_{|\Sigma|}(N_{\max}+1)})$. A Etapa 2 envolve

Figura 11 – Representação em árvore da geração de estados do PFSA construído pelo Algoritmo 6.



Fonte: O autor.

Figura 12 – Grafo obtido a partir da árvore da Figura 11. $l_{\max} = 4$.



Fonte: O autor.

a busca de um sufixo para cada elemento de Γ. Esta tarefa é linear e envolve percorrer a CT seguindo o caminho do nó raiz ε até a folha cujo rótulo é o sufixo do elemento analisado. O número máximo de nós na profundidade n da CT é $|\Sigma|^n$. Para cada nó, são realizadas $|\Sigma|$ buscas, com cada busca tendo uma complexidade igual à profundidade da árvore, n. Portanto, o número estimado de operações para realizar a Etapa 2 é derivado do número máximo de operações de busca por profundidade da árvore, variando de 1 até sua profundidade média, $N = \log_{|\Sigma|} N_{\max}$, que é $\sum_{n=1}^{N} n|\Sigma|^{n+1}$. Essa soma é expressa como

$$\frac{|\Sigma|^2 \left(N|\Sigma|^{N+1} - (N+1)|\Sigma|^N + 1\right)}{(|\Sigma| - 1)^2} \tag{3.2}$$

que é dominado pelo termo $N|\Sigma|^{N+1}$. Assim, a complexidade da Etapa 2 é considerada como $\mathcal{O}(N|\Sigma|^{N+1}) = \mathcal{O}\big(\log_{|\Sigma|}(N_{\max}) \cdot |\Sigma|^{\log_{|\Sigma|}(N_{\max}+1)}\big)$. A Etapa 3 envolve o retorno do PFSA, associando Q ao conjunto de $N_{\max}$ estados em Γ (no pior cenário), cada um associado a uma função de transição δ (calculada na Etapa 2) e um *morph* π (fornecido como dado de entrada). Portanto, esta etapa não contribui com complexidade adicional ao algoritmo. A complexidade total do algoritmo é determinada pela complexidade dominante da Etapa 2, que é $\mathcal{O}\big(\log_{|\Sigma|}(N_{\max}) \cdot |\Sigma|^{\log_{|\Sigma|}(N_{\max}+1)}\big)$.

3.1.2 Convergência para o conjunto de contexto por redução sucessiva de entropia.

A convergência do algoritmo PS2ER é discutida a seguir. A variação de entropia obtida ao substituir $w_i \in \Gamma$ por seus filhos, definida na Etapa 1 do algoritmo, $\Delta H = H(\Gamma) - H(\Gamma^i)$, pode ser escrita na forma

$$\Delta H = H(\omega) - H(\Phi\omega) \tag{3.3}$$

em que

$$H(\omega) = -P_\omega \sum_{\sigma\in\Sigma} \pi(\omega,\sigma)\log\big(\pi(\omega,\sigma)\big), \tag{3.4}$$

$$H(\Phi\omega) = -\sum_{\phi\in\Sigma} P_{\phi\omega} \sum_{\sigma\in\Sigma} \pi(\phi\omega,\sigma)\log\big(\pi(\phi\omega,\sigma)\big). \tag{3.5}$$

Nas equações (3.4) e (3.5) temos que

$$\pi(\omega,\sigma) = \frac{P_{\omega\sigma}}{P_\omega}, \tag{3.6}$$

e

$$\pi(\phi\omega,\sigma) = \frac{P_{\phi\omega\sigma}}{p_{\phi\omega}} \tag{3.7}$$

respectivamente. Assim

$$H(\omega) = -P_\omega \sum_{\sigma\in\Sigma} \frac{P_{\omega\sigma}}{P_\omega}\log\left(\frac{P_{\omega\sigma}}{P_\omega}\right) = -\sum_{\sigma\in\Sigma} P_{\omega\sigma}\log(P_{\omega\sigma}) + P_\omega\log(P_\omega). \tag{3.8}$$

De forma similar, calcula-se

$$\begin{aligned} H(\Phi\omega) &= -\sum_{\phi\in\Sigma} P_{\phi\omega} \sum_{\sigma\in\Sigma} \frac{P_{\phi\omega\sigma}}{P_{\phi\omega}}\log\left(\frac{P_{\phi\omega\sigma}}{P_{\phi\omega}}\right) \\ &= -\sum_{\phi\in\Sigma}\sum_{\sigma\in\Sigma} P_{\phi\omega\sigma}\Big[\log(P_{\phi\omega\sigma}) - \log(P_{\phi\omega})\Big] \\ &= \sum_{\phi\in\Sigma}\Big(-\sum_{\sigma\in\Sigma} P_{\phi\omega\sigma}\log(P_{\phi\omega\sigma}) + P_{\phi\omega}\log(P_{\phi\omega})\Big). \end{aligned} \tag{3.9}$$

Reescrevendo (3.8) como

$$H(\omega) = \sum_{\phi\in\Sigma}\Big(-\sum_{\sigma\in\Sigma} P_{\phi\omega\sigma}\log(P_{\omega\sigma}) + P_{\phi\omega}\log(P_\omega)\Big) \tag{3.10}$$

ΔH é expresso como a diferença entre (3.9) e (3.10),

$$\begin{aligned} \Delta H &= H(\omega) - H(\Phi\omega) \\ &= \sum_{\phi\in\Sigma}\left(\sum_{\sigma\in\Sigma} P_{\phi\omega\sigma}\log\left(\frac{P_{\phi\omega\sigma}}{P_{\omega\sigma}}\right) - P_{\phi\omega}\log\left(\frac{P_{\phi\omega}}{P_\omega}\right)\right) \\ &= -H(\Phi\Omega|\Omega=\omega,\Sigma) + H(\Phi\Omega|\Omega=\omega) \geq 0 \end{aligned} \tag{3.11}$$

uma vez que o condicionamento não aumenta a entropia (COVER; THOMAS, 2012). Logo, o acréscimo de memória reduz ou mantém a entropia do conjunto de folhas da árvore de contexto.

Atingido o conjunto de contexto, obtém-se $\Delta H = 0$, o que segue do fato de $\pi(\phi\omega, \sigma) = \pi(\omega, \sigma)$, para ω um contexto. Se ω é um contexto, então

$$H(\omega) = H(\Phi\omega) \tag{3.12}$$

uma vez que

$$\begin{aligned} H(\Phi\omega) &= -\sum_{\phi\in\Sigma} P_{\phi\omega} \sum_{\sigma\in\Sigma} \pi(\phi\omega, \sigma) \log(\pi(\phi\omega, \sigma)) \\ &= -\sum_{\phi\in\Sigma} P_{\phi\omega} \sum_{\sigma\in\Sigma} \pi(\omega, \sigma) \log(\pi(\omega, \sigma)) \\ &= \left(-\sum_{\sigma\in\Sigma} \pi(\omega, \sigma) \log(\pi(\omega, \sigma)) \right) \times \sum_{\phi\in\Sigma} P_{\phi\omega} \\ &= -P_{\omega} \sum_{\sigma\in\Sigma} \pi(\omega, \sigma) \log(\pi(\omega, \sigma)) \\ &= H(\omega). \end{aligned} \tag{3.13}$$

Assim, é mostrado que há uma redução monotônica na entropia associada à CT. Também é demonstrado que a expansão de um contexto não resulta em redução de entropia, servindo assim como critério para identificar um conjunto de palavras que, no algoritmo proposto, são interpretadas como um conjunto de contextos.

4 ALGORITMO PARA CONSTRUÇÃO DE MODELOS PFSA VIA CLUSTERIZAÇÃO *FUZZY* C-*MEANS* COM DIVISÃO DE ESTADOS (NDPFSA)

Neste capítulo é apresentado o desenvolvimento de PFSAs utilizando a técnica *fuzzy* c-*means* associada com uma metodologia baseada em minimização de grafos. O algoritmo de agrupamento agrega estados com estatísticas de primeira ordem semelhantes em classes de equivalência. A minimização de grafos, por sua vez, permite identificar aqueles com estatísticas de ordem superior diferentes. Isso orienta a divisão da respectiva classe em subclasses que preservam a estatística de ordem superior dos estados nelas contidos.

4.1 ETAPAS DO ALGORITMO

A seguir são descritas as etapas que compõem o algoritmo proposto de criação de modelos PFSA não determinísticos.

Passo 1: Criação da Máquina D-Markov

Na etapa inicial do algoritmo, um PFSA D-Markov (G_D, π_D) é gerado para um D especificado a partir de uma sequência simbólica finita $S = \sigma_1\sigma_2\sigma_3, \ldots, \sigma_N, \sigma \in \Sigma$, cujos estados q são submetidos ao algoritmo de agrupamento na etapa seguinte.

Passo 2: Clusterização *fuzzy* de estados

A etapa atual é dedicada ao agrupamento dos estados do PFSA D-Markov através do método de clusterização *fuzzy* c-*means*, no qual estados que apresentam *morphs* semelhantes podem ser agrupados em c conjuntos simultaneamente.

O conjunto de *morphs* $\mathcal{V}(Q)$ dos estados D-Markov são os dados x_k a serem submetidos à clusterização *fuzzy* c-*means*. Assim, são calculados os valores de pertinência $F_i(x_k)$ conforme (2.22), relacionando cada estado q aos *clusters* C_i, $i = 1, \ldots, c$; a partir desses valores é construída uma matriz de pertinências M_{pert} de dimensão $n \times i$, em que cada linha n está associada a um estado q e cada coluna i remete a um dos *clusters* C_i.

Uma vez que os valores $F_i(x_k)$ calculados pelo algoritmos *fuzzy* c-*means* são não nulos, cada estado q terá um grau de pertinência a todos os *clusters*, e a magnitude desses resultados armazenados na matriz M_{pert} permite inferir o nível de associação de um estado a cada um dos *clusters*, de tal modo que quanto menor o valor do elemento $M_{\text{pert}_{ni}}$ dessa matriz, menor a associação do n-ésimo elemento com o *cluster* C_i. Assim, para limitar a associação de estados à classes cuja pertinência calculada apresente valor tão reduzido que possa ser desconsiderada, um limiar l_{pert} é adotado de modo que, para o n-ésimo estado ser considerado relevante para a classe C_i, deve-se ter $M_{\text{pert}_{ni}} \geq l_{\text{pert}}$. Isso é feito visando favorecer a redução da quantidade de

transições que partem de um estado do PFSA a ser gerado e estabelecer um compromisso entre a complexidade do modelo e a sua capacidade de modelagem.

Após zerar as entradas de M_{pert} com valor inferior a l_{pert}, os valores nas linhas de M_{pert} devem ser normalizados (dividindo cada valor pelo somatório dos valores na linha) para que sua soma seja unitária. A partir dessa matriz normalizada define-se a inclusão dos estado nos diversos agrupamentos de maneira que, havendo valor diferente de 0 na coluna associada a um *cluster* C_i, o respectivo estado q será incluído neste.

Determinada a ocorrência dos estados em cada um dos *clusters*, a etapa seguinte consiste na identificação de estados equivalentes e subsequente agrupamento destes em *subclusters*, descrita na etapa seguinte.

Exemplo 1

Para ilustrar como o algoritmo NDPFSA funciona, é apresentado um exemplo considerando os estados de uma máquina D-Markov para D = 3 gerada a partir de uma sequência simbólica binária S. A Tabela 2 ilustra os resultados obtidos para $F_i(x_k)$ pela clusterização *fuzzy c-means* adotando-se três classes, $c = 3$, para o conjunto de *morphs* dos 8 estados existentes.

Tabela 2 – Valores de pertinência $F_i(x_k)$ dos estados de máquina D-Markov para D = 3 e $c = 3$, calculados via algoritmo *fuzzy c-means*.

estado q	$F_1(x_k)$	$F_2(x_k)$	$F_3(x_k)$
000	$1{,}6690.10^{-7}$	$4{,}8253.10^{-8}$	$9{,}9999.10^{-1}$
001	$9{,}9970.10^{-1}$	$1{,}7857.10^{-4}$	$1{,}2579.10^{-4}$
010	$9{,}9965.10^{-1}$	$2{,}0320.10^{-4}$	$1{,}4267.10^{-4}$
011	$1{,}3318.10^{-5}$	$9{,}9998.10^{-1}$	$2{,}8589.10^{-6}$
100	$9{,}9834.10^{-1}$	$9{,}0727.10^{-4}$	$7{,}5339.10^{-4}$
101	0,0285	0,9664	0,0051
110	0,0196	0,9768	0,0036
111	0,0460	0,9413	0,0127

Fonte: O autor.

A partir da Tabela 2 construímos a matriz de pertinência

$$M_{\text{pert}} = \begin{bmatrix} 1{,}6690.10^{-7} & 4{,}8253.10^{-8} & 9{,}9999.10^{-1} \\ 9{,}9970.10^{-1} & 1{,}7857.10^{-4} & 1{,}2579.10^{-4} \\ 9{,}9965.10^{-1} & 2{,}0320.10^{-4} & 1{,}4267.10^{-4} \\ 1{,}3318.10^{-5} & 9{,}9998.10^{-1} & 2{,}8589.10^{-6} \\ 9{,}9834.10^{-1} & 9{,}0727.10^{-4} & 7{,}5339.10^{-4} \\ 0{,}0285 & 0{,}9664 & 0{,}0051 \\ 0{,}0196 & 0{,}9768 & 0{,}0036 \\ 0{,}0460 & 0{,}9413 & 0{,}0127 \end{bmatrix},$$

para qual a primeira, segunda e terceira colunas são associadas respectivamente a *clusters* rotulados C_1, C_2 e C_3, enquanto suas linhas estão associadas a cada um dos estados D-Markov na mesma ordem em que são visualizados na Tabela 2.

Observando os valores em M_{pert}, notamos a ocorrência de pertinências com ordem de grandeza inferior a 10^{-2}. A fim de desconsiderar a associação de estados à *clusters* com essa característica, adotamos um limiar $l_{\text{pert}} = 0{,}01$, de forma que os valores de pertinência em uma linha menores que o limiar são anulados. O resultado dessa análise gera a matriz intermediária

$$\begin{bmatrix} 0{,}0 & 0{,}0 & 9{,}9999.10^{-1} \\ 9{,}9970.10^{-1} & 0{,}0 & 0{,}0 \\ 9{,}9965.10^{-1} & 0{,}0 & 0{,}0 \\ 0{,}0 & 9{,}9998.10^{-1} & 0{,}0 \\ 9{,}9834.10^{-1} & 0{,}0 & 0{,}0 \\ 0{,}0285 & 0{,}9664 & 0{,}0 \\ 0{,}0196 & 0{,}9768 & 0{,}0 \\ 0{,}0490 & 0{,}9413 & 0{,}0 \end{bmatrix},$$

cuja normalização das linhas resulta na matriz normalizada, M^*_{pert}:

$$M^*_{\text{pert}} = \begin{bmatrix} 0{,}0 & 0{,}0 & 1{,}0 \\ 1{,}0 & 0{,}0 & 0{,}0 \\ 1{,}0 & 0{,}0 & 0{,}0 \\ 0{,}0 & 1{,}0 & 0{,}0 \\ 1{,}0 & 0{,}0 & 0{,}0 \\ 0{,}0286 & 0{,}9714 & 0{,}0 \\ 0{,}0197 & 0{,}9803 & 0{,}0 \\ 0{,}0495 & 0{,}9505 & 0{,}0 \end{bmatrix}.$$

Uma vez obtida M^*_{pert}, a determinação da ocorrência de um estado nos *clusters* é estabelecida de acordo com a ocorrência de valores diferentes de 0 nas suas colunas. Dessa forma, por exemplo, o estado 000 (associado a primeira linha) apresenta ocorrência apenas no *cluster* C_3, visto que apenas o elemento na terceira coluna dessa linha (representado por $M^*_{\text{pert}_{13}}$) tem valor não nulo. Já com relação ao estado 111 (oitava linha), os elementos $M^*_{\text{pert}_{81}}$ e $M^*_{\text{pert}_{82}}$ são diferentes de 0, indicando a ocorrência desse estado nos *clusters* C_1 e C_2.

Na Tabela 3 são exibidos os *clusters* oriundos da análise matricial acima.

Tabela 3 – *Clusters* gerados de acordo com valores na matriz normalizada M^*_{pert}.

Cluster	Estados agrupados
C_1	001, 010, 100, 101, 110, 111
C_2	011, 101, 110, 111
C_3	000

Fonte: O autor.

Passo 3: Metodologia para divisão de estados (G-Moore)

Como mais uma contribuição deste trabalho, é proposta uma metodologia, denominada G-Moore, que generaliza o conceito de divisão de partições do algoritmo Moore e será adotada na construção do modelo proposto.

Uma vez estabelecidos os *clusters* iniciais na etapa anterior, a etapa atual consiste em identificar estados contidos em uma mesma classe cujas sequências simbólicas associadas aos seus rótulos sejam consideradas equivalentes e agrupá-las em *subclusters*, realizando com isso uma divisão dos *clusters* em que estão inseridas, conforme descrito a seguir.

Define-se um conjunto de destino CD_σ com relação a um estado q como aquele formado pelo rótulo dos *clusters* que contêm os estados correspondentes à transição $\delta(q, \sigma)$ conforme descrito na Seção 2.7.1. Assim, dados dois estados q_1 e q_2 presentes em um *cluster* C_i, diz-se que estes são equivalentes se houver interseção entre seus conjuntos CD_σ com relação a cada símbolo de transição σ. Dessa forma, ocorrendo estados em C_i que não atendam essa condição de equivalência, este deve ser dividido em *subclusters*, cada qual contendo os estados considerados equivalentes.

Exemplo 1 (Continuação)

A partir dos estados D-Markov em cada *cluster* conforme exibido na Tabela 3, construímos os conjuntos de destino CD_σ com relação aos símbolos $\sigma \in \{0, 1\}$, conforme visto na Tabela 4.

Observa-se na Tabela 4 que as transições com símbolo $\sigma = 0$ dos estados 001 e 111 destacados no *cluster* C_1 resultam nos estados 010 e 110 respectivamente, constituindo os respectivos conjuntos de transição CD_0 $\{C_1\}$ e $\{C_1, C_2\}$; e para transição com símbolo $\sigma = 1$, obtemos os respectivos estados 011 e 111 e conjuntos de transição CD_1 $\{C_2\}$ e $\{C_1, C_2\}$. A partir disso, verificamos que ocorre interseção entre seus conjuntos CD_1 e CD_0, indicando equivalência entre os estados analisados, sendo, portanto, incluídos em um *subcluster* de rótulo C_{11}. Ainda no *cluster* C_1 os estados 010, 101 e 110 apresentam conjuntos de destino que atendem à condição de equivalência com os estados 001 e 111, dessa forma sendo também incluídos em C_{11}. O estado 100 destacado apresenta interseção em relação ao seu conjunto CD_1 com os demais estados em seu *cluster*, porém não atende a essa condição com relação ao seu conjunto CD_0, $\{C_3\}$; logo, o *subcluster* C_{12} é gerado contendo esse estado, concluindo com isso a divisão de C_1. Todos os elementos no *cluster* C_2 apresentam interseção simultânea entre seus conjuntos de destino

Tabela 4 – Conjuntos de destino CD_σ com relação a cada *cluster*.

C_1				
q	$\delta(q,0)$	$\delta(q,1)$	CD_0	CD_1
001	010	011	$\boldsymbol{\{C_1\}}$	$\boldsymbol{\{C_2\}}$
010	100	101	$\{C_1\}$	$\{C_1, C_2\}$
100	000	001	$\boldsymbol{\{C_3\}}$	$\boldsymbol{\{C_1\}}$
101	010	011	$\{C_1\}$	$\{C_2\}$
110	100	101	$\{C_1\}$	$\{C_1, C_2\}$
111	110	111	$\boldsymbol{\{C_1, C_2\}}$	$\boldsymbol{\{C_1, C_2\}}$
C_2				
q	$\delta(q,0)$	$\delta(q,1)$	CD_0	CD_1
011	110	111	$\{C_1, C_2\}$	$\{C_1, C_2\}$
101	010	011	$\{C_1\}$	$\{C_2\}$
110	100	101	$\{C_1\}$	$\{C_1, C_2\}$
111	110	111	$\{C_1, C_2\}$	$\{C_1, C_2\}$
C_3				
q	$\delta(q,0)$	$\delta(q,1)$	CD_0	CD_1
000	000	001	$\{C_3\}$	$\{C_1\}$

Fonte: O autor.

relativos ao mesmo símbolo σ, sendo então considerados equivalentes, e por isso C_2 não sofre divisão, bem como o *cluster* C_3 unitário.

Assim, realizadas as divisões possíveis, obtemos os *subclusters* listados na Tabela 7.

Tabela 5 – *Subclusters* gerados na etapa de *splits* do G-Moore.

Clusters	Estados agrupados
C_{11}	001, 010, 101, 110, 111
C_{12}	100
C_2	011, 101, 110, 111
C_3	000

Fonte: O autor.

A etapa atual pode ser repetida iterativamente, um número n_{splits} de vezes, sobre o conjunto atual de *subclusters*, permitindo refinar os conjuntos originais em uma quantidade maior de agrupamentos. Algumas ações devem ser realizadas sobre os *subclusters* gerados a fim de remover possíveis configurações indesejadas e também para calcular o grau de pertinência dos estados D-Markov aos novos conjuntos formados. Essas ações são descritas como segue:

- Ocorrência de *subclusters* duplicados

Na ocorrência de *subclusters* duplicados (agrupamentos de estados com rótulos idênticos) oriundos do processo de divisão das classes, os *subclusters* repetidos são removidos, permanecendo assim um conjunto de *subsclusters* distintos.

- Cálculo da matriz de pertinências de sequências contidas em um *subcluster*

Uma vez estabelecido um conjunto de *subclusters*, deve-se determinar os valores de pertinências dos estados D-Markov a cada um destes. Para isso, determina-se inicialmente quais estados que se encontram presentes em um único agrupamento têm grau de pertinência unitário atribuído a este. Já para estados presentes em mais de uma classe, o cálculo de seu grau de pertinência aos *subclusters* onde se encontra inserido deve ser efetuado fazendo-se uso da seguinte equação, baseada na Equação 2.22 do Algoritmo 3:

$$f_i(x_k) = \begin{cases} \left[\sum_{j=1}^{c_s} \left(\frac{D(x_k, \bar{v}_i)}{D(x_k, \bar{v}_j)} \right)^{\frac{1}{m-1}} \right]^{-1}, (x_k \neq v_i) \\ \\ 1, (x_k = v_i), \end{cases} \tag{4.1}$$

em que c_s é igual ao total de *subclusters* na iteração atual do algoritmo ($c_s = c$ na iteração inicial) nos quais o k-ésimo estado ocorre, $\bar{v}_i$ representa o i-ésimo centroide calculado a partir dos *morphs* x_k dos estados no *subcluster* C_i e $D\left(x_k, \bar{v}_i\right)$ correspondente à distância euclidiana entre x_k e v_i.

De acordo com (4.1), caso algum *subcluster* contenha apenas o k-ésimo estado (ou seja, um *subcluster* unitário), o cálculo de $f_i(x_k)$ para este resulta no valor 1, o que se traduz como pertinência unitária desse estado ao *subcluster* i e implicando na desconsideração de sua ocorrência nos demais, sendo assim adotada pertinência nula e sua remoção dos mesmos.

De maneira similar ao realizado no Passo 2, uma matriz de pertinências M_{pert} com os valores $f_i(x_k)$ obtidos deve ser construída e o limiar de pertinências l_{pert} anteriormente definido é aplicado para obtenção da matriz de pertinências normalizada M^*_{pert}, determinando-se a partir desta a ocorrência dos estados em cada *subcluster*. Os novos agrupamentos devem, em seguida, ser verificados quanto a sua constituição: havendo alterações no total de *subclusters* inicialmente considerados (devido a remoção daqueles repetidos ou exclusão de um estado contido em alguma classe de acordo com o resultado gerado por (4.1)), o cálculo de pertinências é retomado recursivamente até que não hajam novas alterações, chegando dessa forma a um conjunto definitivo de *subclusters*. Estes definirão os estados do modelo proposto, conforme descrito na etapa seguinte.

Exemplo 1 (Continuação)

Seja a Tabela 6 obtida pelo uso do Algoritmo 3 para determinação do grau de pertinência dos estados D-Markov aos agrupamentos formados após aplicação da divisão de *clusters* na etapa atual:

Tabela 6 – Valores de pertinência $f_i(x_k)$ dos estados de máquina D-Markov com relação aos $c_s = 4$ *clusters* calculados via algoritmo *fuzzy c-means*.

estado q	$f_1(x_k)$	$f_2(x_k)$	$f_3(x_k)$	$f_4(x_k)$
000	0,0	0,0	0,0	1,0
001	1,0	0,0	0,0	0,0
010	1,0	0,0	0,0	0,0
011	0,0	0,0	1,0	0,0
100	0,0	1,0	0,0	0,0
101	0,0285	0,0	0,9715	0,0
110	0,0321	0,0	0,9679	0,0
111	0,0012	0,0	0,9988	0,0

Fonte: O autor.

Vê-se na Tabela 6, na qual as colunas 1 a 4 estão associadas aos *clusters* C_{11}, C_{12}, C_2 e C_3, respectivamente, que a ocorrência dos estados 001 e 010 apenas em C_{11} e do estado 011 em C_2 implicam diretamente em pertinência unitária desses elementos em seus respectivos *clusters*. Ao realizar o cálculo de $f_i(x_k)$ para os demais estados, obtem-se pertinência unitária do estado 000 a C_3 e do estado 100 a C_{12} (o que está de acordo com o calculado por (4.1), uma vez que estes agrupamentos são unitários, contendo apenas os referidos estados). Para os estados 101, 110 e 111, presentes nas classes C_{11} e C_2, seus valores de pertinências indicam ocorrência em ambos agrupamentos, porém o estado 111 deve ser removido de C_1 segundo o critério de limiar de pertinência $l_{\text{pert}} = 0{,}01$ anteriormente definido. Dessa forma, constrói-se a matriz de pertinência intermediária

$$M_{\text{pert}} = \begin{bmatrix} 0{,}0 & 0{,}0 & 0{,}0 & 1{,}0 \\ 1{,}0 & 0{,}0 & 0{,}0 & 0{,}0 \\ 1{,}0 & 0{,}0 & 0{,}0 & 0{,}0 \\ 0{,}0 & 0{,}0 & 1{,}0 & 0{,}0 \\ 0{,}0 & 1{,}0 & 0{,}0 & 0{,}0 \\ 0{,}0685 & 0{,}0 & 0{,}9315 & 0{,}0 \\ 0{,}0321 & 0{,}0 & 0{,}9679 & 0{,}0 \\ 0{,}0 & 0{,}0 & 0{,}9988 & 0{,}0 \end{bmatrix},$$

a partir da qual obtem-se a matriz normalizada

$$M^*_{\text{pert}} = \begin{bmatrix} 0{,}0 & 0{,}0 & 0{,}0 & 1{,}0 \\ 1{,}0 & 0{,}0 & 0{,}0 & 0{,}0 \\ 1{,}0 & 0{,}0 & 0{,}0 & 0{,}0 \\ 0{,}0 & 0{,}0 & 1{,}0 & 0{,}0 \\ 0{,}0 & 1{,}0 & 0{,}0 & 0{,}0 \\ 0{,}0685 & 0{,}0 & 0{,}9315 & 0{,}0 \\ 0{,}0321 & 0{,}0 & 0{,}9679 & 0{,}0 \\ 0{,}0 & 0{,}0 & 1{,}0 & 0{,}0 \end{bmatrix},$$

que consequentemente leva à nova configuração dos *clusters* como vista na Tabela 7.

Tabela 7 – *Clusters* reajustados pelas verificações realizadas na etapa atual.

Clusters	Estados agrupados
C_{11}	001, 010, 101, 110
C_{12}	100
C_2	011, 101, 110, 111
C_3	000

Fonte: O autor.

Passo 4: Determinação do modelo NDPFSA

Como etapa final deste algoritmo, cada *subcluster* C_τ ($\tau = 1, \ldots, N_{sc}$), proveniente da abordagem G-Moore, obtido na etapa anterior passa a ser considerado um estado não determinístico q_{ND_τ} do modelo NDPFSA. Os *morphs* podem ser determinados por meio de uma matriz $M_{q_{\text{ND}}}$, elaborada utilizando-se os dados dos *morphs* dos estados D-Markov no agrupamento a ser convertido em estado do modelo. Esses dados são associados aos valores de pertinência da matriz M^*_{pert}, definida após as divisões realizadas, conforme descrito a seguir.

Dado um estado q contido em um *subcluster* C_i, efetua-se o produto, com relação ao símbolo de transição $\sigma \in \Sigma$, entre seu *morph* $\pi(q, \sigma)$ e a linha de M^*_{pert} associada ao estado q_{d_σ} = $\delta(q, \sigma)$, representada por $M^*_{\text{pert}_{q_{d_\sigma}}}$. O resultado desse produto para cada q em C_i é utilizado para calcular o vetor $M^{\sigma_j}_{q_{\text{ND}}}$, empregado para especificar a matriz de transição $M_{q_{\text{ND}}}$ do estado q_{ND} pertencente ao PFSA não determinístico, sendo dada por

$$M_{q_{\text{ND}}} = \begin{bmatrix} M^{\sigma_1}_{q_{\text{ND}}} \\ M^{\sigma_2}_{q_{\text{ND}}} \\ \vdots \\ M^{\sigma_k}_{q_{\text{ND}}} \end{bmatrix}, \ \sigma_1, \sigma_2, \ldots, \sigma_k \in \Sigma. \tag{4.2}$$

As linhas $M^{\sigma_j}_{q_{\text{ND}}}$ podem ser calculadas por uma das duas formas a seguir:

- **Cálculo aritmético de $M^{\sigma_j}_{q_{\text{ND}}}$:** Realiza a média aritmética dos N produtos $\pi(q, \sigma_j) M^*_{\text{pert}_{q_{d_{\sigma_j}}}}$:

$$M^{\sigma_j}_{q_{\text{ND}}} = \frac{\sum_{q \in C_i} \pi(q, \sigma_j) M^*_{\text{pert}_{q_{d_{\sigma_j}}}}}{N}; \tag{4.3}$$

- **Cálculo ponderado de $M^{\sigma_j}_{q_{\text{ND}}}$:** Realiza a média ponderada dos N produtos $\pi(q, \sigma_j) M^*_{\text{pert}_{q_{d_{\sigma_j}}}}$, adotando as probabilidades p_q dos estados q em C_i, calculadas a partir da série simbólica S, como pesos:

$$M^{\sigma_j}_{q_{\text{ND}}} = \frac{\sum_{q \in C_i} \pi(q, \sigma_j) M^*_{\text{pert}_{q_{d_{\sigma_j}}}} p_q}{\sum p_q}. \tag{4.4}$$

Uma vez concluído o cálculo da matriz $M_{q_{\text{ND}}}$, determina-se os *morphs* do estado q_{ND_τ} ao qual está relacionada, como sendo

$$\pi(\sigma_i, q_{\text{ND}_j}) = m_{ij}\ ,\ \ i = 1, ..., |\Sigma|\ ,\ \ j = 1, ..., N_{sc}$$

em que os valores m_{ij} correspondem ao elemento da matriz $M_{q_{\text{ND}}}$ localizado na linha i e coluna j.

Exemplo 1 (Continuação)

Concluindo a geração do modelo NDPFSA, calcula-se as transições dos estados que o compõe, que correspondem aos *clusters* gerados até o momento, que recebem os seguintes rótulos:

$$C_{11} \rightarrow q_{\text{ND}1};\quad C_{12} \rightarrow q_{\text{ND}2};\quad C_2 \rightarrow q_{\text{ND}3};\quad C_3 \rightarrow q_{\text{ND}4}.$$

Conforme descrito no Passo 4, para determinar os *morphs* de cada estado q_{ND_τ} através do cálculo ponderado, dispõe-se dos *morphs* e das probabilidades dos estados D-Markov, ambos calculados a partir da sequência S, presentes na Tabela 8.

Tabela 8 – Probabilidades p_q dos estados da máquina D-Markov calculadas a partir da sequência simbólica S.

q	p_q	$\pi(q, \sigma = 0)$	$\pi(q, \sigma = 1)$
000	0,9431	0,986	0,014
001	0,0155	0,8505	0,1495
010	0,0155	0,8504	0,1496
011	0,0031	0,7364	0,2636
100	0,0155	0,855	0,1444
101	0,0031	0,7537	0,2463
110	0,0031	0,7511	0,2489
111	0,0011	0,7041	0,2959

Fonte: O autor.

De posse da configuração dos estados contidos que em cada *cluster*, dos valores na Tabela 8 e da matriz de pertinências M^*_{pert} definida pelo Passo 3, constrói-se a matriz $M_{q_{\text{ND}}}$ para cada estado do modelo NDPFSA, calculando-se cada uma das linhas $M^{\sigma_j}_{q_{\text{ND}}}$ da matriz $M_{q_{\text{ND}}}$. Para demonstrar as operações efetuadas, na Tabela 9 é exibido o desenvolvimento matemático para obtenção dos *morphs* para o estado $q_{\text{ND}3}$ adotando o cálculo aritmético das matrizes linha $M^{\sigma}_{q_{\text{ND}3}}$.

Tabela 9 – Cálculo dos *morphs* do estado q_{ND_3} do modelo NDPFSA.

$M^{\sigma}_{q_{\text{ND}_3}}$ para $\sigma = 0$		
q	$q_{d_0} = \delta(q, 0)$	$\pi(q,0)M^{*}_{\text{pert}_{q_{d0}}}$
011	110	$0{,}7364.[0{,}0321 \quad 0 \quad 0{,}9679 \quad 0] = [0{,}0236 \quad 0 \quad 0.7128 \quad 0]$
101	010	$0{,}7537.[1 \quad 0 \quad 0 \quad 0] = [0{,}7537 \quad 0 \quad 0 \quad 0]$
110	100	$0{,}7511.[0 \quad 1 \quad 0 \quad 0] = [0 \quad 0{,}7511 \quad 0 \quad 0]$
111	110	$0{,}7041.[0{,}0321 \quad 0 \quad 0{,}9679 \quad 0] = [0{,}0226 \quad 0 \quad 0.6815 \quad 0]$
		$M^{0}_{q_{\text{ND}_3}} = \sum_{q \in C_i} \frac{\pi(q,0)M^{*}_{\text{pert}_{q_{d0}}}}{N} = [0{,}1999 \quad 0{,}1878 \quad 0{,}3486 \quad 0]$
$M^{\sigma}_{q_{\text{ND}_3}}$ para $\sigma = 1$		
q	$q_{d_1} = \delta(q, 1)$	$\pi(q,1)M^{*}_{\text{pert}_{q_{d1}}}$
011	111	$0{,}2636.[0 \quad 0 \quad 1 \quad 0] = [0 \quad 0 \quad 0{,}2636 \quad 0]$
101	011	$0{,}2463.[0 \quad 0 \quad 1 \quad 0] = [0 \quad 0 \quad 0.2463 \quad 0]$
110	101	$0{,}2489.[0{,}0685 \quad 0 \quad 0{,}9315 \quad 0] = [0{,}0170 \quad 0 \quad 0.2319 \quad 0]$
111	111	$0{,}2959.[0 \quad 0 \quad 1 \quad 0] = [0 \quad 0 \quad 0{,}2959 \quad 0]$
		$M^{1}_{q_{\text{ND}_3}} = \sum_{q \in C_i} \frac{\pi(q,1)M^{*}_{\text{pert}_{q_{d1}}}}{N} = [0{,}0043 \quad 0 \quad 0{,}2594 \quad 0]$

Fonte: O autor.

Segue dos valores destacados na Tabela 9 que os *morphs* do estado q_{ND_3} são dados pela matriz

$$M_{q_{\text{ND}_3}} = \begin{bmatrix} M^{0}_{q_{\text{ND}_3}} \\ M^{1}_{q_{\text{ND}_3}} \end{bmatrix} = \begin{bmatrix} 0{,}1999 & 0{,}1878 & 0{,}3486 & 0 \\ 0{,}0043 & 0 & 0{,}2594 & 0 \end{bmatrix},$$

que indicam os valores das transições a partir desse estado da seguinte maneira:

- para o estado q_{ND_1} com símbolo 0 e probabilidade 0,1999 (elemento m_{11});
- para o estado q_{ND_2} com símbolo 0 e probabilidade 0,1878 (elemento m_{12});
- para o próprio estado q_{ND_3} com símbolo 0 e probabilidade 0,3486 (elemento m_{13});
- para o estado q_{ND_1} com símbolo 1 e probabilidade 0,0043 (elemento m_{21});
- para o próprio estado q_{ND_3} com símbolo 1 e probabilidade 0,2594 (elemento m_{23}).

É possível notar a soma das probabilidade das transições (elementos m_{ij} da matriz $M_{q_{\text{ND}_3}}$) tem soma unitária, tal como se espera de um estado de um modelo PFSA. Além disso, ve-se que o estado q_{ND_3} apresenta transições com mesmo símbolo σ para mais de um estado, o que caracteriza o modelo NDPFSA gerado como não-determinístico.

Analogamente aos cálculos realizados até aqui, as matrizes $M_{q_{\text{ND}}}$ dos demais estados do modelo são:

$$M_{q_{\text{ND}_1}} = \begin{bmatrix} 0{,}4011 & 0{,}4000 & 0 & 0 \\ 0{,}0070 & 0 & 0{,}1919 & 0 \end{bmatrix};$$

$$M_{q_{\text{ND}_2}} = \begin{bmatrix} 0{,}0 & 0{,}0 & 0{,}0 & 0{,}8556 \\ 0{,}1444 & 0 & 0{,}0 & 0 \end{bmatrix};$$

$$M_{q_{\text{ND}_4}} = \begin{bmatrix} 0{,}0 & 0{,}0 & 0, & 0{,}986 \\ 0{,}0014 & 0 & 0{,}0 & 0 \end{bmatrix}.$$

O algoritmo para geração do modelo proposto é exibido no Algoritmo 7.

Algoritmo 7 NDPFSA

Entradas: $S, D, c, l_{\text{pert}}, n_{\text{splits}}$
Saída: PFSA $(Q_{\text{NDPFSA}}, \pi_{\text{NDPFSA}})$
 # Etapa 1: Criação de Máquina D-Markov
$(G_D, \pi_D) \leftarrow$ *dmarkov*(S, D)
 # Etapa 2: Clusterização *fuzzy* de estados
$M_{\text{pert}} \leftarrow$ *Fuzzy c-means*$(c, \mathcal{V}(Q))$
$M^*_{\text{pert}} \leftarrow normalize(M_{\text{pert}}, l_{\text{pert}})$
para todo $i = 1, ..., c$ **faça**
    para todo $j = 1, ..., |Q|$ **faça**
        se $M^*_{\text{pert}_{ki}} \neq 0.0$ **então**
            $\{C_i\} \leftarrow q_{M^*_{\text{pert}}ki}$
 # Etapa 3: Realização de divisão de estados via G-Moore
$\{C_i\}_{splits} \leftarrow G\text{-}Moore(\text{CD}_\sigma(\{C_i\}), n_{\text{splits}})\ \forall \sigma \in \Sigma$
 # Etapa 4: Determinação do modelo NDPFSA
$Q_{\text{NDPFSA}} \leftarrow q_{\text{ND}}(\{C_i\}_{\text{splits}})$
$\pi_{\text{NDPFSA}} \leftarrow \{M_{q_{\text{ND}}}\}$
retorne $(Q_{\text{NDPFSA}}, \pi_{\text{NDPFSA}})$

5 ANÁLISE DE DESEMPENHO

No atual capítulo, os algoritmos propostos são testados quanto a sua eficiência para construção de PFSAs por meio de alguns exemplos de sistemas dinâmicos, utilizados para realizar uma comparação dos resultados obtidos pelos modelos D-Markov, DCGraM e VLDCGraM, NDPFSA, S2ER e PS2ER.

5.1 SISTEMAS ANALISADOS

Nesta seção são apresentados os cinco tipos de sistemas dinâmicos a serem modelados. São eles: Conjunto de dados térmicos MOSFET, sistema MaFaulDa, banco de dados CWRU, sistema caótico Mapa de Hénon e oscilador Duffing.

5.1.1 Conjunto de dados de envelhecimento por sobrecarga térmica MOSFET

Este conjunto de dados, contidos em repositório da NASA (CELAYA et al., 2012a) relata experimentos executados até a ocorrência de falha em capacitores MOSFETs de potência sob sobrecarga térmica. Os experimentos são detalhados em (CELAYA et al., 2012b). Neste estudo de caso, a série temporal no arquivo 'Test_20_run_2.mat' (disponível no repositório citado) é quantizada com um alfabeto binário e a sequência binária é usada para construir máquinas D-Markov e DCGraM com K = 5 para D variando de 3 a 5. Máquinas NDPFSA com c variando igualmente com D e submetidas a 2 *splits* G-Moore. Os modelos S2ER e PS2ER foram gerados adotando os parâmetros $\eta = 10^{-6}$ e N_{max} entre 7 e 12.

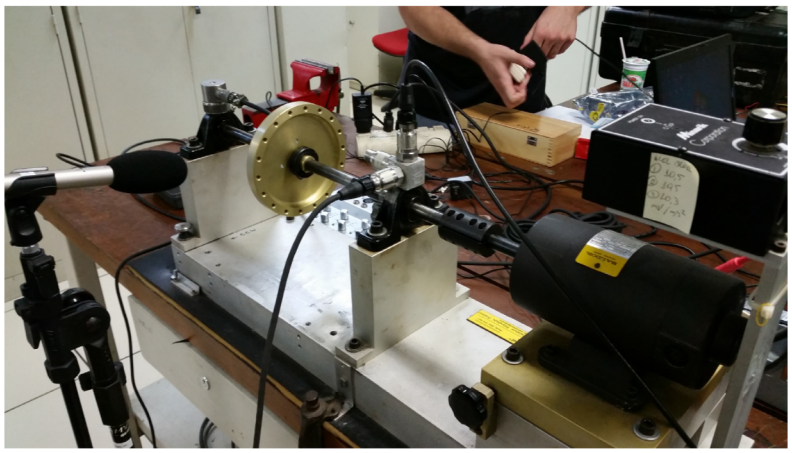
5.1.2 *Machinery Failure Database* (MaFaulDa)

Este banco de dados é composto de séries temporais multivariadas obtidas a partir de leituras realizadas por sensores em um simulador de falhas de alinhamento-equilíbrio-vibração em máquinas (MFS, *Machinery Fault Simulator*) da SpectraQuest (SpectraQuest Inc., 2021). Este equipamento emula a dinâmica de motores com dois mancais de apoio ao eixo, o que permite a análise de múltiplas falhas, como desalinhamento de eixos, desbalanceamento de massa e problemas de mancais. A configuração experimental usada neste trabalho é ilustrada na Figura 13.

O sistema é monitorado por dois conjuntos distintos (um para cada rolamento) de três acelerômetros (nas direções axial, radial e tangencial), um tacômetro (para medir a frequência de rotação do sistema) e um microfone (para captar o som durante a operação de sistema). Durante o procedimento de aquisição de sinal, diversos tipos de falhas foram impostas ao sistema. Algumas dessas falhas são descritas abaixo:

· **Operação normal**: esta classe representa o sistema operando em condições normais, ou seja, sem nenhuma falha. Cenários distintos foram simulados, cada um com uma velocidade

Figura 13 – Configuração experimental usada para produzir o banco de dados MaFaulDa.



fonte: (MARINS et al., 2018).

de rotação fixa na faixa de 737 rpm a 3686 rpm com passos de aproximadamente 60 rpm.

· **Desalinhamento paralelo horizontal**: este tipo de falha foi induzido no MFS deslocando o eixo do motor horizontalmente de 0,5 mm, 1,0 mm, 1,5 mm e 2,0 mm e utilizando a mesma faixa de frequência de rotação que na operação normal para cada deslocamento horizontal.

· **Desalinhamento paralelo vertical**: esta falha foi induzida no MFS deslocando o eixo do motor verticalmente de 0,51 mm, 0,63 mm, 1,27 mm, 1,4 mm, 1,78 mm e 1,9 mm e utilizando a mesma faixa de frequência de rotação que na operação normal para cada deslocamento vertical.

· **Desbalanceamento**: para simular diferentes graus de operação desbalanceada, foram acoplados ao rotor distintos valores de carga de 6 g, 10 g, 15 g, 20 g, 25 g, 30 g e 35 g. Para cada valor de carga abaixo de 30 g, a frequência de rotação assumida foi igual aos valores empregados no caso de operação normal. Para cargas iguais ou superiores a 30 g, entretanto, a vibração resultante inviabiliza que o sistema atinja frequências de rotação acima de 3300 rpm, limitando o número de frequências de rotação distintas nesses casos.

5.1.3 *Case Western Reserve University Dataset* (CWRU)

O terceiro sistema utilizado para analise de quantificadores corresponde ao banco de dados associado à testes de vibração em motores elétricos quando acoplados à rolamentos normais e defeituosos, disponibilizado pelo *Bearing Data Center* da *Case Western Reserve University* (CWRU, 2023). Os rolamentos defeituosos apresentam falhas de usinagem, variando entre 0,007 e 0,028 polegadas de diâmetro, em três diferentes pontos de sua estrutura física. Esses elementos defeituosos são acoplados ao motor separadamente em posições correspondentes às extremidades de acionamento (*Drive End*) e ventilação (*Fan End*) da máquina elétrica e os dados de vibração são registrados à taxas de amostragem 12 kHz e 48 kHz para cargas do motor entre 0 e 3 cavalos de potência (velocidades entre de 1797 a 1720 RPM), gerando 161 cenários de operação, conforme descrito em (SMITH; RANDALL, 2015). A sequência temporal empregada

refere-se aos dados adquiridos a uma taxa de 12kHz na posição *Drive-End* durante o modo de operação normal, submetida a uma decimação adotando um fator $Z = 2$, na qual $Z - 1$ amostras da série temporal são removidas entre duas que são mantidas.

5.1.4 Mapa de Hénon

O Mapa de Hénon (HÉNON, 2004) é um sistema dinâmico de tempo discreto para o qual, dado um ponto (x_n, y_n) em um plano, sua posição é mapeada pela equação

$$x_{n+1} = f(x_n) = \begin{pmatrix} x_{n+1} \\ y_{n+1} \end{pmatrix} = \begin{pmatrix} y_n + 1 - ax_n^2 \\ bx_n \end{pmatrix} \tag{5.1}$$

em que a e b são parâmetros fixos. Para determinados valores desses parâmetros, o sistema apresenta comportamento caótico, como para o caso clássico em que $a = 1{,}4$ e $b = 0{,}3$, os quais foram adotados neste trabalho.

5.1.5 Oscilador Harmônico Duffing

O último sistema analisado é o oscilador harmônico Duffing (ABELL; BRASELTON, 2018), descrito pela equação

$$\frac{d^2y(t)}{dt} + \beta\frac{dy(t)}{dt} + y(t) + by^3(t) = A\cos(\Omega t) \tag{5.2}$$

em que β, b, A e Ω são constantes positivas tais que, dependendo dos valores adotados, as soluções da equação podem apresentar comportamentos distintos. A série é formada para uma resolução temporal de 10^{-1} associada a uma decimação por um fator $Z = 8$ e adotando os valores experimentalmente escolhidos β = 0,1, b = 0,25, A = 2,5 e Ω = 2.

5.2 QUANTIFICADORES DE DESEMPENHO

Nesta seção são apresentados quantificadores adotados para analisar o desempenho dos PFSAs gerados pelos algoritmos considerados neste trabalho. Os quantificadores são: a entropia condicional, que permite o cálculo da taxa de entropia, valor que nos permite ter uma percepção da memória do sistema; a divergência Kullback-Leibler, que aponta um comparativo entre probabilidades de sequências geradas a partir do modelo e do sistema dinâmico.

5.2.1 Taxa de Entropia de ℓ-ésima ordem

Seja $\{X_k\}_{k=1}^{\infty}$ um processo aleatório discreto sobre Σ. Sua taxa de entropia é definida como (COVER; THOMAS, 2012):

$$h \triangleq \lim_{k\to\infty} H(X_k|X_1X_2\ldots X_{k-1}) = -\lim_{k\to\infty}\sum_{x\in\Sigma^k} \Pr(x)\log_{10}\Pr(x_k|x_1x_2\ldots x_{k-1}). \tag{5.3}$$

Para um processo estacionário, a entropia condicional $H(X_k|X_1 \dots X_{k-1})$ é não-crescente em k e converge para h conforme k tende a infinito (COVER; THOMAS, 2012). Como não é viável calcular (5.3), a entropia condicional de ℓ-ésima ordem é utilizada, se definida como

$$h_\ell \triangleq H(X_\ell|X_1X_2 \dots X_{\ell-1}), \tag{5.4}$$

que mede a incerteza de uma variável aleatória X_ℓ, considerando as $\ell - 1$ amostras anteriores. A comparação de h_ℓ de um PFSA gerado a partir da sequência proveniente do sistema original é útil para testar se o primeiro modelo captura corretamente a memória do segundo.

5.2.2 Divergência Kullback-Leibler de ℓ-ésima ordem

Para fins de comparação dos algoritmos, considere duas sequências S_1 e S_2 sobre um alfabeto comum Σ. Estas podem ser a sequência original S ou uma sequência gerada por um PFSA. Seja $\omega \in \Sigma^\ell$ uma palavra de comprimento ℓ e $P_1(\omega)$ e $P_2(\omega)$ sejam as probabilidades de ocorrência de ω em S_1 e S_2, respectivamente. Para um dado ℓ, a divergência de Kullback-Leibler de ℓ-ésia ordem entre S_1 e S_2, denotada por $D_\ell(S_1||S_2)$, é definida como

$$D_\ell(S_1||S_2) = \sum_{\omega \in \Sigma^\ell} P_1(\omega) \log\Big(\frac{P_1(\omega)}{P_2(\omega)}\Big). \tag{5.5}$$

Embora tecnicamente não seja uma distância, pois não obedece à desigualdade triangular nem é comutativa, a divergência de Kullback-Leibler é útil para dar uma ideia de quão semelhantes são duas distribuições. Um valor de divergência reduzido indica que o PFSA gera sequências cujas palavras, até um dado comprimento L, possuem distribuição de probabilidade próxima a da sequência original, o que pode ser interpretado como uma similaridade entre essas, sendo uma boa estimativa para o sistema original.

5.2.3 Resultados

Modelos D-Markov, DCGraM, VL-DCGraM são gerados para valores de D e L variando tipicamente entre 4 a 9 (podendo os valores limites variar dependendo do sistema). O DCGraM e o VL-DCGraM usam $K = 5$ na etapa de clusterização, um valor típico usado em (FRANCH et al., 2020) e (SANTOS; CHAVES; PIMENTEL, 2024). Os outros parâmetros do VL-DCGraM são otimizados para gerar os melhores resultados, sendo $T = 10^{-4}$ e $M = L + 1$ para Duffing e CWRU, $T = 10^{-3}$ para Mapa de Hénon ($M = L + 2$), MOSFET ($M = 5$) e MaFaulDa ($M = L + 1$), além de $\alpha = 2$ para todos os casos. Modelos NDPFSA adotam $l_{pert} = 10^{-2}$ e valores de c similares aos valores D com número de *splits* G-Moore ajustados para obter os melhores resultados a cada sistema modelado. Os modelos S2ER e PS2ER são gerados assumindo $\eta = 10^{-6}$ e diferentes valores de $N_{\max}$; além disso, os comprimentos máximos de palavra $l_{\max}$ adotados correspondem ao D máximo do D-Markov no correspondente sistema. Todos os modelos e dados gerados foram implementados utilizando-se códigos computacionais em linguagem Python (versão 3.7).

Curvas de h_ℓ, para $\ell = 10$, versus o número de estados de cada PFSA são mostradas na Fig. 14 para cada um dos sistemas apresentados anteriormente. A linha horizontal mostrada nestas figuras corresponde ao valor de h_{10} obtido com a série simbólica e serve como um valor de referência de desempenho. Observa-se que o PFSA gerado com o algoritmo PS2ER atinge o valor de entropia h_{10} dos sistemas original Duffing, CWRU e MaFaulDa com menor número de estados em comparação aos demais modelos. Já para os sistemas MOSFET e Mapa de Hénon, os melhores resultados são obtidos pelos modelos NDPFSA. A Fig. 15 exibe D_ℓ versus o número de estados para $\ell = 10$. Observa-se que o PFSA gerado pelo PS2ER se destaca para o caso dos sistemas Duffing, CWRU e MaFaulDa, apresentando o menor D_{10} para um determinado número de estados, enquanto que, para os sistema MOSFET e Mapa de Hénon, os melhores resultados são obtidos pelos modelos gerados pelo NDPFSA.

A Tabela 10 ilustra em quais sistemas cada um dos modelos abordados no trabalho foram superiores com relação aos quantificadores de desempenho utilizados.

Tabela 10 – Indicação do modelo com melhores quantificadores de desempenho, h_{10} e D_{10}, para os cinco sistemas dinâmicos apresentados.

Sistema / Modelo	MOSFET	Hénon	Duffing	CWRU	MaFaulDa
D-Markov					
DCGraM					
VL-DCGraM					
NDPFSA	X	X			
S2ER					
PS2ER			X	X	X

Fonte: O autor.

Com base no que é apresentado na Tabela 10, infere-se a superioridade dos experimentos NDPFSA e PS2ER, tendo o modelo NDPFSA obtido os melhores desempenhos nos conjuntos de dados MOSFET e Mapa de Hénon, enquanto o modelo PS2ER superou todos os demais modelos nos sistema Duffing, CWRU e MaFaulDa, demonstrando uma performance superior na maioria dos casos analisados.

Dado que o PS2ER mostrou-se superior em 3 dos 5 conjuntos de dados disponíveis, ele foi o PFSA escolhido para realizar as análises de detecção de falhas no Capitulo 6. Esta escolha fundamenta-se na sua eficácia e adaptabilidade observada na tarefa de modelagem, o que sugere um potencial elevado para obter resultados positivos também no contexto de detecções de anomalias em sistemas dinâmicos.

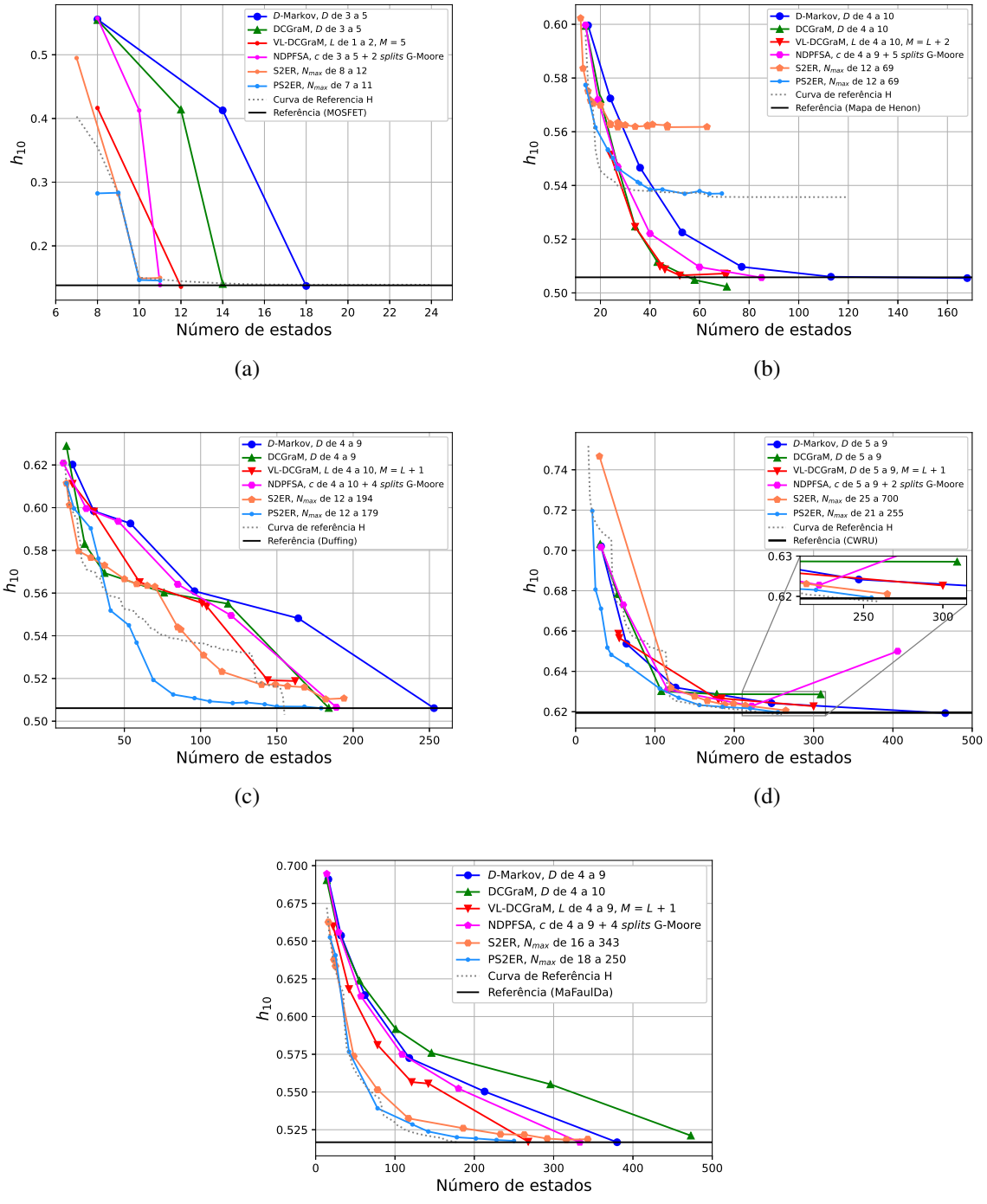


Figura 14 – Curvas h_{10} dos modelos D-Markov, DCGraM, VL-DCGraM, NDPFSA, S2ER e PS2ER calculadas a partir dos sistemas analisados: MOSFET (a), Mapa de Hénon (b), Duffing (c), CWRU (d) e MaFaulDa (e).

Fonte: O autor.

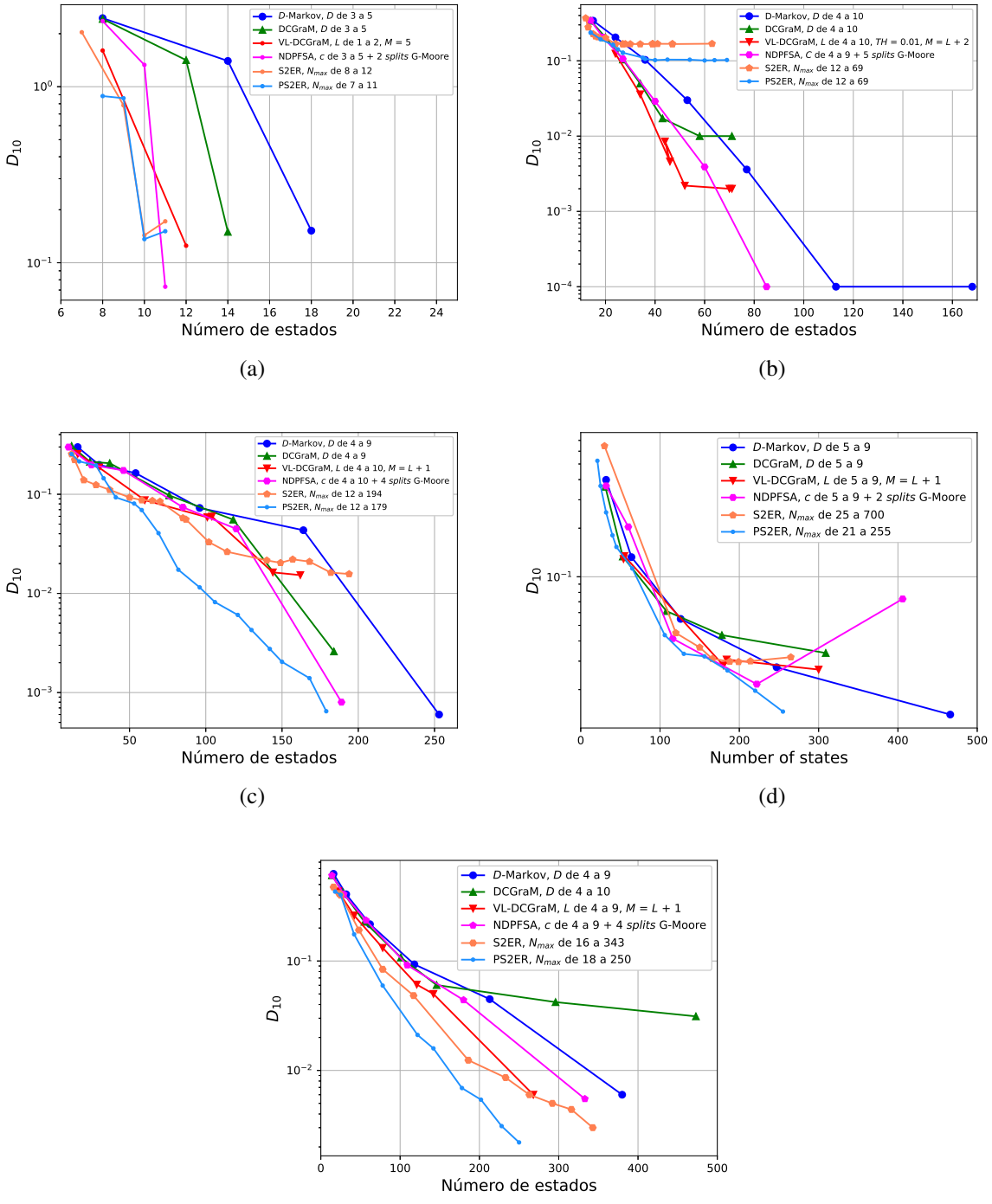


Figura 15 – Curvas D_{10} dos modelos D-Markov, DCGraM, VL-DCGraM, NDPFSA, S2ER e PS2ER calculadas a partir dos sistemas analisados: MOSFET (a), Mapa de Hénon (b), Duffing (c), CWRU (d) e MaFaulDa (e).

Fonte: O autor.

6 ANÁLISES DE DETECÇÕES DE ANOMALIAS

Neste capítulo são apresentados resultados de comparações de analogias adotadas para detecção de características anômalas presente em séries simbólicas geradas a partir das séries temporais de um sistema dinâmico, a fim de demostrar a capacidade dos modelos analisados nesse trabalho de permitirem a eficiente detecção de tais anomalias.

6.1 CURVA CARACTERÍSTICA DE OPERAÇÃO DO RECEPTOR

De maneira simplificada, uma curva característica de operação do receptor (ROC, *Receiver Operating Characteristic Curve*) é definida como um gráfico de sensibilidade *versus* especificidade de um teste analítico (MANDREKAR, 2010), para o qual os diferentes pontos na curva correspondem a diferentes limiares de decisão adotados para determinar se os resultados do teste são positivos ou negativos. Por meio dessa curva é possível averiguar o desempenho de um sistema classificador binário através da variação de um limiar de decisão.

Os pontos que compõem a curva são calculados a partir de dois parâmetros:

- Taxa de verdadeiros positivos (TPR, *True Positive Rate*): indica a relação entre pontos considerados como positivos verdadeiros e os pontos positivos totais:

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}},$$

  onde TP e FN representam os pontos correspondentes a positivos verdadeiros e negativos falsos, respectivamente;

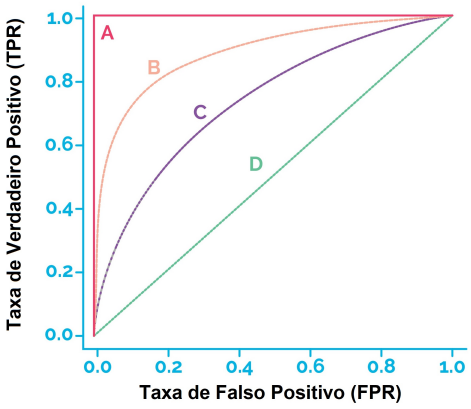
- Taxa de falsos positivos (FPR, *False Positive Rate*): indica a relação entre pontos considerados como positivos falsos e os pontos negativos totais:

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}},$$

  onde FP e TN se referem aos pontos correspondentes a positivos falsos e negativos verdadeiros, respectivamente.

Os dados de TPR e FPR são associados respectivamente aos eixos das abscissas e ordenadas da curva ROC, permitindo a elaboração das curvas exibidas na Figura 16.

Figura 16 – Exemplo de curvas ROC.



Fonte: O autor.

O melhor método de previsão possível produziria uma curva como a representada por A na Figura 16 , apresentando um ponto no canto superior esquerdo do plano, isto é, o ponto com coordenadas (0,1), indicando 100 % de sensibilidade (isto é, ausência de falsos negativos) e 100 % de especificidade (isto é, ausência de falsos positivos). A linha D ao longo da diagonal do plano, denominada linha de não discriminação, divide-o de tal maneira que curvas acima desta (por exemplo B e C) representam bons resultados de classificação, enquanto que curvas abaixo representam maus resultados classificatórios.

A área delimitada sob a curva ROC (AUC, *Area Under Curve*), fornece uma maneira eficaz de expressar a precisão da classificação associada. Seu cálculo assume valores de 0 a 1, onde resultados próximos de 0 indicam imprecisão da classificação associada, enquanto valores próximos de 1 indicam precisão elevada.

6.2 METODOLOGIAS PARA GERAÇÃO DE CURVAS ROC

Três metodologias são comparadas quanto à capacidade de detecção de falhas a partir das series temporais geradas pelas simulações no sistema MaFaulDa. Através dos dados obtidos em cada metodologia, a respectiva curva ROC é gerada a fim de identificar qual delas apresenta melhor desempenho quanto a identificação de uma dada falha.

A primeira metodologia (GHALYAN; RAY, 2021) faz uso dos autovetores associados a máquinas D-Markov construídas a partir de N janelas de símbolos de tamanho w. Esse tamanho é definido segundo um algoritmo que estipula seu valor ótimo e, a partir disso, fornece um valor de limiar para a classificação de séries simbólicas.

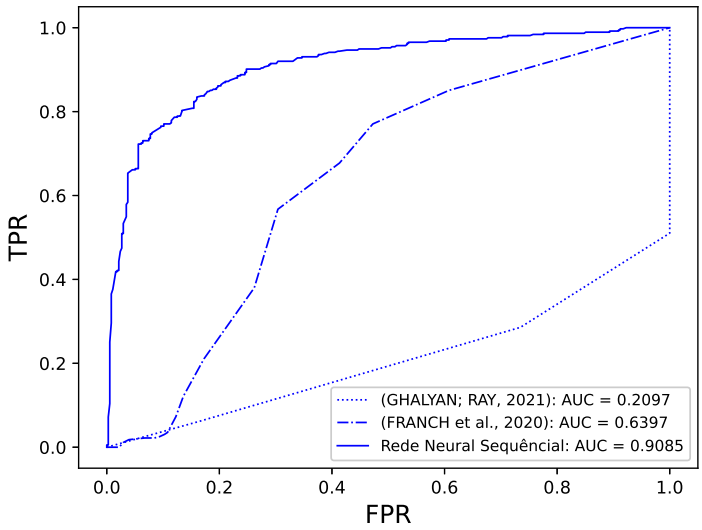
A segunda metodologia (FRANCH et al., 2020) adota a distância euclidiana entre dois conjuntos de valores gerados a partir de um mesmo PFSA: o vetor de ocupação teórico do PFSA, determinado a partir da serie temporal associada à condição normal, e a frequência relativa de visitação dos estados deste PFSA, obtida através de uma caminhada pelo PFSA,

definida pelos símbolos de uma série simbólica associada à condição anômala. O divergente de Kullback-Leibler entre esses dois resultados é calculado, determinando-se um limiar de decisão que, se ultrapassado, implica na detecção de uma anomalia.

Por fim, a terceira metodologia é uma proposta desse trabalho, na qual é construída uma rede neural sequencial cujos dados de treinamento são vetores de ocupação dos estados de um PFSA calculados a partir de janelas simbólicas provenientes de séries normais e anômalas. Esta rede é configurada com uma única camada interna, função de ativação sigmoide e saída única (em conformidade com a classificação binária almejada), fornecendo valores no intervalo $\{0, 1\}$, indicando a associação do vetor de entrada à condição normal (saídas com valores próximos a 0) ou anômala (valores de saída próximos a 1) da série empregada para estimar o vetor de ocupação do PFSA.

Para efeito de comparação inicial, são consideradas máquinas D-Markov com $D = 2$ para as três metodologias e, para cada uma delas, é fornecido um conjunto de dados gerados a partir da leitura de sensores no sistema MaFaulDa e relacionados a operações normais e de anomalia Desalinhamento Horizontal 2mm, das quais se extraíram um número de $W = 1500$ janelas simbólicas de comprimento $w = 50$ (valores escolhidos de forma experimental), e os resultados foram classificados segundo limiares de classificação pertinentes, para construção de suas respectivas curvas ROC, que são apresentadas na Figura 17.

Figura 17 – Curvas ROC obtidas através das três diferentes análises: (GHALYAN; RAY, 2021); (FRANCH et al., 2020)); e rede neural sequencial.



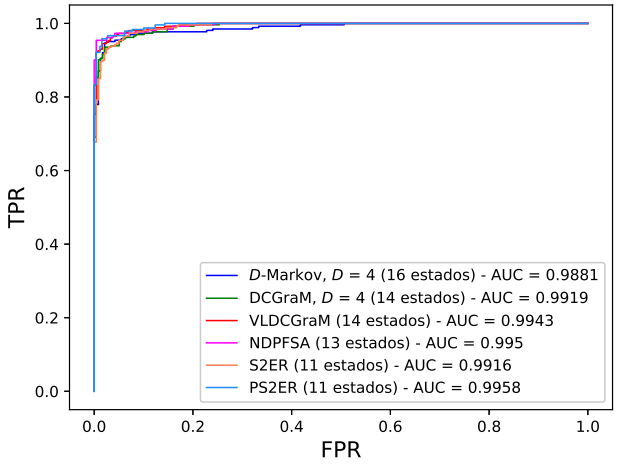
Fonte: O autor.

Fica evidente ao se observar os gráficos exibidos na Figura 17 que a adoção da metodologia proposta fornece melhor curva ROC (e consequentemente maior AUC) quando comparada àquelas das demais metodologias, apontando com isso uma forma eficaz para detecção de séries anômalas no sistema MaFaulDa.

6.3 COMPARAÇÃO ENTRE CURVAS ROC

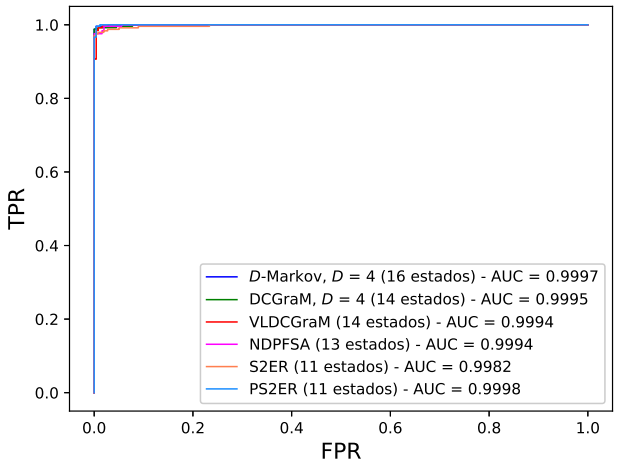
A partir das conclusões obtidas na seção anterior, gera-se maquinas D-Markov, DCGraM, VL-DCGraM, NDPFSA, S2ER e PS2ER, e calcula-se os vetores de ocupação de seus respectivos estados e aplicamos à rede neural descrita anteriormente, a fim de verificar a capacidade de cada um quanto à classificação de três tipos de anomalias associadas ao sistema MaFaulDa: Desalinhamento Horizontal de 2 mm, Desalinhamento Vertical de 1,90 mm e Desbalanceamento de eixo de 15g. As curvas ROC para cada modelo referentes a cada uma das falhas podem ser vistas nas Figuras 18 a 20.

Figura 18 – Curvas ROC geradas a partir de modelos D-Markov, DCGraM, VLDCGraM, NDPFSA, S2ER e PS2ER geradas a partir dos vetores de ocupação de estado aplicados à rede neural simples com relação à falha 'Desalinhamento Horizontal 2mm' no sistema MaFaulDa.



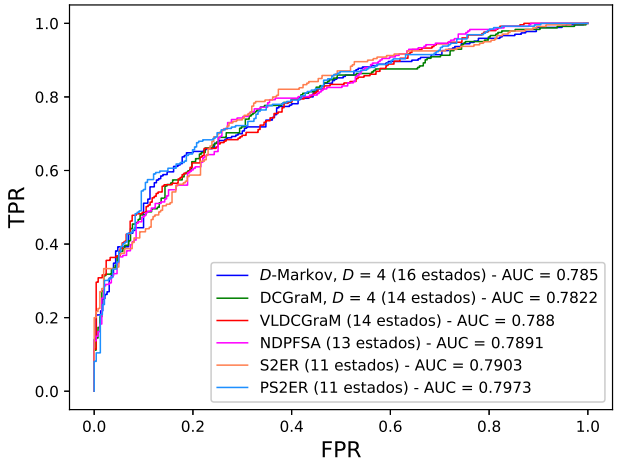
Fonte: O autor.

Figura 19 – Curvas ROC geradas a partir de modelos D-Markov, DCGraM, VLDCGraM, NDPFSA, S2ER e PS2ER geradas a partir dos vetores de ocupação de estado aplicados à rede neural simples com relação à falha 'Desalinhamento Vertical 1.90mm' no sistema MaFaulDa.



Fonte: O autor.

Figura 20 – Curvas ROC geradas a partir de modelos D-Markov, DCGraM, VLDCGraM, NDPFSA, S2ER e PS2ER geradas a partir dos vetores de ocupação de estado aplicados à rede neural simples com relação à falha ‘Desbalanceamento 15g’ no sistema MaFaulDa.

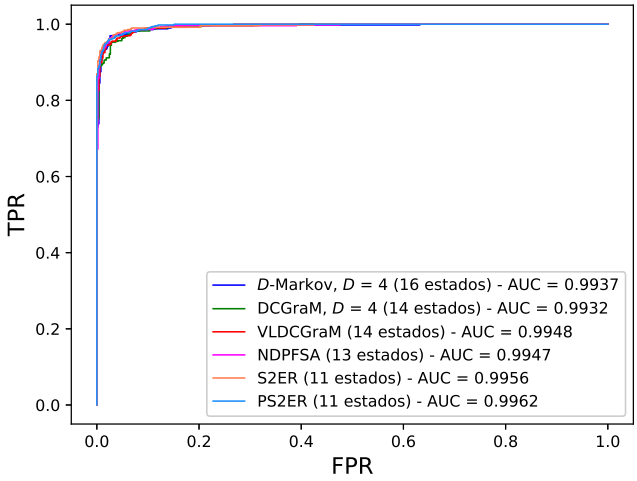


Fonte: O autor.

Analisando as figuras, podemos concluir que os modelos apresentam capacidades de detecção da falha relativamente altas, visto que apresentam valores AUC próximos a 1, porém com os modelos S2ER e PS2ER possuindo menor número de estados que as demais máquinas. É possível notar ainda que o modelo PS2ER proposto, apesar de possuir mesmo número de estados que o modelos S2ER, apresenta desempenho discretamente melhor que este último nos três casos. Assim sendo, o modelo proposto se mostra como opção viável para análises de detecção de anomalias e que, associado a uma rede neural simples, pode favorecer a viabilidade de implementação em *hardwares* com restrição de recursos.

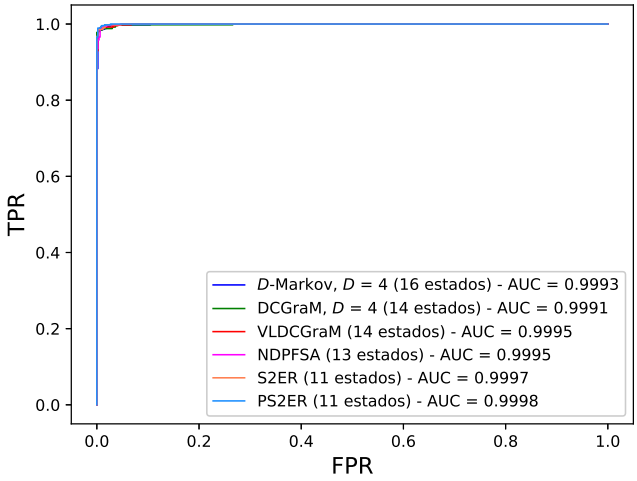
Agora é empregado o algoritmo SVM (*Support Vector Machines*) (BISHOP, 2016, Capítulo 7), para classificar entre o comportamento normal ou o comportamento anômalo. Os vetores de ocupação de estado estimados são aplicados como recursos de entrada para a SVM. As curvas ROC para os três tipos de falta consideradas nesta seção são mostradas nas Figuras 21 a 20. A superioridade do PS2ER também é observada nessas curvas. Além disso, vale ressaltar que a SVM atinge valores de AUC superiores ao comparar curvas associadas à mesma falha e ao mesmo valor de L para o caso da rede neural simples visto acima, Figuras 18 a 20.

Figura 21 – Curvas ROC geradas a partir de modelos *D*-Markov, DCGraM, VLDCGraM, NDPFSA, S2ER e PS2ER geradas a partir dos vetores de ocupação de estado aplicados à SVM com relação à falha ‘Desalinhamento Horizontal 2mm’ no sistema MaFaulDa.



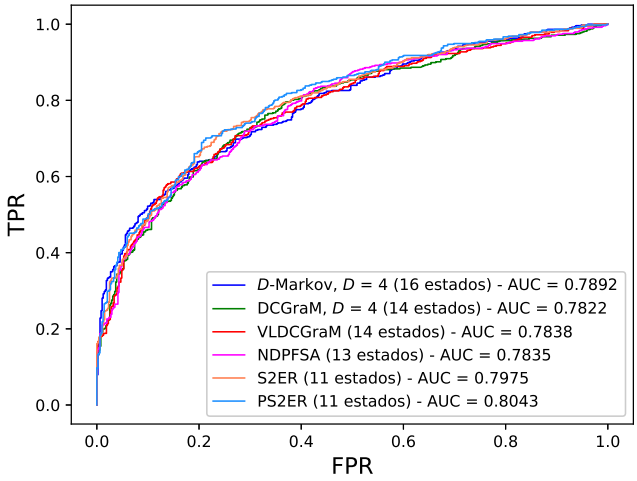
Fonte: O autor.

Figura 22 – Curvas ROC geradas a partir de modelos *D*-Markov, DCGraM, VLDCGraM, NDPFSA, S2ER e PS2ER geradas a partir dos vetores de ocupação de estado aplicados à SVM com relação à falha ´Desalinhamento Vertical 1.90mm’ no sistema MaFaulDa.



Fonte: O autor.

Figura 23 – Curvas ROC geradas a partir de modelos D-Markov, DCGraM, VLDCGraM, NDPFSA, S2ER e PS2ER geradas a partir dos vetores de ocupação de estado aplicados à SVM com relação à falha ‘Desbalanceamento 15g’ no sistema MaFaulDa.



Fonte: O autor.

Os valores de AUC obtidos pela rede neural simples e SVM são agora comparados a uma abordagem de detecção que usa Redes Neurais Convolucionais Profundas com Kernels Amplos de Primeira Camada (WDCNN), tendo uma série temporal de comprimento w como entrada. Esta rede é desenvolvida com o objetivo de permitir a operação direta com dados oriundos da serie temporal e consiste em cinco camadas convolucionais e de *pooling* intercaladas entre si, seguidas por uma camada oculta totalmente conectada e, finalmente, uma camada *softmax*. A normalização em lote é aplicada após cada camada convolucional e cada camada totalmente conectada para otimizar o desempenho do WDCNN. Considera-se também a SVM treinada utilizando a mesma base de dados empregada para o WDCNN. Os resultados são mostrados na Tabela 11 para a falha de desalinhamento horizontal, na Tabela 12 para a falha de desalinhamento vertical e na Tabela 13 para a falha de desequilíbrio. Os valores das Tabelas 11 e 12 são muito próximos enquanto a vantagem da proposta é observada na Tabela 13.

Tabela 11 – Valores AUC calculados para diferentes tamanhos de janela w com diferentes métodos de classificação para o desalinhamento horizontal de 2mm no sistema MaFaulDa.

Tamanho da janela / Método	w = 200	w = 300	w = 400
WDCNN (dados temporais)	1.000	1.000	1.000
SVM (dados temporais)	1.000	1.000	1.000
SVM com vetores de ocupação dos estados do PS2ER	0.9962	0.9982	0.9996

Fonte: O autor.

Tabela 12 – Valores AUC calculados para diferentes tamanhos de janela w com diferentes métodos de classificação para o desalinhamento vertical de 1.90mm no sistema MaFaulDa.

Método \ Tamanho da janela	w = 200	w = 300	w = 400
WDCNN (dados temporais)	1.000	1.000	1.000
SVM (dados temporais)	1.000	1.000	1.000
SVM com vetores de ocupação dos estados do PS2ER	0.9998	1.000	1.000

Fonte: O autor.

Tabela 13 – Valores AUC calculados para diferentes tamanhos de janela w com diferentes métodos de classificação para o desbalanceamento de 15g no sistema MaFaulDa.

Método \ Tamanho de janela	w = 200	w = 300	w = 400
WDCNN (dados temporais)	0.7488	0.7503	0.8229
SVM (dados temporais)	0.7441	0.7857	0.8153
SVM com vetores de ocupação dos estados do PS2ER	0.8043	0.8257	0.8673

Fonte: O autor.

6.4 CLASSIFICAÇÃO DE MÚLTIPLAS FALHAS NO SISTEMA MAFAULDA

Nesta seção, são apresentados resultados de técnicas de detecção de falhas aplicadas ao sistema MaFaulDa. O modo de operação normal bem como os 41 diferentes tipos de falhas com níveis de severidade variados impostos ao aparato experimental durante o procedimento de aquisição de sinais são identificados por rótulos numéricos, conforme listados na Tabela 14.

Tabela 14 – Sumário dos diferentes tipos de falhas do sistema MaFaulDa.

Falha	**Nível de severidade**		**Rótulo de classe**
Operação Normal	Ausente		0
Desalinhamento horizontal	(0.5, 1.0, 1.5 , 2.0) mm		1, 2, 3, 4
Desalinhamento vertical	(0.51, 0.63, 1.27, 1.4, 1.78, 1.90) mm		5, 6, 7, 8, 9, 10
Desbalanceamento	(6, 10, 15, 20, 25, 30, 35) g		11, 12, 13, 14, 15, 16, 17
Falha de rolamento	Overhang	Cage: (0, 6, 20, 35) g	18, 19, 20, 21
		Ball: (0, 6, 20, 35) g	22, 23, 24, 25
		Outer Race: (0, 6, 20, 35) g	26, 27, 28, 29
	Underhang	Cage: (0, 6, 20, 35) g	30, 31, 32, 33
		Ball: (0, 6, 20, 35) g	34, 35, 36, 37
		Outer Race: (0, 6, 20, 35) g	38, 39, 40, 41

Fonte: O autor.

Diversos trabalhos utilizam o banco de dados MaFaulDa para estudos de detecção de falha. Embora seja composto por 41 classes de diferentes níveis de severidade da falha (como visto na Tabela 14), alguns trabalhos abordam diferentes classes sem considerar o nível de falha (MARINS et al., 2018; ALI et al., 2019) ou apenas um subconjunto do elementos do banco de dados completo (PESTANA-VIANA et al., 2016; SOUZA et al., 2021). Em uma abordagem mais recente (KHAN; HWANG; KIM, 2021), a qual é adotada para comparação

com os resultados obtidos neste trabalho, o banco de dados MaFaulDa completo é utilizado, considerando todos modos de operação e níveis de severidade da falha existentes. Além disso, visando lidar com o desequilíbrio de dados (devido à diferença de dados disponíveis associados a operação normal em comparação aos de falha), esse trabalho emprega a ideia de sensores virtuais, na qual as medições realizadas pelos sensores no sistema são replicadas sob a perspectiva de diferentes pontos em um sistema de coordenadas, gerando com isso um aumento artificial dos dados para os diferentes modos de operação. Os dados originais e os aumentados são então fornecidos como dados de entrada na forma de escalogramas para modelos de classificação baseados em aprendizagem profunda. As redes consideradas incluem ResNet18 (HE et al., 2016), SqueezeNet (IANDOLA et al., 2016), bem como uma rede personalizada para o problema específico, conforme descrito em (KHAN; HWANG; KIM, 2021).

6.4.1 Metodologia proposta para multi classificação via vetores de ocupação

Neste trabalho, considera-se uma metodologia para detecção de sequências com característica anômala, baseada na adoção de uma rede neural que utiliza, como dados de treinamento, os vetores que armazenam valores de probabilidade de ocupação dos estados de um PFSA. Esses valores são estimados através de um passeio aleatório sobre um PFSA que modela a operação normal do sistema (sem presença de falhas), empregando sequências associadas às condições de operação normal e de falha. Estes vetores são usados em (SANTOS; CHAVES; PIMENTEL, 2024) para a classificação binária (sequência normal ou anômala) do sistema MaFaulDa e são calculados como descrito a seguir.

Dado um PFSA gerado a partir de uma sequência discreta S^o associada ao comportamento normal do sistema, realiza-se uma caminhada sobre os estados deste PFSA usando uma sequência discreta obtida do sistema MaFaulDa de comprimento W. Essa caminhada consiste na visitação dos estados do PFSA lendo-se sequencialmente os símbolos da sequência e seguindo os rótulos de transição correspondentes, de modo que a transição para um novo estado se dá de acordo com o símbolo indicado pela sequência percorrida. A distribuição da ocupação dos estados nesta caminhada, denotada por μ^0 para o caso da sequência normal e μ' para o caso de falha, correspondem aos vetores de ocupação que são fornecidos como entrada da rede neural a ser utilizada para realizar o processo de classificação.

A arquitetura de rede adotada para esta classificação é a ResNet18 (HE et al., 2016). Por comumente estar associada à aplicações em visão computacional, a ResNet18 costuma ter como entrada matrizes tridimensionais, geralmente associadas a imagens. Neste trabalho, a entrada da rede é adaptada para admitir vetores unidimensionais que correspondem aos vetores de ocupação de estados de um PFSA.

Na Figura 24 é exibida a matriz de confusão gerada a partir da classificação dos vetores de ocupação dos estados do PFSA construído com o algoritmo PS2ER com $N_{\max}$ = 30, $l_{\max} = 9$ e $\eta = 10^{-6}$, e que são fornecidos como entrada da rede neural. Nessa matriz, as linhas e colunas

correspondem aos rótulos das classes verdadeiras e aos rótulos previstos pela rede para as classes fornecidas, respectivamente. As observações classificadas corretamente encontram-se nas células da diagonal principal, e as classificações incorretas nas demais células.

Os vetores de ocupação são calculados adotando-se janelas de símbolos de comprimento $w = 3000$ geradas a partir das series simbólicas binárias provenientes do banco de dados MaFaulDa associados a um dos sensores (terceira coluna de dados dos arquivos provenientes do banco de dados). No total são gerados 5000 vetores tanto para operação Normal (classe 0) como para cada uma das 41 variações de falhas (classes 1 a 41). No total, 210000 vetores de ocupação são gerados, sendo 98% utilizados para treino, 1% para validação e 1% para teste da rede. O otimizador adotado é o Adam, com uma taxa de aprendizado de 10^{-3}, e o número de épocas adotado para a fase de treinamento é 50.

Na Figura 24, verifica-se que a metodologia proposta fornece uma classificação com um nível de exatidão (razão entre o número de classificações corretas e o total de classificações realizadas) de 99,14%. Esse valor é superior ao resultado obtido por (KHAN; HWANG; KIM, 2021) para o mesmo conjunto de dados analisado, que foi de 96,36%.

Esse resultado é reforçado pelos dados na Tabela 15, que exibe os valores de dois quantificadores comumente utilizados para análise da qualidade de classificação de uma rede neural: *Precision*, que representa a razão entre predições verdadeiras positivas e a soma entre predições verdadeiras positivas e falsas positivas; e *Recall*, que corresponde à razão entre predições verdadeiras positivas e a soma entre predições verdadeiras positivas e falsas negativas. Em geral, maiores valores de *Precision* e *Recall* indicam maior exatidão no processo de classificação de uma rede neural.

Na Tabela 15, compara-se os valores desses quantificadores para a metodologia proposta e para o método apresentado por (KHAN; HWANG; KIM, 2021). Observa-se que a metodologia proposta gera valores de *Precision* maiores (em destaque) ou iguais aos obtidos por (KHAN; HWANG; KIM, 2021) para 39 das 42 classes (92,86%). Além disso, valores de *Recall* maiores (em destaque) ou iguais são obtidos para 40 das 42 classes (95,24%).

É importante ressaltar que, no estudo de (KHAN; HWANG; KIM, 2021) o tempo para a tomada de decisão na classificação dos dados está associado ao tempo de aquisição de dados temporais pelo sistema (250.000 amostras captadas à taxa de 50kHz durante um intervalor de 5 segundos, conforme (MARINS et al., 2018)). Já no modelo proposto, o tempo transcorrido para a tomada de decisão está relacionado ao tamanho da janela w adotada para o cálculo do vetor de ocupação do modelo PS2ER (3.000 amostras captadas à taxa de 50kHz durante um intervalo 60 milissegundos). Isso demonstra um ganho significativo no tempo de processamento pelo modelo proposto.

Portanto, os valores presentes na Matriz de Confusão e na Tabela 15 atestam que o uso da metodologia proposta para a classificação de múltiplas falhas se mostra um método eficaz

para a classificação de falhas do sistema MaFaulDa.

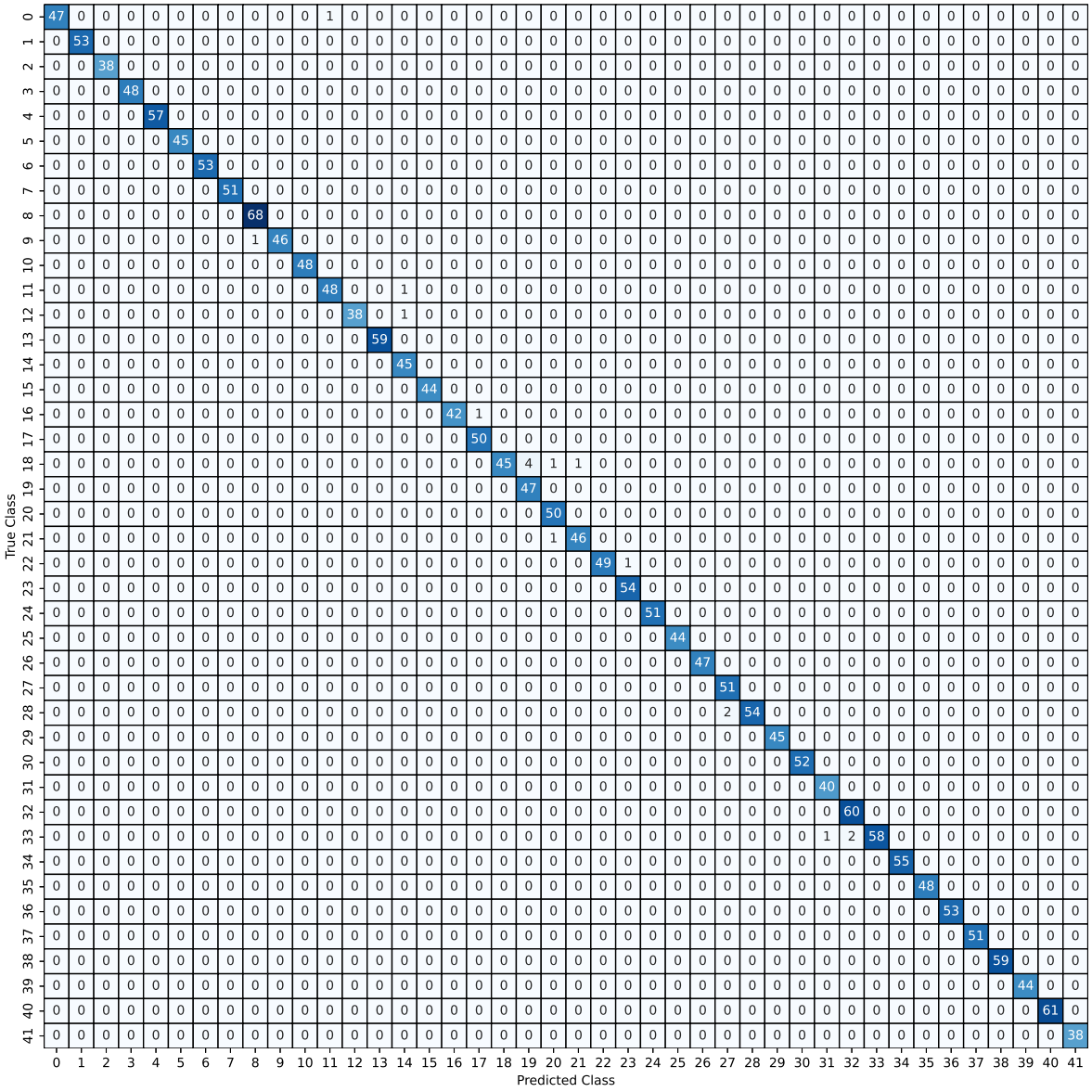


Figura 24 – Matriz de Confusão para classificação de vetores de ocupação via rede neural ResNet18 no sistema MaFaulDa. Precisão de classificação do conjunto de teste de $99{,}14\%$ para vetores de ocupação do estados do modelo PS2ER com $N_{\max}$ = 30. (0: Operação Normal; 1-4: Desalinhamento Horizontal de (0,5, 1,0, 1,5, 2,0) mm; 5-10: Desalinhamento Vertical de (0,51, 0,63, 1,27, 1,40, 1,78, 1,90) mm; 11-17: Desbalanceamento com (6, 10, 15, 20, 25, 30, 35) g de massa; 18-21: falha de rolamento na posição Overhang Ball com (0, 6, 20, 35) g de desbalanceamento; 22-25: falha de rolamento na posição Overhang Cage com (0, 6, 20, 35) g de desbalanceamento; 26-29: falha de rolamento na posição Overhang Outer Race com (0, 6, 20, 35) g de desbalanceamento; 30-33: falha de rolamento na posição Underhang Ball com (0, 6, 20, 35) g de desbalanceamento; 34-37: falha de rolamento na posição Underhang Cage com (0, 6, 20, 35) g de desbalanceamento; 38-41: falha de rolamento na posição Underhang Outer Race com (0, 6, 20, 35) g de desbalanceamento).

Fonte: O autor.

Tabela 15 – Comparação de dois quantificadores, precision e recall, para o sistema MaFaulDa via abordagem proposta e a apresentada em (KHAN; HWANG; KIM, 2021). Os melhores resultados são destacados em negrito.

Quantificador	Precision		Recall	
Modelo / Classe	(KHAN; HWANG; KIM, 2021)	PS2ER	(KHAN; HWANG; KIM, 2021)	PS2ER
0	0.99	**1.00**	0.95	**0.98**
1	0.89	**1.00**	0.99	**1.00**
2	1.00	1.00	0.94	**1.00**
3	1.00	1.00	0.96	**1.00**
4	0.99	**1.00**	1.00	1.00
5	0.97	**1.00**	1.00	1.00
6	1.00	1.00	0.95	**1.00**
7	0.98	**1.00**	0.98	**1.00**
8	0.93	**0.99**	0.96	**1.00**
9	1.00	1.00	0.95	**0.98**
10	0.97	**1.00**	0.97	**1.00**
11	0.96	**0.98**	0.96	**0.98**
12	0.91	**1.00**	0.96	**0.97**
13	0.92	**1.00**	0.91	**1.00**
14	0.91	**0.96**	0.91	**1.00**
15	0.95	**1.00**	0.88	**1.00**
16	0.91	**1.00**	0.97	**0.98**
17	0.93	**0.98**	0.98	**1.00**
18	0.99	**1.00**	**1.00**	0.88
19	**0.99**	0.92	0.96	**1.00**
20	0.86	**0.96**	1.00	1.00
21	**1.00**	0.98	0.80	**0.98**
22	0.98	**1.00**	0.98	0.98
23	0.96	**0.98**	0.98	**1.00**
24	0.99	**1.00**	0.96	**1.00**
25	0.99	**1.00**	1.00	1.00
26	1.00	1.00	0.96	**1.00**
27	**0.98**	0.96	0.95	**1.00**
28	1.00	1.00	**0.98**	0.96
29	0.91	**1.00**	0.99	**1.00**
30	0.95	**1.00**	0.98	**1.00**
31	0.98	0.98	0.92	**1.00**
32	0.89	**0.97**	0.97	**1.00**
33	0.93	**1.00**	0.87	**0.95**
34	0.94	**1.00**	0.99	**1.00**
35	0.97	**1.00**	0.95	**1.00**
36	0.99	**1.00**	1.00	1.00
37	1.00	1.00	0.96	**1.00**
38	1.00	1.00	1.00	1.00
39	0.99	**1.00**	0.99	**1.00**
40	1.00	1.00	0.99	**1.00**
41	1.00	1.00	1.00	1.00

Fonte: O autor.

7 CONCLUSÕES

Neste trabalho foram apresentados dois algoritmos para modelagem de sistemas dinâmicos discretos, denominados NDPFSA e PS2ER. Ambos analisam as estatísticas da sequência simbólica associada ao sistema a ser modelado, contudo, empregam técnicas distintas para obter PFSAs com representação compacta. Enquanto o primeiro utiliza técnicas de aprendizado de máquina para obtenção de um modelo não determinístico, o segundo faz uso de uma estrutura em árvore visando a criação de modelos determinísticos.

O algoritmo NDPFSA parte da criação de uma cadeia de Markov para um determinado D com base na sequência de saída de um sistema original e, em seguida, usa o algoritmo de clsuterização *c-means* seguido de um algoritmo de minimização de grafo proposto, denominado G-Moore, para obter um PFSA final. Já o algoritmo PS2ER estabelece sucessivas quebras de estados com base em uma função custo baseada em redução de entropia para construção de uma árvore na qual as folhas compõem os estados de um PFSA.

Os algoritmos NDPFSA e PS2ER foram aplicados a cinco sistemas dinâmicos: MOSFET, Mapa de Hénon, Duffing, CWRU e MaFaulDa. Seus resultados foram comparados às máquinas D-Markov, além de máquinas DCGraM (FRANCH et al., 2020) e VL-DCGraM (SANTOS; CHAVES; PIMENTEL, 2024), métodos de refinamento sobre máquinas D-Markov que fazem uso de técnicas de aprendizado de máquina.

Foi demonstrado que, para o MOSFET e Mapa de Hénon, as máquinas geradas pelo modelo NDPFSA apresentaram melhor desempenho quando comparada às demais metodologias. Já para os três sistemas restantes, o modelo PS2ER mostrou desempenho superior aos demais, apresentando valores reduzidos de entropia condicional e divergência de Kullback-Leibler.

Para os sistemas Duffing e MaFaulDa, o PFSA gerado pelo PS2ER é notavelmente menor que as demais propostas, apresentando valores de entropia condicional e divergência de Kullback-Leibler mais próximos ao do sistema original do que os outros modelos. Além disso, em decorrência das especificidades do processo de cálculo da função de transição do PFSA final e em vista dos resultados limitados da estratégia S2ER, ficou evidenciada a necessidade de garantir a máxima permanência de estados gerados pelo processo inicial de quebra. A eficácia do método PS2ER é ratificada pela entropia condicional e a divergência de Kullback-Leibler, que refletem a capacidade do modelo em capturar a memória e a estatística da sequência modelada.

Por fim, o destaque da proposta PS2ER é reiterado ao aplicarmos o vetor de ocupação de estados desse modelo como dado de entrada de redes neurais em um processo de identificação de falhas no sistema MaFaulDa. Foi demonstrado que essa abordagem apresentou precisão de classificação superior à outras encontradas na literatura que também fazem uso do vetor de ocupação de estados de um PFSA (GHALYAN; RAY, 2021), além de superar os resultados obtidos com uso de rede neural dedicada à classificação de dados temporais (KHAN; HWANG;

KIM, 2021). Foi utilizada uma rede neural com topologia ResNet18 adaptada para entradas unidimensionais, a fim de classificar os dados de ocupação de estados PS2ER em relação aos diversos tipos de falhas no sistema MaFaulDa. Essa abordagem foi comparada com outra metodologia da literatura, que utiliza uma rede neural customizada para a tarefa de multi-classificação. O resultado obtido demonstrou uma notável superioridade do uso dos vetores de ocupação como dados de treinamento para uma rede neural convolucional na classificação de diversos tipos de falhas no sistema MaFaulDa. Isso sugere um grande potencial dessa metodologia para a classificação de falhas em outros sistemas dinâmicos.

7.1 TRABALHOS FUTUROS

A seguir são apresentadas propostas de possíveis desenvolvimentos futuros para esta tese.

- Aprimoramento do modelo NDPFSA: A metodologia para construção do conjunto de estados do modelo NDPFSA permite uma discussão aprofundada para alterar a forma de conexão entre estados. Uma vez que a probabilidade de transição dos estados não determinísticos é calculada por uma distância euclidiana, e não por uma medida probabilística (fato proveniente da interpretação do algoritmo *c-means*), uma possibilidade de abordagem distinta é a adoção de inferência bayesiana a partir da máquina gerada, a fim de refinar as probabilidades de transição desses estados;
- Extensão da proposta de identificação de falhas baseada em redes neurais tendo vetores de ocupação de estados de PFSA como dados de entrada para sistemas com restrição de hardware, visando aplicação em sistema com rede de sensores e microcontrolados:
  - Avaliação de estrutura de redes neurais adequadas ao emprego de aritmética de ponto fixo em substituição ao uso de ponto flutuante, com o objetivo de viabilizar a aplicação do algoritmo em unidades de hardware com restrições de hardware ou de energia;
  - Estudos para aplicação do par PFSA + Rede Neural visando aplicações em sistemas de sensoriamento utilizando metodologia de fusão de dados. Nesse método, os dados adquiridos pelos sensores são condensados, e a transmissão de sinais ocorre quando há relevância para alguma tomada de decisão. Dessa forma, a transmissão é realizada apenas na ocorrência de dados potencialmente relevantes, o que favorece a redução do consumo de energia pelo sistema.
- Aprimoramento da geração da *context-tree* no modelo VLDCGraM adotando técnicas de teoria de codificação de fonte:
  - Análise de uma *context-tree* como uma estrutura de codificação de fonte sobre sequências (*stream codes* (Mackay, 2005, Cap. 6)) ao se estabelecer uma relação com

códigos aritméticos. Estes códigos apresentam a vantagem de se adaptar à estatística da fonte a partir de uma distribuição de probabilidade previamente definida;

– Justificativa do valor da constante α na Equação (2.27) que determina o limiar de quebra de estados no algoritmo VL-DCGraM empregando procedimentos universais de compressão de dados aplicados à fonte com memória finita.

7.2 TRABALHOS PUBLICADOS

- H. Í. dos Santos, D. P. B. Chaves, and C. Pimentel, "Determinação de modelo markoviano para canal sem fio empregando aprendizado de máquina," in XXXVIII Simpósio Brasileiro de Telecomunicações e Processamento de Sinais, Florianópolis, SC, 2020.
- H. Í. dos Santos, D. P. B. Chaves, and C. Pimentel, "Construção de modelos markovianos por redução de entropia sobre Árvores," in XLI Simpósio Brasileiro de Telecomunicações e Processamento de Sinais, São José dos Campos, SP, 2023.
- H. Í. dos Santos, D. P. B. Chaves, and C. Pimentel, "Symbolic dynamical filtering via variable length markov model and machine learning," IEEE Access, vol. 12, pp. 19 778–19 789, 2024.

REFERÊNCIAS

ABELL, M. L.; BRASELTON, J. P. **Introductory differential equations**. 5. ed. London, United Kingdom: Academic Press, 2018.

ALI, M. A. et al. The influence of handling imbalance classes on the classification of mechanical faults using neural networks. In: IEEE. **2019 8th International Conference on Modeling Simulation and Applied Optimization (ICMSAO)**. [S.l.], 2019. p. 1–5.

ALTINEL, D.; KURT, G. K. Finite-state markov channel based modeling of rf energy harvesting systems. **IEEE Transactions on Vehicular Technology**, IEEE, v. 67, n. 2, p. 1713–1725, 2017.

BABICH, F.; KELLY, O. E.; LOMBARDI, G. A context-tree based model for quantized fading. **IEEE communications letters**, IEEE, v. 3, n. 2, p. 46–48, 1999.

BANDYOPADHYAY, S.; BHATTACHARYA, R. **Discrete and continuous simulation: theory and practice**. [S.l.]: CRC Press, 2014.

BARKER, M.; RAYENS, W. Partial least squares for discrimination. **Journal of Chemometrics: A Journal of the Chemometrics Society**, Wiley Online Library, v. 17, n. 3, p. 166–173, 2003.

BERSTEL, J. et al. Minimization of automata. 2010. Disponível em: <https://arxiv.org/pdf/1010.5318>.

BEZDEK, J. C. Objective function clustering. In: **Pattern recognition with fuzzy objective function algorithms**. USA: Springer, 1981. p. 43–93.

BHATTACHARYA, C.; RAY, A. Thresholdless classification of chaotic dynamics and combustion instability via probabilistic finite state automata. **Mechanical Systems and Signal Processing**, Elsevier, v. 164, p. 108213, 2022.

BISHOP, C. **Pattern Recognition and Machine Learning**. [S.l.]: Springer New York, 2016. (Information Science and Statistics).

BRADLEY, E.; KANTZ, H. Nonlinear time-series analysis revisited. **Chaos: An Interdisciplinary Journal of Nonlinear Science**, AIP Publishing, v. 25, n. 9, p. 097610, 2015.

BÜHLMANN, P. Model selection for variable length markov chains and tuning the context algorithm. **Annals of the Institute of Statistical Mathematics**, Springer, v. 52, n. 2, p. 287–315, 2000.

CELAYA, J. et al. Capacitor electrical stress data set-2. **NASA Ames Research Center, Moffett Field, CA**, 2012.

CELAYA, J. R. et al. Towards a model-based prognostics methodology for electrolytic capacitors: A case study based on electrical overstress accelerated aging. **Int. J. Prognostics Health Manage.**, v. 3, n. 2, p. 33, 2012.

CHEN, Z. et al. Fault detection for non-gaussian processes using generalized canonical correlation analysis and randomized algorithms. **IEEE Transactions on Industrial Electronics**, IEEE, v. 65, n. 2, p. 1559–1567, 2017.

COVER, T. M.; THOMAS, J. A. **Elements of information theory**. [S.l.]: John Wiley & Sons, 2012.

CWRU. 2023. Disponível em: <https://engineering.case.edu/bearingdatacenter>. Acesso em 06 de Setembro de 2023.

DING, S. X. **Model-based fault diagnosis techniques: design schemes, algorithms, and tools**. [S.l.]: Springer Science & Business Media, 2008.

DUNTEMAN, G. H. **Principal components analysis**. [S.l.]: Sage, 1989.

FRANCH, D. K. **Dynamical System Modeling With Probabilistic Finite State Automata**. Dissertação (Mestrado) — Universidade Federal de Pernambuco, 2017.

FRANCH, D. K. et al. Markov modeling of dynamical systems via clustering and graph minimization. **Digital Signal Processing**, v. 104, p. 102769, 2020. ISSN 1051-2004. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1051200420301147>.

GAO, Z.; CECATI, C.; DING, S. X. A survey of fault diagnosis and fault-tolerant techniques—part i: Fault diagnosis with model-based and signal-based approaches. **IEEE Transactions on Industrial Electronics**, IEEE, v. 62, n. 6, p. 3757–3767, 2015.

GHALYAN, N. F.; RAY, A. Measure invariance of ergodic symbolic systems for low-delay detection of anomalous events. **Mechanical Systems and Signal Processing**, Elsevier, v. 159, p. 107746, 2021.

GRABEN, P. B. Estimating and improving the signal-to-noise ratio of time series by symbolic dynamics. **Physical Review E**, APS, v. 64, n. 5, p. 051104, 2001.

HAJEK, B. **Random processes for engineers**. [S.l.]: Cambridge university press, 2015.

HAMERLY, G.; ELKAN, C. Alternatives to the k-means algorithm that find better clusterings. In: ACM. **Proceedings of the eleventh international conference on Information and knowledge management**. [S.l.], 2002. p. 600–607.

HE, K. et al. Deep residual learning for image recognition. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2016. p. 770–778.

HÉNON, M. A two-dimensional mapping with a strange attractor. **The theory of chaotic attractors**, Springer, p. 94–102, 2004.

HOPCROFT, J. E. **Introduction to automata theory, languages, and computation**. [S.l.]: Pearson Education India, 2008.

IANDOLA, F. N. et al. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size. **arXiv preprint arXiv:1602.07360**, 2016.

KANUNGO, T. et al. An efficient k-means clustering algorithm: Analysis and implementation. **IEEE transactions on pattern analysis and machine intelligence**, IEEE, v. 24, n. 7, p. 881–892, 2002.

KHAN, A.; HWANG, H.; KIM, H. S. Synthetic data augmentation and deep learning for the fault diagnosis of rotating machines. **Mathematics**, MDPI, v. 9, n. 18, p. 2336, 2021.

KIM, T. et al. An on-board model-based condition monitoring for lithium-ion batteries. **IEEE Transactions on Industry Applications**, IEEE, v. 55, n. 2, p. 1835–1843, 2018.

LALLEMENT, G. **Semigroups and combinatorial applications**. [S.l.]: John Wiley & Sons, Inc., 1979.

LI, F. et al. Anomaly detection in gas turbine fuel systems using a sequential symbolic method. **Energies**, Multidisciplinary Digital Publishing Institute, v. 10, n. 5, p. 724, 2017.

LI, Y.; RAY, A. Unsupervised symbolization of signal time series for extraction of the embedded information. **Entropy**, Multidisciplinary Digital Publishing Institute, v. 19, n. 4, p. 148, 2017.

LIANG, G. et al. An unsupervised fault diagnosis method based on iterative multi-manifold spectral clustering. **IET Collaborative Intelligent Manufacturing**, IET, 2019.

LIND, D.; MARCUS, B. **An Introduction To Symbolic Dynamics and Codings**. [S.l.]: Cambridge University Press, 1995.

LORENA, A. C.; CARVALHO, A. C. de. Uma introdução às support vector machines. **Revista de Informática Teórica e Aplicada**, v. 14, n. 2, p. 43–67, 2007.

MACKAY, D. J. **Information theory, inference and learning algorithms**. [S.l.]: Cambridge university press, 2003.

MANDREKAR, J. N. Receiver operating characteristic curve in diagnostic test assessment. **Journal of Thoracic Oncology**, v. 5, n. 9, p. 1315–1316, 2010.

MARINS, M. A. et al. Improved similarity-based modeling for the classification of rotating-machine failures. **Journal of the Franklin Institute**, Elsevier, v. 355, n. 4, p. 1913–1930, 2018.

MIYAMOTO, S. et al. **Algorithms for fuzzy clustering**. [S.l.]: Springer, 2008. Cap. 2, 16-20 p.

MOORE, E. F. Gedanken-experiments on sequential machines. **Automata studies**, v. 34, p. 129–153, 1956.

MUKHERJEE, K.; RAY, A. State splitting and merging in probabilistic finite state automata for signal representation and analysis. **Signal Processing**, v. 104, p. 105–119, 2014.

PENG, T. et al. A probabilistic finite state automata-based fault detection method for traction motor. In: IEEE. **2020 IEEE 29th International Symposium on Industrial Electronics (ISIE)**. [S.l.], 2020. p. 1199–1204.

PESTANA-VIANA, D. et al. The influence of feature vector on the classification of mechanical faults using neural networks. In: IEEE. **2016 IEEE 7th Latin American Symposium on Circuits & Systems (LASCAS)**. [S.l.], 2016. p. 115–118.

PIMENTEL, C.; ALAJAJI, F. Packet-based modeling of reed–solomon block-coded correlated fading channels via a markov finite queue model. **IEEE Transactions on Vehicular Technology**, IEEE, v. 58, n. 7, p. 3124–3136, 2009.

RAY, A. Symbolic dynamic analysis of complex systems for anomaly detection. **Signal Processing**, Elsevier, v. 84, n. 7, p. 1115–1130, 2004.

REN, H.; YE, Z.; LI, Z. Anomaly detection based on a dynamic Markov model. **Information Sciences**, Elsevier, v. 411, p. 52–65, 2017.

SANTOS, H. D.; CHAVES, D. P. B.; PIMENTEL, C. Symbolic dynamical filtering via variable length Markov model and machine learning. **IEEE Access**, v. 12, p. 19778–19789, 2024.

SAXENA, K. et al. Symbolic dynamic filtering for online power quality anomaly detection. In: IEEE. **2020 IEEE Power & Energy Society General Meeting (PESGM)**. [S.l.], 2020. p. 1–5.

SMITH, W. A.; RANDALL, R. B. Rolling element bearing diagnostics using the case western reserve university data: A benchmark study. **Mechanical Systems and Signal Processing**, v. 64-65, p. 100–131, Dec. 2015.

SOUZA, R. M. et al. Deep learning for diagnosis and classification of faults in industrial rotating machinery. **Computers & Industrial Engineering**, Elsevier, v. 153, p. 107060, 2021.

SpectraQuest Inc. 2021. Disponível em: <https://spectraquest.com/simulators/details/mfs/>. Acesso em 23 Maio de 2021.

VIDAL, E. et al. Probabilistic finite-state machines-part i. **IEEE transactions on pattern analysis and machine intelligence**, IEEE, v. 27, n. 7, p. 1013–1025, 2005.

VIDAL, E. et al. Probabilistic finite-state machines-part ii. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, IEEE, v. 27, n. 7, p. 1026–1039, 2005.

WANG, L.; ZHONG, Y.; MA, W. Gps-data-driven dynamic destination prediction for on-demand one-way carsharing system. **IET Intelligent Transport Systems**, IET, v. 12, n. 10, p. 1291–1299, 2018.

WEISS, M. A. Data structures and problem solving using java. **ACM SIGACT News**, ACM New York, NY, USA, v. 29, n. 2, p. 42–49, 1998.

YONAMINE, F. S. et al. Aprendizado não supervisionado em domínios fuzzy - algoritmo fuzzy c-means. **São Carlos: UFSCAR**, p. 43, 2002.

ZADEH, L. A. Fuzzy sets. In: **Fuzzy sets, fuzzy logic, and fuzzy systems: selected papers by Lotfi A Zadeh**. [S.l.]: World Scientific, 1996. p. 394–432.