MODELAMENTO DE SISTEMAS DINÂMICOS USANDO AUTÔMATOS PROBABILÍSTICOS

Daniel K. Franch*, Diego M. Hamilton*, Daniel P. B. Chaves*, Cecilio Pimentel*

*Departamento de Eletrônica e Sistemas
Universidade Federal de Pernambuco
CEP 50740-550, Recife, PE, Brasil

Emails: daniel.k.br@ieee.org, diego.hamilton@ufpe.br, daniel.chaves@ufpe.br, cecilio@ufpe.br

Abstract— Discrete dynamical systems are widely used in a variety of scientific and engineering applications. This work presents a new algorithm to model these systems using probabilistic finite state automata (PFSA) by initially finding a PFSA called D-Markov machines and then applying machine learning algorithms and graph minimization techniques to obtain compact and precise PFSA models.

Keywords— Clustering, dynamical systems, graph minimization, Markov processes, symbolic dynamics.

Resumo— Sistemas discretos dinâmicos são amplamente utilizados em uma variedade de aplicações científicas bem como em engenharia. Este trabalho apresenta um novo algoritmo para a modelagem de sistemas discretos dinâmicos utilizando autômatos probabilísticos de estados finitos (PFSA, *probabilistic finite state automata*), encontrando inicialmente uma classe especial de PFSA chamada máquina D-Markov e então aplicando algoritmos de aprendizado de máquina e técnica de minimização de autômatos para obter modelos PFSA precisos e compactos.

Palavras-chave— Clusterização, sistemas dinâmicos, minimização de grafos, processos Markovianos, dinâmica simbólica.

1 Introdução

A análise de dados empregando séries temporais simbólicas encontra aplicação ampla: codificação de dados em teoria da informação (Shannon, 2001), extração de dados em mineração de dados supervisionada (Lin et al., 2003), análise de dados experimentais por séries temporais simbólicas (Daw et al., 2003), detecção de anomalias e reconhecimento de padrões (Ray, 2004; Mukherjee and Ray, 2014a), processos decisórios markovianos em sistemas dinâmicos discretos (Puterman, 2014). Tal aplicabilidade advém da capacidade dessas séries em expressar alterações nos padrões estatísticos do sistema dinâmico subjacente em escala de tempo lenta (em que o processo apresenta comportamento estatístico não estacionário) quando confrontados com os padrões em escala de tempo rápida (comportamento estatístico estacionário).

No contexto apresentado, as variações estatísticas do processo subjacente são desprezíveis em escala de tempo rápida, sendo esta escala adotada para a aquisição dos dados de sensores, ou seja, na suposição de um comportamento estatisticamente estacionário do sistema. Para a aquisição da série temporal representativa da dinâmica de um sistema são necessários dois passos:

- Quantização da série temporal gerada inicialmente com valores escalares ou vetoriais, que é convertida em uma série simbólica sobre um alfabeto finito (Lind and Marcus, 1995);
- Geração de automato finito (PFSA - *Probabilistic Finite State Automata*) a partir da série temporal quantizada (Mukherjee and Ray, 2014a; Chattopadhyay et al., 2011; Rajagopalan and Ray, 2006).

Com relação à etapa de quantização, a perda de informação decorrente da conversão de uma sequência sobre o contínuo para uma sobre um alfabeto finito requer particular atenção. Esse tema é abordado em (Graben, 2001), em que a possível melhoria da relação sinal-ruído no caso de sinais contaminados por ruído é discutida. Além disso, a quantização propicia comunicação digital e computação numérica mais eficientes e efetivas quando comparadas a operações similares envolvendo dados com valores contínuos. As vantagens de operar com séries simbólicas são identificadas em diversas aplicações (Ray, 2004; Mukherjee and Ray, 2014b; Piccardi, 2004; Makki Alamdari et al., 2015; Li and Ray, 2017).

A segunda etapa, geração de automato finito, tem como essência a obtenção de modelos analíticos simples que reflitam com confiabilidade o comportamento estatístico de uma sequência S associada a um sistema dinâmico simbólico. Isso envolve dois passos principais: i) escolher uma classe de modelos capaz de representar o comportamento do sistema; ii) desenvolver métodos para parametrizar o modelo. Desta forma, pode-se usar estatísticas do modelo para analisar o comportamento do sistema e aplicá-lo para a análise de desempenho (Moreira et al., 2017), detecção de anomalias (Ray, 2004; Mukherjee and Ray, 2014a).

Autômatos probabilísticos de estados finitos (PFSA, *probabilistic finite state automata*) cons-

tituem modelos eficientes para representar matematicamente a estrutura de sistemas dinâmicos. Estes modelos são simples de construir e permitem uma implementação computacional eficiente. Um algoritmo importante para a construção de um PFSA é a máquina D-Markov (Ray, 2004), um processo Markoviano de ordem finita D, que considera todas as sequências de comprimento D sobre um dado alfabeto para formar seus estados. À medida que D aumenta, a máquina D-Markov se torna capaz de reproduzir o comportamento do sistema de forma mais precisa ao custo de um crescimento exponencial do número de estados.

Neste trabalho, foi desenvolvido um algoritmo para a construção de um PFSA a partir das estatísticas de uma sequência simbólica observada S de um certo sistema dinâmico discreto. Este é chamado de D-Markov com Clusterização e Minimização de Grafos (DCGraM) e utiliza o algoritmo de clusterização *K-Means* (MacKay, 2003) e técnica de minimização de grafos (Berstel et al., 2010) da Teoria de Autômatos (como por exemplo, o algoritmo de Moore ou Hopcroft) para obter modelos de PFSA precisos e compactos. O algoritmo proposto é estruturado de acordo com as seguintes etapas:

- Construção de uma máquina de D-Markov para um certo valor D.
- Clusterização de estados com características similares, criando uma partição inicial $\mathcal{P}$ de estados.
- Aplicação de um algoritmo de minimização de grafos, refinando a partição inicial por quebra de estados a fim de melhorar a exatidão do modelo.

É apresentado um exemplo em que o DCGraM gera um modelo de PFSA com substancialmente menos estados que as máquinas D-Markov originais com precisão similar. Variando o parâmetro D e o número de *clusters* iniciais estabelece-se um compromisso entre número de estados e precisão.

O restante deste trabalho está organizado em quatro seções. A Seção 2 revisa conceitos sobre PFSA e algoritmos de clusterização e minimização de grafos. A Seção 3 apresenta o algoritmo DCGraM. O algoritmo é exemplificado na Seção 4, em que se compara seu desempenho e o número de estados obtidos em relação aos resultados obtidos com as máquinas D-Markov. Por último, as conclusões deste trabalho são apresentadas na Seção 5.

2 Preliminares

Esta seção fornece uma introdução a grafos (Lind and Marcus, 1995), PFSA (Mukherjee and Ray, 2014a), clusterização (MacKay, 2003) que é relavante para o entendimento do algoritmo proposto. Uma sequência finita u de símbolos de um alfabeto Σ é chamada uma palavra e seu comprimento é denotado por $|u|$. A palavra vazia ε é uma sequência de tamanho 0, tal que $u\varepsilon = \varepsilon u = \varepsilon$. O conjunto de todas as possíveis palavras de comprimento n sobre Σ é Σ^n e o conjunto de todas as sequências sobre Σ com todos os comprimentos possíveis, incluindo a sequência vazia ε, é Σ^*. Neste trabalho, por simplicidade de notação, denotamos por grafo um grafo direcional rotulado, conforme a Definição 1.

Definição 1 (Grafo) *Um grafo G sobre o alfabeto Σ consiste em uma tripla (Q, Σ, δ):*

- *Q é um conjunto finito de estados com cardinalidade $|Q|$;*
- *Σ é um alfabeto finito com cardinalidade $|\Sigma|$;*
- *δ é uma função de transição de estados tal que $Q \times \Sigma \to Q$.*

É possível estender a definição da função de transição para que esta aceite também palavras, e não apenas símbolos. Dada $\omega \in \Sigma^n$, em que $\omega = \sigma_1\sigma_2 \ldots \sigma_n$ com $\sigma_m \in \Sigma$, para $m = 1, \ldots, n$, e dados os estados $q_0, q_1, \ldots, q_n \in Q$, definimos a função $\delta^*(q_0, \omega) = q_n$ se $\delta(q_0, \sigma_1) = q_1, \delta(q_1, \sigma_2) = q_2, \ldots, \delta(q_{n-1}, \sigma_n) = q_n$. Se $\exists\, \omega \in \Sigma^*$ tal que para dois estados $q_1, q_2 \in Q$, $\delta^*(q_1, \omega) = q_2$, é dito então que existe um caminho entre q_1 e q_2 e que ω é gerado por q_1. Um grafo é **irredutível** se para qualquer $q_1,\ q_2 \in Q$ existe um caminho que comece em q_1 e termine em q_2. Um estado $q \in Q$ é dito sem aresta incidente se nenhuma aresta começa em q ou nenhuma aresta termina em q. Um grafo é dito **essencial** se não existem estados sem arestas incidentes.

Definição 2 (Contexto à direita) *O contexto à direita de um estado $q \in Q$ de um grafo G é definido como o conjunto de todas as palavras geradas por caminho que começam no estado q e terminam em algum estado de Q.*

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

Um grafo é chamado de reduzido se os contextos à direita de cada estado são distintos. Dado um grafo original, existem dois algoritmos para gerar um grafo reduzido: Moore e Hopcroft (Berstel et al., 2010). Se o grafo original é irredutível, o grafo reduzido é único com menor número de estados e é denominado de grafo mínimo.

Definição 3 (PFSA) *Um PFSA é um par (G, π) no qual G é um grafo e π é uma atribuição de probabilidade $\pi : Q \times \Sigma \to [0,1]$ que satisfaz $\sum_{\sigma \in \Sigma} \pi(q,\sigma) = 1$, para todo $q \in Q$.*

Consideramos na Definição 3 que todos os estados de Q são estados iniciais e finais tal que um PFSA é uma ênupla (Q, Σ, δ, π). Dado um estado $q \in Q$, a distribuição de probabilidade $\mathcal{V}(q) = \{\pi(q, \sigma); \forall \sigma \in \Sigma\}$ associada com q é chamada de *morph* deste estado.

2.1 Máquinas D-Markov

A máquina D-Markov (Mukherjee and Ray, 2014a) é um PFSA que gera símbolos que dependem apenas de D símbolos passados. Esta gera um processo Markoviano $\{s_n\}$ de ordem D

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

em que $\Pr(\cdot)$ é uma mediada de probabilidade. Para construir uma máquina D-Markov, $|\Sigma|^D$ estados são criados gerando o conjunto Q em que cada estado corresponde a uma sequência de comprimento D sobre um alfabeto Σ. Para cada $\tau \in \Sigma$, existe uma transição de estados rotulado com o símbolo τ a partir do estado $q = \sigma_1 \sigma_2 \dots \sigma_D$, $\sigma_i \in \Sigma$, para o estado $q' = \sigma_2 \dots \sigma_D \tau$, isto é $\delta(q, \tau) = q'$, com probabilidade

$$\pi(q, \tau) = \frac{\Pr(q\tau)}{\Pr(q)}, \tag{1}$$

desde que $\Pr(q\tau) > 0$, em que $q\tau$ corresponde a sequência que rotula o estado q seguida do símbolo τ. Após a remoção de estados sem aresta incidente, uma máquina D-Markov essencial é gerada com menos de $|\Sigma|^D$ estados.

2.2 Clusterização

Uma classe de técnicas de aprendizagem de máquina não supervisionada que são empregadas para particionar um conjunto de M objetos em K subconjuntos, denominados por *clusters*, com elementos munidos de algum tipo de similaridade, é denominada de clusterização (MacKay, 2003; Goodfellow et al., 2016). Há várias aplicações para as diversas técnicas de clusterização. Neste artigo, foi aplicado o algoritmo *K-Means* para definir uma partição inicial de estados do D-Markov com *morphs* similares.

O algoritmo *K-Means*, também chamado de algoritmo de Lloyd, tem por objetivo particionar um conjunto de M objetos ou pontos dispostos em um espaço de dimensão I em K *clusters*. Cada um dos K *clusters* é representado por um vetor $\mathbf{m}^{(K)}$ denominado centroide do *cluster*. Os pontos são representados por $\mathbf{x}^{(n)}$, com $n = 1, \dots, M$. Os centroides dos *clusters* $\mathbf{m}^{(K)}$ são inicializados em pontos de $\mathbb{R}^I$ e os objetos mais próximos de cada centroide são agrupados entorno deste. Em seguida, o valor do centroide é modificado para o valor médio dos objetos recém agrupados em seu entorno. Esse processo é repetido até os centroides não serem modificados após uma nova iteração, o que indica que os centroides do conjunto final de *clusters* foi obtido.

Neste trabalho, os *morphs* de cada estado são empregados para construir postos em um espaço de dimensão $|\Sigma|$, $(\pi(q,\sigma_1), \pi(q,\sigma_2), \dots, \pi(q,\sigma_{|\Sigma|}))$. Como os *morphs* correspondem a distribuições de probabilidades e, portanto, satisfazem a expressão $\Sigma_{\sigma \in \Sigma} \Pr(\sigma|q) = 1$, $\forall\ q \in Q$, os objetos de entrada do algoritmo estão definidos em um subespaço de dimensão $|\Sigma| - 1$. Portanto, como é apresentado na Seção 4, podemos representar no plano o resultado do algoritmo *K-Means* aplicado sobre o conjunto de estados de um grafo com alfabeto ternário.

3 O Algoritmo DCGraM

O modelamento de sistemas dinâmicos usando máquinas D-Markov geralmente fornece bons resultados à medida que o valor de D cresce, à custa de um crescimento exponencial no número de estados. O algoritmo DCGraM utiliza técnicas de clusterização e minimização de grafos para reduzir o tamanho do PFSA final. O algoritmo completo possui 3 passos principais (Algoritmo 1).

1. Criação de uma máquina D-Markov: o DCGraM começa com a criação de uma máquina D-Markov essencial (G_D, π_D) para um parâmetro fixo D, tendo como entrada uma sequência S gerada por um sistema dinâmico discreto.

2. Clusterização de estados: o algoritmo *K-Means* com o parâmetro K é aplicado em $\mathcal{V}(Q)$, o conjunto de *morphs* de todos os estados de Q, agrupando aqueles com *morphs* similares.

3. Aplicação do algoritmo de minimização de grafos: o passo final é a aplicação de um algoritmo de minimização de grafos (Moore ou Hopcroft) com uma partição inicial $\mathcal{P}$, gerando um grafo reduzido com um comportamento estatístico semelhante ao da máquina D-Markov gerada a partir do sistema original.

Como os estados são agrupados apenas de acordo com os seus *morphs*, estados no mesmo *cluster* têm a mesma probabilidade de gerar símbolos, entretanto a probabilidade destes estados gerar sequências de símbolos pode diferir. Este é o critério usado pelo algoritmo de minimização para dividir estes *clusters* e gerar novas partições de estados. Na partição final, G_o, os estados na mesma classe de equivalência têm contextos a direita similares.

Os estados do grafo final são as classes de equivalência G_o e a função de probabilidade de transição é determinada calculando-se a média dos *morphs* dos estados em cada classe de equivalência G_o. O PFSA (G_o, π_o) é a saída do algoritmo.

Table 1: Algoritmo DCGraM(S, D, K)

procedure DCGRAM
    ## Passo 1: Criar Máquina D-Markov
    $(G_D,\pi_D) \leftarrow dmarkov(S,D)$
    ## Passo 2: Agrupar estados via clusterização
    $\mathcal{P} \leftarrow kmeans(K,Q,\mathcal{V}(Q))$
    ## Passo 3: Aplicar algoritmo de minimização de grafos
    $G_o \leftarrow graphMinimization(G_D, \mathcal{P})$
    $\pi_o \leftarrow averageMorphs(G_o)$
    return (G_o, π_o)
end procedure

4 Resultados

Nesta seção, a eficiência dos algoritmos D-Markov e DCGraM para a construção do PFSA é verificada para o sistema dinâmico construído a partir de um mapa caótico. Primeiramente, gera-se uma órbita de um mapa logístico de tamanho $N = 10^7$ e em seguida esta sequência é discretizada com um alfabeto ternário, formando uma sequência S de mesmo tamanho. Então, são calculadas as probabilidades de subsequências de S para criar máquinas D-Markov para diversos valores de D. Os valores de K usados no algoritmo de clusterização são obtidos empiricamente visando obter os melhores resultados. Para cada valor de D, a máquina D-Markov (G_D,π_D) é salva e sua confiabilidade é avaliada com os quantificadores definidos a seguir.

4.1 Quantificadores de Desempenho

Esta subseção apresenta dois quantificadores utilizados na comparação de desempenho dos PFSA gerados pelos algoritmos: a taxa de entropia e a divergência de Kullback-Leibler. A divergência compara as probabilidades de sequências geradas pelos PFSA com aquelas do sistema original.

4.1.1 Taxa de entropia

Seja $\{X_k\}_{k=1}^{\infty}$ um processo aleatório discreto sobre um alfabeto Σ. Sua taxa de entropia é (Cover and Thomas, 2012)

$$h \triangleq \lim_{k\to\infty} H(X_k|X_1X_2\ldots X_{k-1}) \\ -\lim_{k\to\infty}\sum_{x\in\Sigma^k} \Pr(x)\log\Pr(x_k|x_1x_2\ldots x_{k-1}). \quad (2)$$

Para um processo estacionário, a entropia condicional $H(X_k|X_1\ldots X_{k-1})$ não aumenta com k e h converge à medida que k tende a infinito (Cover and Thomas, 2012). A taxa de entropia para um PFSA estacionário é determinada por (Cover and Thomas, 2012)

$$h = -\sum_{q\in Q}\mu_q\sum_{\sigma\in\Sigma}\pi(q,\sigma)\log\pi(q,\sigma),$$

em que μ_q é a probabilidade estacionária do estado $q \in Q$. Uma vez que não é viável calcular (2) para o sistema original, a entropia condicional de ordem ℓ é adotada neste trabalho

$$h_\ell \triangleq H(X_\ell|X_1X_2\ldots X_{\ell-1}). \quad (3)$$

Esta entropia mede a incerteza da variável X_ℓ dada as $\ell-1$ variáveis passadas.

4.1.2 Divergência de Kullback-Leibler

Considere duas sequências S e S_1 sobre um alfabeto Σ geradas pelo sistema original e por um PFSA, respectivamente. Seja $\omega \in \Sigma^\ell$ uma palavra de comprimento ℓ e $P(\omega)$ e $P_1(\omega)$ as probabilidades de ω existir em S e S_1, respectivamente. Para um dado ℓ, a divergência de Kullback-Leibler de ordem ℓ é definida por

$$D_\ell = \sum_{\omega\in\Sigma^\ell} P(\omega)\log\Big(\frac{P(\omega)}{P_1(\omega)}\Big). \quad (4)$$

Apesar de não ser tecnicamente uma medida de distância, já que não obedece à desigualdade triangular e não é necessariamente comutativa, a divergência de Kullback-Leibler mede quão próximas duas distribuições são entre si. Uma pequena divergência indica que a sequência gerada pelo PFSA é estatisticamente próxima à sequência original. Na Seção 4.2 aplicaremos o DCGraM e o D-Markov para a quantização do mapa logístico proposta em (Mukherjee and Ray, 2014b), em que adotamos $\ell = 10$, uma vez que a taxa de entropia em (3) permanece inalterada a partir desse valor.

4.2 Aplicação

O mapa logístico é um sistema dinâmico representado pela equação de diferenças (Strogatz, 2001)

$$x_{k+1} \triangleq rx_k(1-x_k), \text{ for } k = 0,1,2,\ldots \quad (5)$$

em que $x_k \in \mathbb{R}$ e r é um parâmetro de controle. Este sistema apresenta um comportamento caótico para diversos valores de r (Strogatz, 2001). A sequência discreta é gerada a partir de (5) com uma condição inicial x_0 para $r = 3{,}75$, e é quantizada em três níveis: os valores de $x_k \leq 0{,}67$ são são mapeados no símbolo 0; quando $0{,}67 < x_k \leq 0{,}79$ mapeia-se no símbolo 1 e para $x_k > 0{,}79$ mapeia-se no símbolo 2. Um trecho dessa sequência ternária e seus níveis de quantização são mostrados na Figura 1.

A Figura 2 ilustra os possíveis *morphs* de uma máquina D-Markov (com $D = 6$) e o resultado da

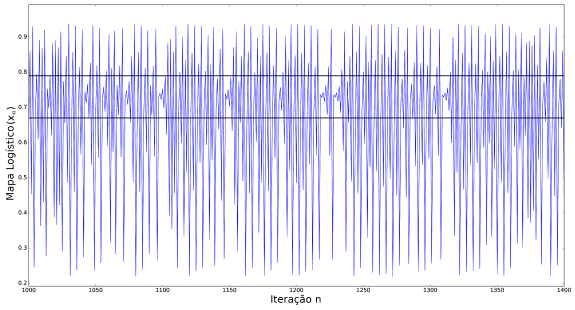

Figura 1: Níveis de quantização e amostras da sequência gerada pelo mapa logístico para $x_0 = 0,5$ e $r = 3,75$.

clusterização com $K = 5$. Nesta figura, é mostrado apenas a probabilidade do símbolo 0 versus a probabilidade do símbolo 1, uma vez que a probabilidade de símbolo 2 é determinada a partir destas probabilidades. O resultado do algoritmo de clusterização é indicado pelos centróides (marcas +) e os cinco grupos de estados estão indicados com marcadores distintos. Foram comparados valores da entropia condicional h_{10} (Figura 3) e da divergência de Kullback-Leibler D_{10} (Figura 4) para valores de D entre 4 e 9 e $K = 5$. Cada marca em uma curva corresponde a um valor de D começando com $D = 4$ na parte superior direita da figura. Na Figura 3, a curva horizontal corresponde a entropia $h_{10} = 0,47$ obtida da sequência original. O valor de $K = 5$ foi escolhido uma vez que o aumento deste valor não acarreta uma melhoria dos quantificadores considerados.

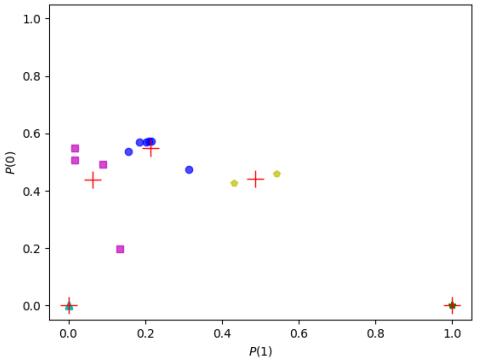

Figura 2: *Morphs* da máquina D-Markov ($D = 6$) para o mapa logístico com $K = 5$.

É possível concluir que o algoritmo proposto DCGraM gera uma PFSA com a mesma entropia $h_{10} = 0,47$ do sistema original com 28 estados ($D = 8$) enquanto a máquina D-Markov tem 104 estados para o mesmo valor de D. Portanto o PFSA obtida com o algoritmo proposto é substancialmente mais compacto que a obtida com a máquina D-Markov. Um PFSA com menos estados é obtido com um valor de D menor, embora com menor precisão. Uma comparação do número de estados obtidos pelo DCGraM e máquina D-markov é indicada na Tabela 2 para cada valor de D, juntamente com a redução percentual no número de estados.

Para a divergência de Kullback-Leibler (Figura 4), as curvas comportam-se de forma semelhante, com o valor da divergência tendendo a zero com o aumento da memória da máquina ($D_{10} = 0,003$ para $D = 8$ e $K = 5$). Observa-se uma saturação dos valores dos quantificadores para $D > 8$.

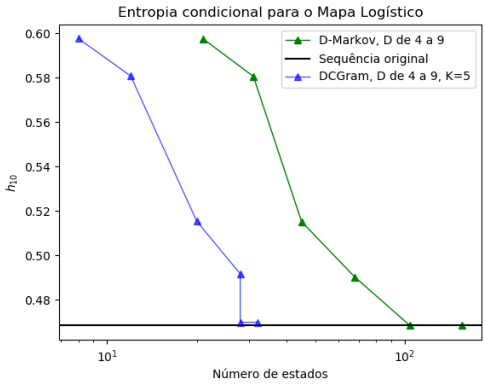

Figura 3: Entropia condicional h_{10} para a máquina D-Markov e DCGraM para diversos valores de D e $K = 5$.

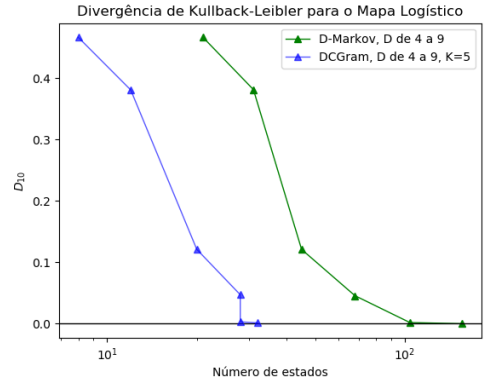

Figura 4: Divergência de Kullback-Leibler D_{10} para máquina D-Markov e DCGraM para diversos valores de D e $K = 5$.

Tabela 2: Comparação do número de estados obtido pelos algoritmos DCGraM e D-Markov para cada valor de D.

D	DCGraM	D-Markov	Redução (%)
4	8	21	61,9
5	12	31	61,3
6	20	45	55,6
7	28	68	58,9
8	28	104	73,1

5 Conclusões

O algoritmo DCGraM para geração de um FPSA foi proposto neste trabalho e envolve duas etapas de pós-processamento de uma máquina D-Markov. A primeira etapa consiste de clusterização de estados, formando-se assim um FPSA com K estados. Em seguida, um algoritmo eficiente de divisão de estados (Moore) usado em minimização de autômatos é empregado para se obter um PFSA mais preciso. Usando como estudo de caso uma sequência discreta obtida a partir de um mapa caótico, observou-se que o PFSA obtido com o DCGraM é mais compacto que a máquina D-Markov para a mesma precisão. Um trabalho em andamento consiste em testar o DCGraM para outras classes de sistemas dinâmicos bem como investigar o uso de outras técnicas de clusterização.

Referências

Berstel, J., Boasson, L., Carton, O. and Fagnot, I. (2010). Minimization of automata, *arXiv:1010.5318* .

Chattopadhyay, I., Wen, Y., Ray, A. and Phoha, S. (2011). Unsupervised inductive learning in symbolic sequences via recursive identification of self-similar semantics, *American Control Conference* .

Cover, T. M. and Thomas, J. A. (2012). *Elements of information theory*, John Wiley & Sons.

Daw, C. S., Finney, C. E. A. and Tracy, E. R. (2003). A review of symbolic analysis of experimental data, *Review of Scientific Instruments* **74**(2): 915–930.

Goodfellow, I., Bengio, Y. and Courville, A. (2016). *Deep Learning*, Adaptive computation and machine learning, MIT Press.

Graben, P. b. (2001). Estimating and improving the signal-to-noise ratio of time series by symbolic dynamics, *Phys. Rev. E* **64**: 051104.

Li, Y. and Ray, A. (2017). Unsupervised symbolization of signal time series for extraction of the embedded information, *Entropy* **19**(4).

Lin, J., J. Keogh, E., Lonardi, S. and Chiu, B. (2003). A symbolic representation of time series, with implications for streaming algorithms.

Lind, D. and Marcus, B. (1995). *An Introduction To Symbolic Dynamics and Codings*, Cambridge University Press.

MacKay, D. J. (2003). *Information theory, inference and learning algorithms*, Cambridge university press.

Makki Alamdari, M., Samali, B. and Li, J. (2015). Damage localization based on symbolic time series analysis, *Structural Control and Health Monitoring* **22**(2): 374–393.

Moreira, I., Pimentel, C., Barros, F. P. and Chaves, D. P. B. (2017). Modeling fading channels with binary erasure finite-state markov channels, *IEEE Transactions on Vehicular Technology* **66**(5): 4429–4434.

Mukherjee, K. and Ray, A. (2014a). State splitting and merging in probabilistic finite state automata for signal representation and analysis, *Signal Processing* **104**: 105–119.

Mukherjee, K. and Ray, A. (2014b). State splitting and merging in probabilistic finite state automata for signal representation and analysis, *Signal Processing* **104**: 105 – 119.

Piccardi, C. (2004). On the control of chaotic systems via symbolic time series analysis, *Chaos: An Interdisciplinary Journal of Nonlinear Science* **14**(4): 1026–1034.

Puterman, M. (2014). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, Wiley Series in Probability and Statistics, Wiley.

Rajagopalan, V. and Ray, A. (2006). Symbolic time series analysis via wavelet-based partitioning, *Signal Processing* **86**(11): 3309–3320.

Ray, A. (2004). Symbolic dynamic analysis of complex systems for anomaly detection, *Signal Processing* **84**(7): 1115–1130.

Shannon, C. E. (2001). A mathematical theory of communication, *SIGMOBILE Mob. Comput. Commun. Rev.* **5**(1): 3–55.

Strogatz, S. (2001). Nonlinear dynamics and chaos: With applications to physics, biology, chemistry and engineering.