Aprendizagem de processos semi-Markovianos generalizados: dos sistemas de eventos discretos estocásticos aos testes e à verificação

André de Matos Pedro[1⋆], Maria João Frade[1], Ana Paula Martins[2] e Simão Melo de Sousa[2⋆⋆]

[1] Universidade do Minho, Braga, Portugal,
apedro@gmx.com, mjf@di.uminho.pt,
[2] Universidade da Beira Interior, Covilhã, Portugal,
amartins@ubi.pt, desousa@di.ubi.pt

Resumo *Sistemas de eventos discretos* são uma importante subclasse de sistemas (à luz da teoria dos sistemas). Estes têm sido usados, em particular na indústria, para analisar e modelar uma grande variedade de sistemas reais, como por exemplo sistemas de produção, sistemas computacionais, sistemas de tráfego e sistemas híbridos. Este artigo propõe uma metodologia e um algoritmo para aprendizagem, através de execuções amostra, de *processos semi-Markovianos generalizados*, um modelo estocástico para sistemas de eventos discretos estocásticos. Introduz-se também uma noção de modelação, análise e verificação de sistemas contínuos e de modelos de perturbação, verificáveis no contexto de *model checking* estatístico.

1 Introdução

Sistemas de eventos discretos (DES) são sistemas largamente usados na indústria como modelos padrão para a solução de problemas de modelação que envolvam um espaço de estados discreto [2]. Normalmente os DES são usados de modo híbrido com *sistemas dinâmicos de variável contínua* (CVDS). Os CVDS evoluem de acordo com quantidades medíveis, tais como a temperatura, a pressão e a aceleração (caracterizam-se essencialmente por um espaço de estados contínuo). Ao contrário dos CVDS, os DES evoluem num espaço de estados discreto e cada mudança de estado é guiada por um evento (i.e. o espaço de estados apenas muda quando ocorre um evento). Neste contexto, este artigo propõe a exploração dos DES com maior incidência na vertente estocástica, *sistemas de eventos discretos estocásticos* (SDES), de forma a reforçar a relação entre os SDES e os *processos semi-Markovianos generalizados* (GSMP) [4,8]. Em termos práticos, as contribuições aqui apresentadas procuram abordar o problema de modelação, teste e validação de *sistemas dinâmicos* (DS), que atuam em contextos complexos.

⋆ Trabalho financiado pelo projecto Evolve (Ref[a]BI6-2010_EVOLVE_UMINHO).

⋆⋆ Trabalho parcialmente financiado pelo Laboratório de inteligência Artificial e Ciência de Computadores (LIACC) e pelo projecto FCT CANTE (Ref[a]PTPC/EIA-CCO/101904/2008).

Assim, pretende-se conseguir capturar o comportamento de fenómenos para os quais é difícil obter uma definição analítica precisa e para os quais os DS alvo devem dar resposta. Neste enquadramento, propõe-se a construção do modelo com base em medições recolhidas e pretende-se que o modelo sirva de base à geração de testes e à verificação formal. Assim, a modelação estocástica baseada em *autómatos estocásticos* (SA) é a candidata certa, em detrimento de outros modelos como exposto em [1], pelas seguintes razões: é possível definir um modelo com um número razoável de amostras; é possível dar fortes garantias na correção do processo; e é possível usar este tipo de modelos no contexto da verificação de modelos probabilísticos [8].

A classe de processos estocásticos é extensa. Como tal, a unificação desses mesmo processos, por meio de modelos mais gerais (como por exemplo modelos guiados por eventos) são uma mais valia para a análise, verificação e teste (i.e., reduz-se o uso de ferramentas e modelos diferentes). Assim, propomos um algoritmo de aprendizagem de GSMP que servirá de modelo estocástico em SDES.

A obtenção de modelos, em muitos casos, é necessária para a aprendizagem e optimização da dinâmica de sistemas contínuos e de modelos de perturbação para DS. Neste contexto, propomos uma abordagem de validação e teste para sistemas contínuos. Com o uso de processos generalizados (i.e., que seguem uma estrutura guiada por eventos), tendo em conta a fiabilidade do teste de *Kolmogorov-Smirnov* (K-S), consegue-se uma simplificação da metodologia para a obtenção da equivalência de estados relativamente a outros trabalhos [6,1,5,7]. Consegue-se também efectuar a garantia para a convergência do método K-S, de que dados os caminhos de execuções simbólicas, no limite, o nosso modelo aprendido converge para o original.

O artigo organiza-se da seguinte forma: na secção 2 será efectuada uma descrição breve das nossas definições para o processo de aprendizagem, do nosso algoritmo e da correcção do mesmo; e na secção 3 descrevem-se as aplicações práticas, conclusões e direcções para o trabalho futuro.

2 Definições e processo de aprendizagem

Definem-se nesta secção os elementos essenciais à metodologia de aprendizagem proposta. Neste contexto, requer-se a relação entre o modelo estocástico GSMP e SDES definida em [9, p. 49]. Um processo GSMP [4] analogamente à definição de autómato estocástico (SA) é um processo caracterizado essencialmente por uma estrutura baseada em eventos, onde existe uma dependência histórica da idade dos estados (i.e., necessita-se saber os tempos de espera de cada estado). No entanto, como propriedade essencial nas cadeias de Markov o estado futuro apenas depende exclusivamente do seu estado actual.

Dado $\mathcal{M} = (\mathcal{X}, \mathcal{E}, f, \Gamma, x_0, G, RV)$ o modelo de SA, onde $\mathcal{X}$ é o conjunto de estados; $\mathcal{E}$ é o conjunto de eventos; f é a função de transição associada a cada transição de estado de s para s', dado um evento e, onde $s, s' \in \mathcal{X}$ e $e \in \mathcal{E}$; Γ é uma função de eventos ativos para cada estado; x_0 é o estado inicial; G é uma estrutura de relógios estocásticos, associados a cada evento; e RV é um

conjunto de variáveis de recompensa (*reward variables*) para a análise qualitativa do modelo.

Similitude estatística. Definimos *similitude estatística de estados* (PSS) como uma relação de equivalência entre os estados. Mostra-se mais à frente que a decisão de junção de estados é, no limite, a correcta. Para que esta definição seja válida, como [6] refere, terá que ter como entrada ao processo de aprendizagem amostras estruturalmente completas. Seja $\mathcal{M} = (\mathcal{X}, \mathcal{E}, f, \Gamma, x_0, G, RV)$ um autómato estocástico que modela um GSMP, dados dois estados $s, s' \in \mathcal{X}$, define-se PSS se e somente se,

$$|\Gamma(s)| = |\Gamma(s')| \tag{1}$$

e se existe uma correspondência f, de um para um, entre os conjuntos $\Gamma(s)$ e $\Gamma(s')$, para qualquer evento $e, e' \in \mathcal{E}$, i.e.,

$$G(s, f(e)) = G(s', e') \tag{2}$$

onde $|\Gamma(s)|$ é o número de eventos ativos no estado s e G uma função de distribuição. Como exemplo, dado $|\Gamma(s)| = |\Gamma(s')| = 2, \Gamma(s) = \{a, b\}, \Gamma(s') = \{c, d\}$, verifica-se (1) (i.e., o conjunto de eventos activos de s e s' tem o mesmo tamanho), no entanto, verifica-se (2) se $G(s, a) = G(s', c)$ e $G(s, b) = G(s', d)$, ou $G(s, a) = G(s', d)$ e $G(s, b) = G(s', c)$. Contudo, o conhecimento do identificador do evento, para a definição proposta, revela-se não significativa, i.e., não se necessita conhecer, à priori, o identificador nem a distribuição associada a um determinado evento.

Aprendizagem de GSMP. Esta baseia-se, em traços gerais, numa junção de estados equivalentes (*state merge*) efectuada sobre uma árvore de prefixo construída através de execuções amostra. Referindo-se a processos com necessidade de um histórico de idades (o que não ocorre em processos com a propriedade de Markov) é necessário conseguir reconstruir os seus estados passados à custa apenas de execuções amostra. Assim, propomos o algoritmo 1, para estimar os estados passados dos relógios; o algoritmo 2, que permite concluir a similaridade entre e estados e construir um SA; e por último, a estimação dos parâmetros das distribuições. Deste modo, conseguimos aprender qualquer que seja o GSMP alvo.

O ISE verifica se ao longo do caminho n o valor de relógio actual, na amostra l, é o valor original do relógio, permitindo pré-estimar os estados da estrutura de relógios.[3] Caso não seja o valor original, reconstrói-se este valor com o auxilio de um mapeamento *map* entre a árvore de prefixo $Pr(S_{n,l})$ e os caminhos $S_{n,l}$. Assim, verificamos se o nodo predecessor ρ poderia ter inicializado um novo relógio (*new clock*) na simulação.[4]

[3] Entende-se por valor original do relógio, o valor da função de distribuição, no momento de atribuição de um novo relógio.

[4] Note-se que num GSMP a decisão de seguir uma determinada transição é efectuada segundo a competição pelo menor tempo de espera de cada evento.

Algorithm 1: Inverse scheduler estimator (ISE)

input : A set of paths $S_{n,l}$ of size $|S_n|$ and a prefix tree $Pr\,(S_{n,l})$
output: A prefix tree with original clock samples

```
for n ← 1 to |S| do                                          // For all paths n
    for l ← 2 to |S_n| do                                    // For all nodes l of path n
        for fn ← l to 1 do                                   // Decrement fn
            if ¬(ϵ(S_{n,l}) ∈ C(map(S_{n,fn}) ∧ |C(map(S_{n,fn}))| > 1 ∧
            ϵ(S_{n,fn}) ≠ ϵ(S_{n,l})) then fn ← fn + 1; break;
        if S_{n,fn} ≠ S_{n,l} then
            for t ← fn to l do                               // Estimate original clock value
                val ← val + cl(S_{n,t});
                if S_{n,t} = S_{n,l} then break;
            replace(clk(map(S_{n,l})), val) ;                // Set the estimated clock value
```

Define-se o algoritmo 2, como um algoritmo de aprendizagem de GSMP através de junção de estados (*state merge*). Esta junção é efectuada sobre uma árvore de prefixo e com o apoio do teste de K-S [3], para testar se duas distribuições distintas são ou não iguais. Encontram-se algumas diferenças relativamente a outros trabalhos relacionados [6,1], das quais se destacam: o tipo de teste usado (K-S), a construção do algoritmo de forma a ser possível paralelizar os ciclos (suportando modelos mais complexos) e uma maior abrangência dos processos estocásticos (com a consideração de um GSMP). Contudo, e de modo idêntico, segue também uma abordagem para a junção de estados equivalentes sobre uma árvore de prefixo.

O algoritmo começa por ordenar todas as palavras da árvore de prefixo $Pr\,(S_n)$, segundo o número de nodos sucessores. Aplica-se de seguida uma função de separação, em que todos os nós possíveis (com o mesmo tamanho $C(n)$, para todas as palavras n) são divididos em sub-conjuntos (grupos). Para cada um dos grupos testa-se a *similitude estatística* de cada um dos nodos existentes, através do teste K-S. De seguida, caso se verifiquem nodos similares, efectua-se uma junção determinística entre os respectivos nodos de $Pr\,(S_n)$. A função *merge* junta recursivamente todos os nodos até que não exista nenhuma transição não determinista.

Define-se por último, para concluir o processo de aprendizagem, a determinação dos parâmetros para cada uma das distribuições associadas a cada evento. Estimam-se os parâmetros aplicando o método da *estimação de máxima verosimilhança* (MLE). Por exemplo, estima-se o parâmetro λ para uma distribuição exponencial, dois parâmetros $\{\lambda, k\}$ para uma distribuição de *Weibull* e outros dois parâmetros $\{\mu, \sigma\}$ para uma distribuição *Log-Normal*. O estimador MLE é dado por $\widehat{\theta}_{MLE} = \underset{\theta \in \Theta}{arg\,max} \sum_{i=1}^{n} ln\ f\,(x_i \mid \theta)$, onde θ é um conjunto de parâmetro de f, f denota alguma das funções densidade anteriores e $(x_1, ..., x_n)$ uma amostra particular de f.

Algorithm 2: Probabilistic similarity of states (PSS)

input : A prefix tree $Pr\,(S)$ and type I error α
output: A stochastic automata

$C \leftarrow$ `Split` $(\texttt{Sort}(Pr(S)))$;
for $c \leftarrow 1$ **to** $|C|$ **do**
  for $n \leftarrow 1$ **to** $|C_c|$ **do** $nd \leftarrow next(C_{c,n})$
    while $nd \neq last(C_c)$ **do** $G \leftarrow \mathcal{C}(nd)$
      foreach a *in* $\mathcal{C}(C_{c,n})$ **do** `// Compare two children sets`
        foreach b *in* G **do** $F_{n_1} = \mathcal{T}(\varrho(a)); F_{n_2} = \mathcal{T}(\varrho(b))$
          if $\sqrt{\frac{n_1 n_2}{n_1+n_2}}\ \underset{x}{sup}\,|F_{n_1}(x) - F_{n_2}(x)| > K_\alpha$ **then** $remove(G, b)$;
      if $G = \emptyset$ **then** `// Children set` $\mathcal{C}(nd)$ `and` $\mathcal{C}(C_{c,n})$ `are similar`
        if $deg(nd) > deg(C_{c,n})$ **then** $merge(Pr(S), C_{c,n}, nd)$;
        else $merge(Pr(S), nd, C_{c,n})$;
        $remove(\mathcal{C}(C_{c,n}), nd)$;
      $nd \leftarrow next(nd)$;

De modo a garantir a correcta junção de estados no algoritmo, precisamos mostrar que no limite a junção de estados será eventualmente a correcta. Assim, necessita-se assegurar um conjunto de simulações estruturalmente completas para garantir que, no limite, sobre um grande número de amostras, temos informação suficiente para formar o nosso modelo. Este problema é conhecido como a insuficiência de dados para a criação de um modelo equivalente.[5] No contexto PSS temos que ter atenção a dois tipos de erros: erros de tipo I (α) (onde rejeitamos a equivalência dos estados, quando na verdade não deveria ser feito); e erros de tipo II (β) (onde não rejeitamos a equivalência dos estados, quando na verdade o deveríamos ter feito), por forma a garantir que o nosso algoritmo decide corretamente.

Proposição 1 *Consideremos o teste de Kolmogorov-Smirnov para duas amostras, de dimensão n_1 e n_2, ao nível de significância α. Para amostras suficientemente grandes, i.e., quando $n_1 \to \infty$ e $n_2 \to \infty$, β tende para zero.*

Apresentamos aqui um esquema da prova. A prova desta proposição baseia-se nas seguintes constatações: pelo teorema de Glivenko–Cantelli quando H_0 é verdadeira e n_1 e n_2 tendem para infinito, $\underset{x\in\mathbb{R}}{sup}\,|F_{n_1}(x) - F_{n_2}(x)|$ converge quase certamente para zero. Então, da unicidade do limite, tem-se quando H_0 é verdadeira e $n_1 \to \infty$, $n_2 \to \infty$ que $\sqrt{\frac{n_1 n_2}{n_1+n_2}}\ \underset{x\in\mathbb{R}}{sup}\,|F_{n_1}(x) - F_{n_2}(x)|$ tende quase certamente para $+\infty$. Logo, sob a validade de H_1, a probabilidade de rejeitar H_0 tende quase certamente para 1, o que permite concluir o pretendido.

Pela proposição 1 conclui-se, nos mesmos propósitos do trabalho relacionado [6], que a extensão que propomos garante, no limite, uma correta aprendizagem.

[5] Apenas com trajetórias de tamanho infinito se garante que para qualquer modelo, que o modelo aprendido poderá eventualmente convergir para um equivalente.

3 Conclusão e trabalho futuro

Num contexto de verificação de modelos (*model checking*), um GSMP tem que ser expresso numa linguagem baseada em eventos de modo a poder ser verificado. Com esse intuito, desenvolveu-se um conversor, que traduz os modelos aprendidos numa linguagem guiada por eventos, aceite pelo Ymer [8].

Numa perspectiva de aprendizagem, a aquisição de modelos de perturbação para sistemas contínuos é efectuada com sucesso, em particular nos sistemas com um espaço de estados contínuo de uma dimensão. Assim, modelos sobre condições muito específicas podem ser analisados, verificados e testados. Contudo, o processo de aprendizagem implica passos como discretização do espaço de estados, a aprendizagem estocástica de um conjunto de trajectórias discretas e a consequente regressão das simulações efectuadas do modelo (i.e., gera-se um conjunto idêntico de trajectórias às aprendidas). Por ausência de espaço, não é aqui apresentada a metodologia em detalhe, mas será reportado futuramente. Como caso prático, foram construídos modelos de perturbação de uma dimensão sobre reais, para um pêndulo invertido dado por um conjunto de simulações de uma equação diferencial de segunda ordem.

Desenvolveu-se a ferramenta *SDES framework* para Matlab, que dá suporte à simulação, modelação, aprendizagem e conversão de modelos.[6]

Está em curso a aprendizagem de um sistema contínuo de controle de uma aeronave, que será verificado usando o verificador Ymer [8] para garantir se determinadas acções são efectuadas dentro de um conjuntos de valores requeridos.

Agradecimentos

Agradece-se ao Prof. Kouamana Bousson e ao Prof. Thierry Brouard pelas conversas extremamente construtivas sobre as MC, sobre a inferência de modelos de perturbação e sobre caso de estudo aqui mencionado.

Referências

1. Carrasco, R.C., Oncina, J.: Learning deterministic regular grammars from stochastic samples in polynomial time. RAIRO, (1999)
2. Cassandras, C.G., Lafortune, S.: Introduction to Discrete Event Systems. Springer-Verlag New York, Inc., Secaucus, NJ, USA (2006)
3. DeGroot, M.: Probability and Statistics, 2nd edition. Addison Wesley (1989)
4. Glynn, P.W.: A gsmp formalism for discrete event systems. Proceedings of The IEEE 77, 14–23 (1989)
5. Kermorvant, C., Dupont, P.: Stochastic grammatical inference with multinomial tests. Proceedings of the 6th ICGI pp. 149–160., Springer-Verlag, London, UK (2002)
6. Sen, K., Viswanathan, M., Agha, G.: Learning continuous time markov chains from sample executions. Quantitative Evaluation of Systems, 146–155 (2004)
7. Wei, W., Wang, B., Towsley, D.: Continuous-time hidden markov models for network performance evaluation. Perform. Eval. 49, 129–146 (September 2002)
8. Younes, H.L.S.: Verification and planning for stochastic processes with asynchronous events. Ph.D. thesis, Pittsburgh, PA, USA (2004)
9. Zimmermann, A.: Stochastic Discrete Event Systems: Modeling, Evaluation, Applications. Springer-Verlag New York, Inc., Secaucus, NJ, USA (2007)

[6] http://desframework.sourceforge.net/