Signal Processing 104 (2014) 105–119

Contents lists available at ScienceDirect

Signal Processing

journal homepage: www.elsevier.com/locate/sigpro

State splitting and merging in probabilistic finite state automata for signal representation and analysis[☆]

Kushal Mukherjee [a,1], Asok Ray [b,*]

[a] United Technology Research Center, Cork, Ireland
[b] The Pennsylvania State University, University Park, PA, USA

ARTICLE INFO

Article history:
Received 27 July 2013
Received in revised form
24 March 2014
Accepted 28 March 2014
Available online 4 April 2014

Keywords:
Probabilistic finite state automata
Symbolic dynamics
D-Markov machines
State splitting
State merging

ABSTRACT

Probabilistic finite state automata (PFSA) are often constructed from symbol strings that, in turn, are generated by partitioning time series of sensor signals. This paper focuses on a special class of PFSA, which captures *finite history* of the symbol strings; these PFSA, called *D*-Markov machines, have a simple algebraic structure and are computationally efficient to construct and implement. The procedure of PFSA construction is based on (i) *state splitting* that generates symbol blocks of different lengths based on their information contents; and (ii) *state merging* that assimilates histories by combining two or more symbol blocks without any significant loss of the embedded information. A metric on the probability distribution of symbol blocks is introduced for trade-off between loss of information (e.g., entropy rate) and the number of PFSA states. The underlying algorithms have been validated with three test examples. While the first and second examples elucidate the key concepts and the pertinent numerical steps, the third example presents the results of analysis of time series data, generated from laboratory experimentation, for detection of fatigue crack damage in a polycrystalline alloy.

1. Introduction

Symbolic time series analysis (STSA) [1,2] is built upon the concept of symbolic dynamics [3] that deals with discretization of dynamical systems in both space and time. The notion of STSA has led to the development of a pattern recognition tool, in which a time series of sensor signals is represented as a symbol sequence that, in turn, leads to the construction of probabilistic finite state automata (PFSA) [4–9]. The paradigm of PFSA has been used for behavior modeling of dynamical systems and its applications are widespread in various fields including computational linguistics [10] and speech recognition [11]. Since PFSA models are capable of efficiently compressing the information embedded in sensor time series [12,13], these models could enhance the performance and execution speed of information fusion [14] and information source localization [15] that are often computation-intensive. Rao et al. [16], Jin et al. [17] and Bahrampour et al. [18] have shown that the performance of this PFSA-based tool as a feature extractor for statistical pattern recognition is comparable (and often superior) to that of other existing techniques (e.g., Bayesian filters, Artificial Neural Networks, and Principal Component Analysis [19]).

Statistical patterns of slowly evolving dynamical behavior in physical processes can be identified from sensor time series data [1]. Often the changes in these statistical

[☆] This work has been supported in part by the U.S. Army Research Laboratory and the U.S. Army Research Office under Grant nos. W911NF-07-1-0376 and W911NF-13-1-0461, and by the U.S. Air Force Office of Scientific Research (AFOSR) under Grant no. FA9550-12-1-0270. Any opinions, findings and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the sponsoring agencies.

[*] Corresponding author.
E-mail addresses: MukherK@utrc.utc.com (K. Mukherjee), axr2@psu.edu (A. Ray).

[1] Formerly with The Pennsylvania State University, University Park, PA, USA.

http://dx.doi.org/10.1016/j.sigpro.2014.03.045
0165-1684/© 2014 Elsevier B.V. All rights reserved.

patterns occur over a slow time scale with respect to the fast time scale of process dynamics. In this context, the concept of two time scales is succinctly presented below.

Definition 1.1 (*Fast scale*). The fast scale is defined to be a time scale over which the statistical properties of the process dynamics are assumed to remain invariant, i.e., the process is assumed to have statistically stationary dynamics at the fast scale.

Definition 1.2 (*Slow scale*). The slow scale is defined to be a time scale over which the statistical properties of the process dynamics may gradually evolve, i.e., the process may exhibit statistically non-stationary dynamics at the slow scale.

In view of Definition 1.1, statistical variations in the internal dynamics of the process are assumed to be negligible at the fast scale. Thus, sensor time series data are acquired based on the assumption of statistical stationarity at the fast scale. In view of Definition 1.2, an observable non-stationary behavior could be associated with the gradual evolution of anomalies (i.e., deviations from the nominal behavior) in the process at the slow scale. In general, a long time span at the fast scale is a tiny (i.e., several orders of magnitude smaller) interval at the slow scale. A pictorial view of the two-time-scales operation in Fig. 1 illustrates the concept.

The major steps for construction of PFSA from sensor signal outputs (e.g., time series) of a dynamical system are as follows:

(1) Coarse-graining of time series to convert the scalar or vector-valued data into symbol strings, where the symbols are drawn from a (finite) alphabet [20].
(2) Encoding of probabilistic state machines from the symbol strings [12,21].

In the process of symbol generation, the space of time series is partitioned into finitely many mutually exclusive and exhaustive cells, each corresponding to a symbol belonging to a (finite) alphabet. As a trajectory of the dynamical system passes through or touches various cells of the partition, the symbol assigned to the cell is inserted in the symbol string. In this way, a time series corresponding to a trajectory is converted into a symbol string. Fig. 2 illustrates the concept of constructing finite state automata (FSA) from time series, which provides the algebraic structure of probabilistic finite state automata (PFSA).

The next step is to construct probabilistic finite state automata (PFSA) from the symbol strings to encode their statistical characteristics so that the dynamical system's behavior is captured by the patterns generated by the PFSA in a compact form. The algebraic structure of PFSA (i.e., the underlying FSA) consists of a finite set of states that are interconnected by transitions [22–24], where each transition corresponds to a symbol in the (finite) alphabet. At each step, the automaton moves from one state to another (possibly including self loops) via these transitions, and thus generates a corresponding block of symbols so that the probability distributions over the set of all possible strings defined over the alphabet are represented in the space of PFSA. The advantage of such a representation is that the PFSA structure is simple enough to be encoded as it is characterized by the set of states, the transitions (i.e., exactly one transition for each symbol generated at a state), and the transition's probability of occurrence.

D-Markov machines are models of probabilistic languages where the future symbol is causally dependent on the (most recently generated) finite set of (at most) *D* symbols and form a proper subclass of PFSA with applications in various fields of research such as anomaly detection [12] and robot motion classification [25]. The

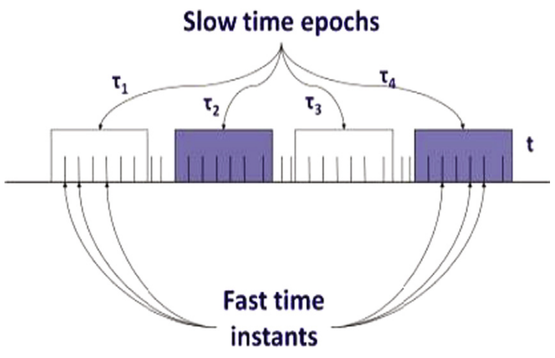

Fig. 1. Underlying concept of fast and slow time scales.

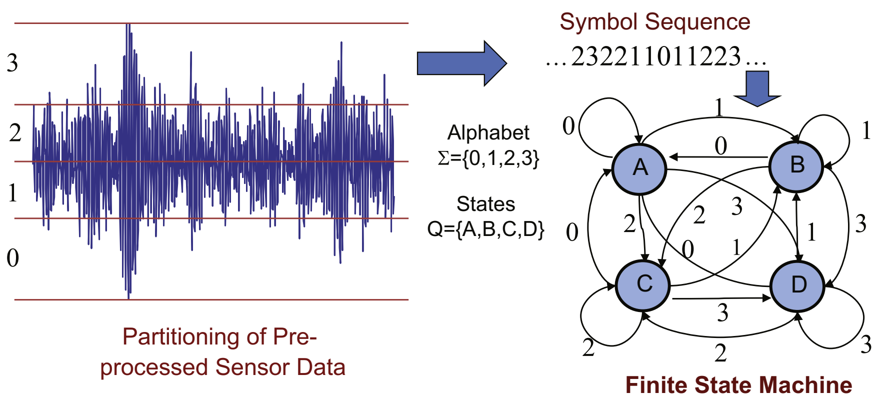

Fig. 2. Construction of probabilistic finite state automata (PFSA).

underlying FSA in the PFSA of *D*-Markov machines are deterministic, i.e., the future state is a deterministic function of the current state and the observed symbol. Therefore, *D*-Markov machines essentially encode two entities: (1) probability of generating a symbol at a given state, and (2) deterministic evolution of future states from the current state and the symbol. It is noted that if the symbol is not observed, the description of the state transition in the *D*-Markov machine becomes a probabilistic Markov chain. Furthermore, under the assumption of irreducibility, the statistically stationary distribution of states is unique and can be computed. This class of PFSA (in *D*-Markov machines) is a proper subclass of probabilistic nondeterministic finite state automata which is equivalent to Hidden Markov Models (HMMs) [4]. Even though HMMs form a more general class of models, the deterministic properties of *D*-Markov machines present significant computational advantages. For example, due to the constrained algebraic structure of *D*-Markov machines, it is possible to construct algorithms for efficient implementation and learning [26,27].

This paper makes a symbolic representation of time series signals generated from dynamical systems. Since long-range dependencies in the time series rapidly diminish under the assumption of strong mixing [28], such a dynamical system could be modeled as a *D*-Markov machine with a sufficiently large value of *D*. However, increasing the value of *D* may lead to an exponential growth in the number of machine states and hence the computational complexity of the model. The main issue addressed in this paper is order reduction of the states of a *D*-Markov machine model for representing the stationary probability distribution of the symbol strings that are generated from the times series of a dynamical system [29]. In addition, this paper addresses the trade-off between modeling accuracy and model order, represented by the number of states, for the proposed algorithm. Along this line, major contributions of the paper are conceptual development, formulation and validation of the underlying algorithms in *D*-Markov machines by

(1) Merging of (possibly redundant) states of the PFSA for state-order reduction.
(2) Retaining the *D*-Markov properties with a specified bound on the error between the constructed PFSA and the symbol string.
(3) Trade-off between model error and model complexity (i.e., number of states).
(4) Validation of the underlying concept on time series data generated from laboratory experimentation for detection of fatigue crack damage in a polycrystalline alloy.

The power of the proposed tool of PFSA-based *D*-Markov machines is its capability of real-time execution at local nodes of sensor networks for anomaly detection, pattern classification, condition monitoring, and control of diverse physical applications. It is noted that the nodes of such a sensor network may often have very limited memory and slow execution in terms of flops. This paper is organized into five sections including the introduction. Section 2 presents the preliminary concepts and a brief background on PFSA including the definition of a *D*-Markov machine. Section 3 develops the algorithms of state splitting and state merging. Section 4 presents three examples to explain and validate the algorithms. Section 5 concludes this paper with recommendations for future research.

2. Mathematical preliminaries and background

This section presents pertinent information regarding construction of *D*-Markov machines and other mathematical tools (e.g., entropy rate and metric to quantify the distance between two PFSA). The following standard definitions are recalled.

Definition 2.1 (*Finite state automaton, Hopcroft et al. [23], Sipser [24]*). A finite state automaton (FSA) G, having a deterministic algebraic structure, is a triple (Σ, Q, δ) where

- Σ is a (nonempty) finite alphabet with cardinality $|\Sigma|$;
- Q is a (nonempty) finite set of states with cardinality $|Q|$;
- $\delta: Q \times \Sigma \to Q$ is a state transition map.

Definition 2.2 (*Symbol block*). A symbol block, also called a word, is a finite-length string of symbols belonging to the alphabet Σ, where the length of a word $w \triangleq s_1 s_2 \cdots s_\ell$ with $s_i \in \Sigma$ is $|w| = \ell$, and the length of the empty word ϵ is $|\epsilon| = 0$. The parameters of FSA are extended as

- The set of all words constructed from symbols in Σ, including the empty word ϵ, is denoted as $\Sigma^\star$.
- The set of all words, whose suffix (respectively, prefix) is the word w, is denoted as $\Sigma^\star w$ (respectively, $w\Sigma^\star$).
- The set of all words of (finite) length ℓ, where $\ell > 0$, is denoted as Σ^ℓ.

Definition 2.3 (*Extended map, Hopcroft et al. [23], Sipser [24]*). The extended state transition map $\delta^\star: Q \times \Sigma^\star \to Q$ transfers one state to another through finitely many transitions such that, for all $q \in Q, \sigma \in \Sigma$ and $w \in \Sigma^\star$,

$$\delta^\star(q, \varepsilon) = q \text{ and } \delta^\star(q, w\sigma) = \delta(\delta^*(q, w), \sigma)$$

where $w\sigma$ is suffixing of the word w by the symbol σ.

Definition 2.4 (*Irreducible FSA*). An FSA G is said to be irreducible if, for all $q_1, q_2 \in Q$, there exists a word $w_{1,2} \in \Sigma^*$ such that $q_1 = \delta^*(q_2, w_{1,2})$.

Definition 2.5 (*PFSA*). A probabilistic finite state automaton (PFSA) K is a pair (G, π), where

- The deterministic FSA G is called the *underlying FSA* of the PFSA K.
- The probability map $\pi: Q \times \Sigma \to [0, 1]$ is called the morph function (also known as symbol generation probability function) that satisfies the condition: $\sum_{\sigma \in \Sigma} \pi(q, \sigma) = 1$ for all $q \in Q$.

Equivalently, a PFSA is a quadruple $K=(\Sigma, Q, \delta, \pi)$, where

- The alphabet Σ of symbols is a (nonempty) finite set, i.e., $0<|\Sigma|<\infty$, where $|\Sigma|$ is the cardinality of Σ.
- The set Q of automaton states is (nonempty) finite, i.e., $0<|Q|<\infty$, where $|Q|$ is the cardinality of Q.
- The state transition function $\delta: Q\times\Sigma\to Q$.
- The morph function $\pi: Q\times\Sigma\to[0,1]$, where $\sum_{\sigma\in\Sigma}\pi(q,\sigma)=1$ for all $q\in Q$. The morph function π generates the $(|Q|\times|\Sigma|)$ morph matrix Π.

Remark 2.1. The underlying FSA (see Definition 2.1) of the PFSA model does not make use of an initial state q_0. By the assumption of statistical stationarity in the fast scale (see Definition 1.1), the connotation here is that the stochastic process, resulting from the PFSA, started from an arbitrary state sufficiently long ago so that the current state is not affected by the initial state q_0. Furthermore, the FSA model has no accept states [23,24,4].

Remark 2.2. The structure of the PFSA model in Definition 2.5 is simpler than that used by Vidal et al. [7]. However, the PFSA model in Definition 2.5 is largely similar to the model used by Thollard et al. [30] except that Thollard et al. used an initial state q_0 and a special symbol # not belonging to Σ.

Definition 2.6 (*Extended morph function*). The morph function $\pi: Q\times\Sigma\to[0,1]$ of PFSA is extended as $\pi^*: Q\times\Sigma^*\to[0,1]$ such that, for all $q\in Q, \sigma\in\Sigma$ and $w\in\Sigma^\star$,

$$\pi^\star(q,\varepsilon)=1 \textit{ and } \pi^\star(q,w\sigma)=\pi^\star(q,w)\times\pi(\delta^\star(q,w),\sigma)$$

where $w\sigma$ is suffixing of the word w by the symbol σ.

Remark 2.3. Any state of an irreducible FSA can be reached from another state (including the originating state) in a finite number of transitions represented by a word $w_{1,2}\in\Sigma^*$. Graphically, an irreducible FSA is connected and the associated PFSA is irreducible if and only if any state can be reached from another state with non-zero probability [31]. Irreducible PFSA are widely used as language models in symbolic systems [3]. In many applications such as anomaly detection and pattern classification, the transitions between the states capture the (slow-scale) dynamical evolution of the (fast-scale) stationary system [12], where the initial condition is not important.

2.1. Symbolization of time series

This step requires partitioning (also known as quantization) of the time series data of the measured signal. The signal space is partitioned into a finite number of cells that are labeled as symbols, i.e., the number of cells is identically equal to the cardinality $|\Sigma|$ of the (symbol) alphabet Σ. As an example for the one-dimensional time series in Fig. 2, the alphabet $\Sigma=\{\alpha,\beta,\gamma,\delta\}$, i.e., $|\Sigma|=4$, and three partitioning lines divide the ordinate (i.e., y-axis) of the time series profile into four mutually exclusive and exhaustive regions. These disjoint regions form a partition, where each region is labeled with one symbol from the alphabet Σ. If the value of time series at a given instant is located in a particular cell, then it is coded with the symbol associated with that cell. As such, a symbol from the alphabet Σ is assigned to each (signal) value corresponding to the cell where it belongs. (Details are reported in [13].) Thus, a (finite) array of symbols, called a symbol string (or symbol block), is generated from the (finite-length) time series data.

The ensemble of time series data is partitioned by using a partitioning tool (e.g., maximum entropy partitioning (MEP) or uniform partitioning (UP) methods [13]). In UP, the partitioning lines are separated by equal-sized cells. On the other hand, MEP maximizes the entropy of the generated symbols and therefore, the information-rich cells of a data set are partitioned finer and those with sparse information are partitioned coarser, i.e., each cell contains (approximately) equal number of data points under MEP. In both UP and MEP, the choice of alphabet size $|\Sigma|$ largely depends on the specific data set and the allowable loss of information (e.g., leading to error of detection and classification).

2.1.1. Selection of alphabet size

Considerations for the choice of alphabet size $|\Sigma|$ include the maximum discrimination capability of a symbol sequence and the associated computational complexity. The maximum discrimination capability is characterized by the entropy of the sequence that should be maximized to the extent it is possible, or alternatively by minimizing the information loss that is denoted as the negative of the entropy. As the alphabet size is increased, there is both an increase in computational complexity and a possible reduction in loss of information; in addition, the effects of a large alphabet may become more pronounced for an insufficiently long time series [32].

In partitioning of a (one-dimensional) time series for symbolization, the alphabet size $|\Sigma|$ must be appropriately chosen in order to transform the real-valued finite-length data set $\mathbb{S}$ into a symbol string. The data set $\mathbb{S}$ is partitioned into a (finite) number of (mutually exclusive and exhaustive) segments to construct a mapping between $\mathbb{S}$ and the alphabet of symbols $\{\sigma|\sigma\in\Sigma\}$. To do so, a choice must be made as to the number of symbols, i.e., the cardinality $|\Sigma|$ of the symbol alphabet Σ. Presented below is a brief discussion on how to make the tradeoff between information loss and computational complexity.

Let the alphabet size be $k=|\Sigma|$ and the method of partitioning the time series be maximum entropy partitioning [13], i.e., a uniform probability distribution on the symbols with $P(\sigma)=1/k\ \forall\sigma\in\Sigma$. Then, the information loss, represented by the negative of the entropy [33] of the symbol sequence, is given as

$$I=-H=\Sigma_{\sigma\in\Sigma}P(\sigma)\ln P(\sigma)=-\ln k \qquad (1)$$

By representing the computational complexity as a function $g(k)$ of the alphabet size k and choosing an appropriate scalar tradeoff weighting parameter $\alpha\in(0,1)$, the cost functional to be optimized becomes

$$J(k)=-\alpha\ln k+(1-\alpha)g(k) \qquad (2)$$

The optimal alphabet size $|\Sigma|$ is obtained by solving for k in the equation $J(k+1)-J(k)=0$ along with additional constraints that may have to be imposed in the optimization procedure to realize the effects of critical issues such as any bounds on the alphabet size. As an example, if the computational cost $g(k)$ is linear in k with a constant real coefficient $c>0$, i.e., $g(k)=ck$, then the optimal alphabet size is obtained as

$$|\Sigma| = ceil\left[\frac{1}{\exp\left(\frac{1-\alpha}{\alpha}\,c\right)-1}\right] \tag{3}$$

where $ceil[\star]$ is the smallest integer greater than or equal to $\star$.

Remark 2.4. The information loss in Eq. (1) is independent of the time series data being symbolized. Furthermore, it may not always be possible to identify and parameterize a computational complexity model $g(k)$ that would unambiguously represent the class of time series data. Alternative methods for alphabet size selection, based on tools of machine learning (e.g., k-means and mixture models [19]), are potentially viable as listed in Section 5 as a topic of future research.

2.2. D-Markov machines

This subsection introduces the pertinent definitions that are necessary to construct a D-Markov machine. The PFSA model of the D-Markov machine generates symbol strings $\{s_1 s_2 \cdots s_\ell : \ell \in \mathbb{N}\ \forall s_j \in \Sigma\}$ on the underlying Markov process. The morph function π implicitly alludes to the fact that the PFSA satisfies the Markov condition, where generation of a symbol *only* depends on the current state. However, if the state is unknown, the next symbol generation may depend on the past history of the symbols generated by the PFSA. In the PFSA model, a transition from one state to another is independent of the previous history of states. Therefore, the states and transitions form a Markov process, which is a special class of Hidden Markov Models (HMMs) [7,34]. However, from the perspectives of PFSA construction from a symbol sequence, the states are implicit and generation of the next symbol may depend on the complete history of the symbol sequence. In the construction of a D-Markov machine [12], generation of the next symbol depends only on a *finite* history of at most D consecutive symbols, i.e., a symbol block of length not exceeding D. A formal definition of the D-Markov machine follows.

Definition 2.7 (*D-Markov machine, Ray [12]*). A D-Markov machine is a PFSA in the sense of Definition 2.5 and it generates symbols that solely depend on the (most recent) history of at most D symbols in the sequence, where the positive integer D is called the *depth* of the machine. Equivalently, a D-Markov machine is a statistically stationary stochastic process $S=\cdots s_{-1}s_0s_1\cdots$, where the probability of occurrence of a new symbol depends only on the last D symbols, i.e.,

$$P[s_n|\cdots s_{n-D}\cdots s_{n-1}] = P[s_n|s_{n-D}\cdots s_{n-1}] \tag{4}$$

Consequently, for $w\in\Sigma^D$ (see Definition 2.2), the equivalence class $\Sigma^\star w$ of all (finite-length) words, whose suffix is w, is qualified to be a D-Markov state that is denoted as w.

Remark 2.5. It is noted that D-Markov machines belong to the class of shifts of finite type, i.e., shift spaces that can be described by a finite set (possibly the empty set) of forbidden symbol blocks [3]. Given a probability distribution, construction of an exact PFSA model appears to be computationally infeasible. Furthermore, a D-Markov machine is a finite-history automaton, where the past information is embedded as words of length D or less. That is, if w is a word of length D or more, then $\delta^*(q,w)$ in a D-Markov machine is independent of the state q. Whatever the initial state q is, the finite sequence of transitions represented by the word $w\in\Sigma^D$ always leads to the same terminal state that could be represented by the word w itself. Consequently, in a D-Markov machine, a word $w\in\Sigma^D$ can be associated with a state of the machine. Moreover, the state transition map δ can be automatically constructed from the words that correspond to individual states. Then, the morph functions π (that are the entries of the morph matrix Π) can be estimated by counting the frequency of occurrences of the individual symbols emanating from each state [32]. Similarly, the state probability vector of a D-Markov machine can be estimated by frequency counting.

Considering the set of all symbol blocks of length D as the set of states, one may construct a D-Markov machine from a symbol sequence by frequency counting to estimate the probabilities of each transition. Since the number of states increases exponentially as the depth D is increased, state merging might be necessary for order reduction of D-Markov machines with relatively large values of D. For example, with the alphabet size $|\Sigma|=4$ (i.e., 4 symbols in the alphabet Σ) and a depth $D=5$, the D-Markov machine could have at most $|\Sigma|^D=1024$ states.

Given a finite-length symbol sequence $\mathbb{S}$ over a (finite) alphabet Σ, there exist several PFSA construction algorithms to discover the underlying irreducible PFSA model K of $\mathbb{S}$, such as causal-state splitting reconstruction (CSSR) [21], D-Markov [12,13], and Compression via Recursive Identification of Self-Similar Semantics (CRISSiS [35]). All these algorithms start with identifying the structure of the PFSA $K \triangleq (Q,\Sigma,\delta,\pi)$. Then, a $|Q|\times|\Sigma|$ count matrix C is initialized to the matrix, each of whose elements is equal to 1.

Let N_{ij} denote the number of times that a symbol σ_j is generated from the state q_i upon observing the sequence $\mathbb{S}$. The maximum a posteriori probability (MAP) estimate of the probability map for the PFSA K is computed by frequency counting as

$$\hat{\pi}_{MAP}(q_i,\sigma_j) \triangleq \frac{C_{ij}}{\sum_\ell C_{i\ell}} = \frac{1+N_{ij}}{|\Sigma|+\sum_\ell N_{i\ell}} \tag{5}$$

The rationale for initializing each element of the count matrix C to 1 is that if no event is generated at a state $q\in Q$, then there should be no preference to any particular symbol and it is logical to have $\hat{\pi}_{MAP}(q,\sigma)=1/|\Sigma|\ \forall\sigma\in\Sigma$, i.e., the uniform distribution of event generation at the state q. The above procedure guarantees that the PFSA, constructed from a (finite-length) symbol string, must have an (elementwise) strictly positive morph map Π and that the state transition map δ in Definition 2.1 is a total function.

Alternatively, the maximum likelihood estimate (MLE) of the probability map for the PFSA K is computed by frequency counting as

$$\hat{\pi}_{MLE}(q_i, \sigma_j) \triangleq \frac{N_{ij}}{\sum_{\ell} N_{i\ell}} \tag{6}$$

2.3. Entropy rate

This subsection introduces the notion of entropy rate that, given the current state, represents the predictability of PFSA.

Definition 2.8 (*Conditional entropy and entropy rate, Cover and Thomas [33]*). The entropy of a PFSA (Σ, Q, δ, π) conditioned on the current state $q \in Q$ is defined as follows:

$$H(\Sigma|q) \triangleq -\sum_{\sigma \in \Sigma} P(\sigma|q) \log P(\sigma|q)) \tag{7}$$

The entropy rate of a PFSA (Σ, Q, δ, π) is defined in terms of the conditional entropy as follows:

$$\begin{aligned} H(\Sigma|Q) &\triangleq \sum_{q \in Q} P(q) H(\Sigma|q) \\ &= -\sum_{q \in Q} \sum_{\sigma \in \Sigma} P(q) P(\sigma|q) \log P(\sigma|q) \end{aligned} \tag{8}$$

where $P(q)$ is the (unconditional) probability of a PFSA state $q \in Q$; and $P(\sigma|q)$ is the (conditional) probability of a symbol $\sigma \in \Sigma$ emanating from the PFSA state $q \in Q$.

Remark 2.6. Lower is the entropy rate $H(\Sigma|Q)$, more predictable is the machine if conditioned on the current state. As $H(\Sigma|Q)$ approaches 0, the PFSA tends to be completely deterministic.

2.4. Metric for the distance between two PFSA

This subsection introduces the notion of a metric to quantify the distance between two PFSA.

Definition 2.9 (*Metric*). Let $K_1 = (\Sigma, Q_1, \delta_1, \pi_1)$ and $K_2 = (\Sigma, Q_2, \delta_2, \pi_2)$ be two PFSA with a common alphabet Σ. Let $P_1(\Sigma^j)$ and $P_2(\Sigma^j)$ be the steady state probability vectors of generating words of length j from the PFSA K_1 and K_2, respectively, i.e., $P_1(\Sigma^j) \triangleq [P(w)]_{w \in \Sigma^j}$ for K_1 and $P_2(\Sigma^j) \triangleq [P(w)]_{w \in \Sigma^j}$ for K_2. Then, the metric for the distance between the PFSA K_1 and K_2 is defined as

$$\Phi(K_1, K_2) \triangleq \lim_{n \to \infty} \sum_{j=1}^{n} \frac{\| P_1(\Sigma^j) - P_2(\Sigma^j) \|_{\ell_1}}{2^{j+1}} \tag{9}$$

where the norm $\|\star\|_{\ell_1}$ indicates the sum of absolute values of the elements in the vector $\star$.

The norm on the right side of Eq. (9) yields

$$\|P_1(\Sigma^j) - P_2(\Sigma^j)\|_{\ell_1} \le \|P_1(\Sigma^j)\|_{\ell_1} + \|P_2(\Sigma^j)\|_{\ell_1} = 2$$

because each of the probability vectors $P_1(\Sigma^j)$ and $P_2(\Sigma^j)$ has non-negative entries that sum to 1. Furthermore, convergence of the infinite sum on the right side of Eq. (9) is guaranteed due to the weight $1/2^{j+1}$ and satisfies the relation $0 \le \Phi(\cdot,\cdot) \le 1$. It is noted that alternative forms of norms could also be used, because of norm equivalence in finite-dimensional vector spaces.

Since the metric in Definition 2.9 assigns more weight to words of smaller length, the infinite sum could be truncated to a relatively small order D (e.g., typically in the range of 5 to 20) [13] for a given tolerance $\varepsilon \ll 1$. That is, the distance $\Phi(\cdot,\cdot)$ in Eq. (9) effectively compares the probabilities of generating words of length D in two PFSA and is especially adaptable to D-Markov machines whose dynamical behavior is characterized by words of a specified maximal depth [12].

The metric $\Phi(\cdot,\cdot)$ can also be used to calculate the distance between a PFSA and a symbol string, in which case the probabilities are expressed in terms of relative frequency of occurrence of a word. In fact, it has been shown by Vidal et al. [7] that this metric quantifies the distance between probability distributions. However, this issue is not addressed in the current paper.

3. Algorithm development

This section develops the algorithms for construction of D-Markov machines. The underlying procedure consists of two major steps, namely, *state splitting* and *state merging*. In general, state splitting increases the number of states to achieve more precision in representing the information content in the time series. This is performed by splitting the states that effectively reduce the entropy rate $H(\Sigma|Q)$, thereby focusing on the critical states (i.e., those states that carry more information). Although this process is executed by controlling the exponential growth of states with increasing depth D, the D-Markov machine still may have a large number of states. The subsequent process reduces the number of states in the D-Markov machine by merging those states that have similar statistical behavior. Thus, a combination of state splitting and state merging, described in Algorithms 1 and 2, respectively, leads to the final form of the D-Markov machine.

3.1. Development of the state splitting algorithm

In D-Markov machines, a symbol block of (finite) length D is sufficient to describe the current state. In other words, the symbols that occur prior to the last D symbols do not affect the subsequent symbols observed. Therefore, the number of states of a D-Markov machine of depth D is bounded by $|\Sigma|^D$, where $|\Sigma|$ is the cardinality of the alphabet Σ. As this relation is exponential in nature, the number of states rapidly increases as D is increased. However, from the perspective of modeling a symbol string, some states may be more important than others in terms of their embedded information contents. Therefore, it is advantageous to have a set of states that correspond to symbol blocks of different lengths. This is accomplished by starting off with the simplest set of states (i.e., $Q = \Sigma$ for $D = 1$) and subsequently splitting the current state that results in the largest decrease of the entropy rate.

The process of splitting a state $q \in Q$ is executed by replacing the symbol block q by its *branches* as described by the set $\{\sigma q : \sigma \in \Sigma\}$ of words. Maximum reduction of the entropy rate is the governing criterion for selecting the state to split. In addition, the generated set of states must satisfy the self-consistency criterion, which only permits a unique transition to emanate from a state for a given symbol. If $\delta(q, \sigma)$ is not unique for each $\sigma \in \Sigma$, then the state q is split further. In the state splitting algorithm, a stopping rule is constructed by specifying the threshold parameter η_{spl} on the rate of decrease of conditional entropy. An alternative stopping rule for the algorithm is to provide a maximal number of states N_{max} instead of the threshold parameter η_{spl}. The operation of state splitting is described in Algorithm 1.

Algorithm 1. State splitting.

```
Input: Symbol sequence $s_1 s_2 s_3 \cdots$.. , where each $s_i$ belongs
  to the symbol alphabet $\Sigma$
User defined parameters: Maximum number
  of states $N_{max}$ and threshold $\eta_{spl}$ for state splitting
Output: PFSA $K = \{\Sigma, Q, \delta, \pi\}$
Initialize: Create a 1-Markov machine $\tilde{Q} := \Sigma$
repeat
  $Q := \tilde{Q}$;
  $\tilde{Q} = \arg\min_{Q'} H(\Sigma|Q')$;
  where $Q' = (Q \backslash \{q\}) \cup (\{\sigma q : \sigma \in \Sigma\})$ and $q \in Q$
until $|\tilde{Q}| < N_{max}$ or $H(\Sigma|Q) - H(\Sigma|\tilde{Q}) < \eta_{spl}$
for all $q \in \tilde{Q}$ and $\sigma \in \Sigma$ do
  if $\delta(q, \sigma)$ is not unique then
    $\tilde{Q} := (\tilde{Q} \backslash \{q\}) \cup (\{\sigma q : \sigma \in \Sigma\})$;
  end if
end for
return $K = \{\Sigma, Q, \delta, \pi\}$
```

Remark 3.1. Other distances such as the KL-divergence [33] between the distribution of symbols could also be used before and after state splitting. This procedure is conceptually similar to variable length n-grams [36] used in language modeling, where a tree structure is implemented to allow contexts with different lengths.

Let q be a D-Markov state (see Definition 2.7), which is split to yield new states σq, where $\sigma \in \Sigma$ and σq represents the equivalence class of all (finite-length) symbol strings with the word σq as the suffix. Fig. 3 illustrates the process of state splitting in a PFSA with alphabet $\Sigma = \{0, 1\}$, where each terminal state is circumscribed by an ellipse. For example, the states in the third layer from the top are $00q$, $10q$, $01q$, and $11q$, of which all but $10q$ are terminal states. Consequently, the state *10q* is further split as *010q* and $110q$ that are also terminal states, i.e, $Q = \{00q, 01q, 11q, 010q, 110q\}$, as seen in the split PFSA diagram of Fig. 3. Given the alphabet Σ and the associated set Q of states, the morph matrix Π can be computed in the following way:

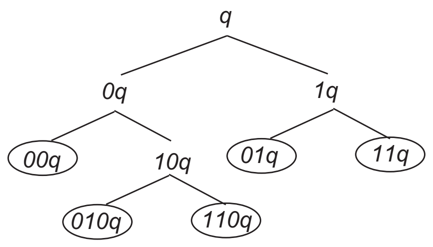

Fig. 3. Tree-representation of state splitting in D-Markov machines.

$$\pi(q, \sigma) = P(\sigma|q) = \frac{P(q\sigma)}{P(q)}, \quad \forall \sigma \in \Sigma, \ \forall q \in Q \tag{10}$$

where $P(\cdot)$ and $P(\cdot|\cdot)$ are the same as those used in Definition 2.8 and Eq. (8) therein.

For construction of PFSA, each element $\pi(\sigma, q)$ of the morph matrix Π is estimated by frequency counting as the ratio of the number of times, $N(q\sigma)$, the state q is followed (i.e., suffixed) by the symbol σ and the number of times, $N(q)$, the state q occurs. By using the structures of Eqs. (5) and (6), it follows from Eq. (10) that each element $\hat{\pi}(\sigma, q)$ of the estimated morph matrix $\widehat{\Pi}$ is obtained as

$$\hat{\pi}_{MAP}(q, \sigma) \triangleq \frac{1 + N(q\sigma)}{|\Sigma| + N(q)} \quad \text{and} \quad \hat{\pi}_{MLE}(q, \sigma) \triangleq \frac{N(q\sigma)}{N(q)}, \quad \forall \sigma \in \Sigma, \ \forall q \in Q \tag{11}$$

where $\sum_{\sigma \in \Sigma} \hat{\pi}_{MAP}(\sigma, q) = 1$ and $\sum_{\sigma \in \Sigma} \hat{\pi}_{MLE}(\sigma, q) = 1, \ \forall q \in Q$.

Similar to Eq. (11) and following the structures of Eqs. (5) and (6), each element $P(q)$ of the stationary state probability vector is estimated by frequency counting as

$$\widehat{P}_{MAP}(q) \triangleq \frac{1 + N(q)}{|Q| + \sum_{q' \in Q} N(q')} \quad \text{and} \quad \widehat{P}_{MLE}(q) \triangleq \frac{N(q)}{\sum_{q' \in Q} N(q')}, \quad \forall q \in Q \tag{12}$$

where $\widehat{P}(q)_{MAP}$ (respectively, $\widehat{P}(q)_{MLE}$) is an element of the estimated stationary state probability vector, which implies the estimated stationary probability of the PFSA being in the state $q \in Q$. Wen et al. [32] have statistically modeled the error of estimating the state probability vector from finite-length symbol strings.

Now the entropy rate (see Eq. (8) in Section 2.3) is computed in terms of the elements of estimated state probability vector and estimated morph matrix as

$$\begin{aligned} H(\Sigma|Q) &= -\sum_{q \in Q} \sum_{\sigma \in \Sigma} P(q) P(\sigma|q) \log P(\sigma|q) \\ &\approx -\sum_{q \in Q} \sum_{\sigma \in \Sigma} \widehat{P}(q) \hat{\pi}(q, \sigma) \log \hat{\pi}(q, \sigma) \end{aligned} \tag{13}$$

Remark 3.2. Under limited availability of data, smoothing algorithms may be used to estimate the morph matrix, state probability vector, and entropy rate. Such techniques have been implemented for probabilistic suffix trees and variable-order Markov chains [37]; however, for sufficiently long time series data, there would be no need for smoothing. It is also noted that, for MAP estimation of the state probability vector, the conditions $\sum_{q \in Q} \widehat{P}_{MAP}(q) = 1$ and $\widehat{P}_{MAP}(q) \in (0, 1), \ \forall q \in Q$ satisfy criteria for an irreducible PFSA [31]; in contrast, for MLE estimation, these conditions become $\sum_{q \in Q} \widehat{P}_{MLE}(q) = 1$ and $\widehat{P}_{MLE}(q) \in [0, 1], \ \forall q \in Q$. However, for sufficiently long time series data, the MAP and MLE estimates of the state probability vector in Eq. (12) tend to converge. Similarly, the probability maps in Eqs. (5) and (6) tend to converge for sufficiently long data. The the MAP probability map has been used in each of the three examples in Section 4, because of very long data sets.

3.2. Development of the state merging algorithm

Once state splitting is performed, the resulting D-Markov machine is a statistical representation of the symbol string under consideration. Depending on the choice of alphabet size $|\Sigma|$ and depth D, the number of states after splitting may run into hundreds. Although increasing the number of states of the machine may lead to a better representation of the symbol string, it rapidly increases the execution time and memory requirements. The motivation behind the state merging is to reduce the number of states, while preserving the D-Markov structure of the PFSA. Of course, such a process may cause the PFSA to have degraded precision due to loss of information. The state merging algorithm aims to mitigate this risk.

In the state merging algorithm, a stopping rule is constructed by specifying an acceptable threshold η_{mrg} on the distance $\Phi(\cdot,\cdot)$ between the merged PFSA and the PFSA generated from the original time series. Before embarking on the state merging algorithm, the procedure for merging of two states is described below.

3.2.1. Notion of merging two states

The process of state merging is addressed by creating an equivalence relation [38], denoted as $\sim$, between the states. The equivalence relation specifies which states are identified to belong to the same class, thereby partitioning the original set of states into a smaller number of equivalence classes of states, each being a nonempty collection of the original states. The new states are, in fact, equivalence classes as defined by $\sim$.

Let $K_1 = \{\Sigma, Q_1, \delta_1, \pi_1\}$ be the *split PFSA*, and let $q, q' \in Q_1$ be two states that are to be *merged* together. Initially, an equivalence relation is constructed, where none of the states are equivalent to any other state except itself, i.e., each equivalence class is represented by a singleton set. To proceed with the merging of states q and q', an equivalence relation is imposed between q and q', denoted as $q \sim q'$; however, the transitions between original states may not be well-defined anymore, in the following sense: there may exist $\sigma \in \Sigma$ such that the states $\delta_1(q,\sigma)$ and $\delta_2(q',\sigma)$ are not equivalent. In essence, the same symbol may cause a transition to two different states from the merged state $\{q, q'\}$. As the structure of D-Markov machines does not permit this ambiguity of non-determinism [12], the states $\delta_1(q,\sigma)$ and $\delta_2(q',\sigma)$ are also *required* to be merged together, i.e., $\delta_1(q,\sigma) \sim \delta_2(q',\sigma)$ (this procedure is known as *determinization* in the state merging literature). Therefore, the symbol σ will cause a transition from the merged state $\{q,q'\}$ to the merged state $\{\delta_1(q,\sigma), \delta_2(q',\sigma)\}$. This process is recursive and is performed until no ambiguity in state transitions occurs. Indeed at each iteration, the number of states of the future machine is reduced, and the machine where all the states are merged is always consistent. Therefore, the number of states is a decreasing sequence of positive integers, which must eventually converge. The recursive operation of the equivalence relation $\sim$ is described in Algorithm 2.

Algorithm 2. Minimal equivalence relation given $q \sim q'$.

```
Input: $\delta$, $q$, $q'$, Initial equivalence relation $\sim$
Output: Updated equivalence relation $\sim$
NOTE: Recursive function $(\sim)$:=Merge$(\delta, q, q', \sim)$
Set $q \sim q'$;
for all $\sigma \in \Sigma$ do
  if $\delta(q,\sigma) \nsim \delta(q',\sigma)$ then
    Set $\sim$:=Merge$(\delta, \delta(q,\sigma), \delta(q',\sigma), \sim)$;
  end if
end for
return $\sim$
```

Let $K_1 = \{\Sigma, Q_1, \delta_1, \pi_1\}$ be the *split PFSA* that is merged to yield the reduced-order PFSA $K_2 = \{\Sigma, Q_2, \delta_2, \pi_2\}$, where the state-transition map δ_2 and the morph function π_2 for the merged PFSA K_2 are defined on the quotient set $Q_2 \triangleq Q_1/\sim$, and $[q] \in Q_2$ is the equivalence class of $q \in Q_1$. Then, the associated morph function π_2 is obtained as

$$\begin{aligned}\pi_2([q],\sigma) &= P\left[s_{i+1}=\sigma \,|\, \bigcup_{\tilde{q}\in[q]} \{X_i=\tilde{q}\}\right] \\ &= \frac{\sum_{\tilde{q}\in[q]} P[s_{i+1}=\sigma; X_i=\tilde{q}]}{\sum_{\tilde{q}\in[q]} P(X_i=\tilde{q})} \\ &= \frac{\sum_{\tilde{q}\in[q]} P[s_{i+1}=\sigma | X_i=\tilde{q}]P(X_i=\tilde{q})}{\sum_{\tilde{q}\in[q]} P(X_i=\tilde{q})} \\ &\approx \frac{\sum_{\tilde{q}\in[q]} \hat{\pi}_1(\tilde{q},\sigma) \times \widehat{P}_1(\tilde{q})}{\sum_{\tilde{q}\in[q]} \widehat{P}_1(\tilde{q})} \end{aligned} \quad (14)$$

As seen in Eq. (14), the morph function π_2 of the merged PFSA K_2 is estimated as the sum of $\hat{\pi}_1$ weighted by the stationary state probabilities $\widehat{P}_1$ of the PFSA K_1. By construction, δ_2 is naturally obtained as

$$\delta_2([q],\sigma) = [\delta_1(q,\sigma)] \quad (15)$$

Algorithm 3 presents the procedure to obtain the PFSA, where the objective is to merge the states q and q'.

Algorithm 3. Minimal PFSA K_2 after merging of two states.

```
Input: $K_1 = \{\Sigma, Q_1, \delta_1, \pi_1\}$, $q$, $q'$
Output: Merged PFSA $K_2 = \{\Sigma, Q_2, \delta_2, \pi_2\}$
Compute the equivalence relation $\sim$ using Algorithm 2;
Set $Q_2$:=$Q_1/\sim$; % $Q_2$ is the quotient set of $Q_1$ under $\sim$
Compute the stationary-probability vector $\widehat{P}_1$ of the PFSA $K_1$;
for all $[q] \in Q_2$ do
  for all $\sigma \in \Sigma$ do
    Set $\delta_2([q],\sigma)$:=$[\delta_1(q,\sigma)]$;
    Compute $\pi_2([q],\sigma)$ using Eq. (14);
  end for
end for
return $K_2 = \{\Sigma, Q_2, \delta_2, \pi_2\}$
```

3.2.2. Identification of the states to be merged

The next task is to decide which states have to be merged. States that behave similarly (i.e., have similar morph probabilities) have a higher priority for merging. The similarity of two states, $q, q' \in Q$, is measured in terms of morph functions (i.e., conditional probabilities) of future symbol generation as the distance between the two rows of the estimated morph matrix $\widehat{\Pi}$ corresponding to the states q and q'. The ℓ_1-norm (i.e., the sum of absolute values of the vector components) has been adopted to be the distance function as

$$\mathcal{M}(q,q') \triangleq \|\hat{\pi}(q,\cdot)-\hat{\pi}(q',\cdot)\|_{\ell_1} = \sum_{\sigma \in \Sigma} |\hat{\pi}(q,\sigma)-\hat{\pi}(q',\sigma)| \quad (16)$$

A small value of $\mathcal{M}(q,q')$ indicates that the two states have close probabilities of generating each symbol. Note that this measure is bounded above as $\mathcal{M}(q,q') \le 2\ \forall q,\ q' \in Q$, because $0 \le \sum_{\sigma \in \Sigma}\hat{\pi}(q,\cdot) \le 1$ and $0 \le \sum_{\sigma \in \Sigma}\hat{\pi}(q',\cdot) \le 1$. Now the procedure of state merging is briefly described below.

First, the two closest states (i.e., the pair of states $q, q' \in Q$ having the smallest value of $\mathcal{M}(q,q')$) are merged using Algorithm 3. Subsequently, distance $\Phi(\cdot,\cdot)$ (see Eq. (9) in Section 2.4) of the merged PFSA from the initial symbol string is evaluated. If $\Phi < \eta_{mrg}$ where η_{mrg} is a specified threshold, then the machine structure is retained and the states next on the priority list are merged. On the other hand, if $\Phi \ge \eta_{mrg}$, then the process of merging the given pair of states is aborted and another pair of states with the next smallest value of $\mathcal{M}(q,q')$ is selected for merging. This procedure is terminated if no such pair of states exists, for which $\Phi < \eta_{mrg}$. The operation of the procedure is described in Algorithm 4.

Remark 3.3. In the state merging algorithm, each of the two metrics, $M(q,q')$ in Eq. (16) and $\Phi(K_1,K_2)$ in Eq. (9), serves its own purpose. While $M(q,q')$ represents the distance between the distributions of symbols at two states q and q' of a PFSA, independent of the respective future distributions, $\Phi(K_1,K_2)$ is a metric defined between two PFSA, which takes into account all future distributions of symbols. If $\Phi(K_1,K_2)=0$, then K_1 and K_2 are the same PFSA. Denoting K_1 as the unmerged PFSA and K_2 as the merged PFSA, it is possible to use the metric $\Phi(K_1,K_2)$ directly for state merging. Since the computation of Φ is expensive, the metric $M(q,q')$ should be used as an initial check of the feasibility of merging two states to save (possibly) unnecessary computations of Φ. A small value of $M(q,q')$ does not guarantee that $\Phi(K_1,K_2)$ is small. However, a large value of $M(q,q')$ does guarantee that $\Phi(K_1,K_2)$ is large.

Algorithm 4. State merging in PFSA.

```
Input: PFSA $K=(\Sigma,Q,\delta,\pi)$ and symbol sequence $\{s_i\}$
User defined parameter: Threshold $\eta_{mrg}$ for state merging
Output: Merged PFSA $K_m=(\Sigma,Q_m,\delta_m,\pi_m)$
Set $K_m$:=$K$;
for all $q,q' \in Q_m$ do
  if $q \ne q'$ then
    Set LIST_STATES $(q,q')=\mathcal{M}(q,q')$ using Eq. (16);
  else
    Set LIST_STATES $(q,q')=2$;
  end if
end for
sort (LIST_STATES); % Place the pair $(q,q')$ with the smallest $\mathcal{M}(q,q')$
    on top of the list
Set $(q,q')$:=pop (LIST_STATES); % Select the pair $(q,q')$ that is on top
    of the sorted list
loop
  Compute $K_1$ from $K_m$ by merging the states $q$ and $q'$ via
    Algorithm 3;
  if $d[K_1,\{s_i\}] < \eta_{mrg}$ then
    Set $K_m$:=$K_1$;
    Recompute LIST_STATES;
    Set $(q,q')$:=pop (LIST_STATES);
  else
    Set $(q,q')$:=pop (LIST_STATES);
  if $q==q'$ then
      Break loop;
    end if
  end if
end loop
return $K_m=(\Sigma,Q_m,\delta_m,\pi_m)$
```

Remark 3.4. Given a data set of finite length, while the constructed PFSA may yield degraded performance after state merging due to loss of information, the performance may also be increased because the frequency counting estimation of probability parameters could be improved due to possible reduction in the number of states.

4. Examples

This section presents three examples. In the first example, a symbolic sequence is generated from a non-D-Markov PFSA and the symbol string under consideration is modeled as a reduced-order D-Markov machine. In the second example, time series data, generated from a chaotic dynamical system, is partitioned. The resulting symbolic string is modeled as a D-Markov machine by using the algorithms developed in Section 3. The third example is based on experimental data collected from a laboratory apparatus for detection of fatigue crack damage in a polycrystalline alloy. It demonstrates the efficacy of the algorithms for order reduction of PFSA models in a physical process. The following procedures have been adopted in these examples.

- For state splitting, the stopping rule in Algorithm 1 is based on the maximal number of states (N_{max}) instead of the threshold parameter (η_{spl}).
- The MAP probability has been used for estimation of the morph probabilities (see Eq. (5)) and state probabilities (see Eq. (12)). Referring to Remark 3.2, since the data length in each example is sufficiently long, the results obtained by using either $\hat{\pi}_{MAP}(\cdot,\cdot)$ (see Eq. (5)) or $\hat{\pi}_{MLE}(\cdot,\cdot)$ (see Eq. (6)) are expected to be numerically similar.

Example 4.1 (*Symbol string model from a non-D-Markov PFSA*). The PFSA K_0 in Fig. 4 is a variation of the *even shift* machine [3] with three symbols, i.e., $\Sigma=\{0,1,2\}$, which is not a shift of finite type because the minimal set of forbidden strings has a subset $\{01^{(2k-1)}0,\ 02^{(2k-1)}0:\ k \in \mathbb{N}\}$ that does not have a finite cardinality. Hence, K_0 ia a non-D-Markov machine. A data set of 1,000,000 points is generated from K_0 and Algorithm 1 for state splitting is used to obtain a D-Markov PFSA (K_1) with depth D=8 and alphabet size $|\Sigma|=3$ from the symbol string that is obtained by partitioning the data set. Note that, without sequential state splitting, the PFSA would have at most $|\Sigma|^D=3^8=6561$ possible states. The selected parameter for state splitting (see Algorithm 1) is $N_{max}=32$ and that for state merging (see Algorithm 4) is $\eta_{mrg}=0.010$.

The evolution of the entropy rate during state splitting is presented in Fig. 5. Moreover, the entropy rate of K_1

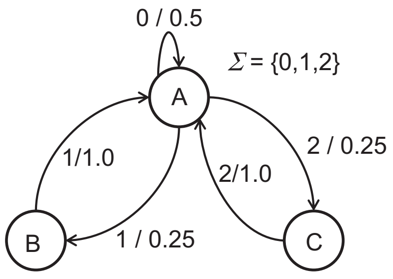

Fig. 4. The PFSA K_0 to generate the symbol sequences in Example 4.1.

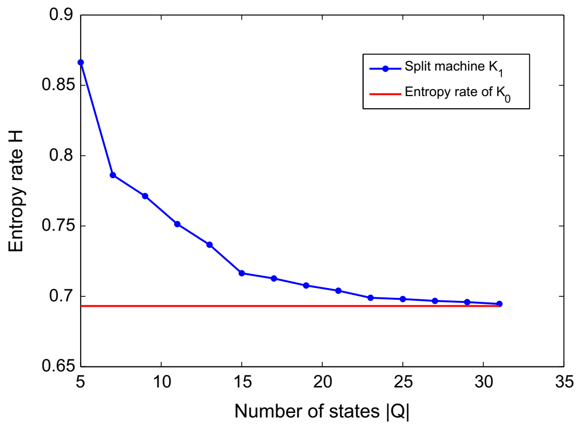

Fig. 5. Monotonic decrease of entropy rate in Example 4.1.

approaches that of the initial machine K_0 as the number of states increases. This is evident from the fact that K_1 would represent K_0 better if the number of states is increased. Algorithm 1 splits the states $111\cdots1$ and $222\cdots2$ that are the non-synchronizing words[2] of the PFSA K_0. State splitting is continued until the probability of being in one of these states becomes very low. The state merging algorithm (see Algorithm 4 in Section 3.2.2) is used to construct a reduced-order D-Markov PFSA (K_2).

Although there is a loss of information by modeling K_0 as K_2, the results in Fig. 5 and Table 1 show that this loss is monotonically decreasing. The D-Markov machine has an alphabet size $|\Sigma|=3$ and a depth $D=8$; its order is diminished to only 11 states from $3^8=6561$. This example illustrates how a non-D-Markov PFSA can be modeled by a significantly reduced order D-Markov machine by making a trade-off between the number of D-Markov states and the modeling error. Table 1 summarizes the pertinent results.

Example 4.2 (*Time series from a chaotic system*). The time series $\{x_k\}$ is generated iteratively from the logistic map [28,39]:

$$x_{k+1} \triangleq rx_k(1-x_k) \tag{17}$$

The onset of chaos for the logistic map [28] in Eq. (17) begins from $r\approx3.57$. In this example, the initial state is chosen as $x_0 \triangleq 0.5$ and the parameter $r=3.75$. The iterations x_k always lie in the interval [0, 1]. The space [0, 1] is partitioned into three mutually disjoint intervals that are associated with the symbols 0, 1 and 2. The boundaries of disjoint intervals are shown in horizontal dashed lines in Fig. 6. A symbol string with symbols in $s_k \in \{0,1,2\}$ is generated by replacing the value of x_k by the corresponding symbol associated with the cell of the partition within which x_k lies. The symbol sequence is assumed to be statistically stationary, thereby permitting the use of PFSA to model its stochastic behavior.

[2] Non-synchronizing words are symbol blocks that do not uniquely determine the current state of the PFSA.

Table 1
Results of Example 4.1.

PFSA	$\Phi(\cdot,K_0)$	$\Phi(\cdot,\{s_n\})$	$\|Q\|$	depth
K_0	0	0.0010	3	non D-Markov
K_1	0.0009	0.0003	31	8
K_2	0.0088	0.0085	11	8

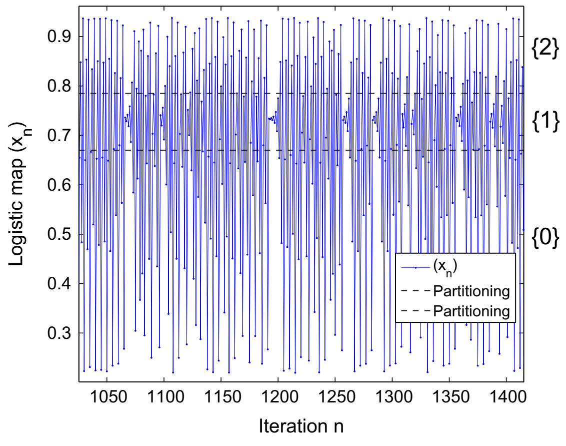

Fig. 6. Output $\{x_n\}$ of the logistic map in Example 4.2.

The logistic map exhibits two types of behaviors as seen in Fig. 6. The first behavior is represented as oscillations between a high value and a low value, and the second behavior is represented as oscillations of small amplitude around 0.73. Table 2 shows an excerpt of the symbol sequence, where the two behaviors are well reproduced in terms of '0202' blocks alternating with '11' blocks.

The selected parameters for state splitting (see Algorithm 1) and state merging (see Algorithm 4) are $N_{max}=55$ and $\eta_{mrg}=0.005$, respectively. The pertinent results of this state merging are presented in Table 3. It is seen that the merging step allows reduction of the number of states from 55 to 8, represented by PFSA K_1 and K_2.

Tables 4, 5 and 6 respectively present the lists of merged states, state transition map $\delta: Q\times\Sigma\to Q$ (see Definition 2.1, and the estimated morph matrix $\hat{\pi}_{MAP}(\cdot,\cdot)$ (see Eq. (5)) to elucidate how the operations of state splitting and subsequent merging take place in the 11-state merged PFSA K_2 in the third row of Table 3. It is noted that each ε_i, i=2, 4, 6, 8, 9 in Table 6 is in the order of N^{-1}, i,e,, $\varepsilon_i \sim O(N^{-1})$, where N is number of time series data points. Since $N=10^6$ in this

Table 2
Excerpt of the symbol sequence in Example 4.2$\{s_n\}$.

…2 0 1 1 0 2 0 2 0 1 1 1 1 1 0 2 0 2 0 1 0 2 0 2 0 0 2 0 2 0 1 2 0 2 0 2 0 2 0 0 2 0 2 0 1 2 0 2 0 1 1 1 2 0 2 0 2 0 2 0 0 2 0 2 0 0 2 0 2 0 1 2 0 2 0
1 2 0 2 0 1 1 1 1 1 1 2 0 2 0 1 1 1 1 1 1 2 0 2 0 1 0 2 0 2 0 0 2 0 2 0 1 2 0 2 0 2 0 2 0 0 2 0 2 0 0 2 0 2 0 0 2 0 2 0 0 2 0 2 0 1 1 0 2 0 2 0 1 1 1 2 0 2
0 1 2 0 2 0 1 2 0 2 0 1 1 1 …

Table 3
Results of Example 4.2.

PFSA	$\Phi(\cdot,(s_n))$	$\lvert Q\rvert$	depth
K_1	0.00012	55	10
K_2	0.00269	8	6

Table 4
The states of the merged 11-state PFSA in Example 4.2.

State	Corresponding suffix of symbol string
A	{0, 2111111}
B	{01, 21111111}
C	{011, 0111, 01111, 011111, 0111111, 01111111, 11111111}
D	{211111}
E	{21111}
F	{2111}
G	{211}
H	{21}
I	{02, 12}
J	{022, 122}
K	{222}

Table 5
State transition map (δ) for the 11-state PFSA in Example 4.2.

Symbols	0	1	2
States			
A	A	B	I
B	A	C	I
C	A	C	I
D	A	A	I
E	A	D	I
F	A	E	I
G	A	F	I
H	A	G	I
I	A	H	J
J	A	H	K
K	A	H	K

Table 6
Morph matrix for 11-state PFSA in Example 4.2.

Symbols	0	1	2
States			
A	0.5	0.25	0.25
B	ε_2	$1-2\varepsilon_2$	ε_2
C	0.4	0.4	0.2
D	ε_4	$1-2\varepsilon_4$	ε_4
E	0.5	0.25	0.25
F	ε_6	$1-2\varepsilon_6$	ε_6
G	0.5	0.25	0.25
H	ε_8	$1-2\varepsilon_8$	ε_8
I	ε_9	ε_9	$1-2\varepsilon_9$
J	0.5	0.25	0.25
K	0.25	0.125	0.625

where each $\varepsilon_i > 0$ for $i = 2,4,6,8,9$ is in the order of 10^{-6} because the number of time series data points is 1,000,000 in this example.

example, the approximation of $\varepsilon_i \approx 0$ would yield numerically similar results as if the estimated morph matrix $\hat{\pi}_{MLE}(\cdot,\cdot)$ was used instead of the estimated morph matrix $\hat{\pi}_{MAP}(\cdot,\cdot)$.

Example 4.3 (*Order reduction in PFSA-based fatigue damage modeling in a polycrystalline alloy*). This example addresses the construction of a PFSA model and the associated model order reduction based on time series of ultrasonic signals for fatigue damage detection in mechanical systems [40]. The rationale for using the PFSA model to predict the fatigue damage-induced anomalous behavior of structural materials (e.g., polycrystalline alloys) is briefly presented below.

Following Fig. 1 in Section 1 and referring to Definitions 1.1 and 1.2, anomalous behavior of fatigue damage dynamics can be identified as a two-time-scale model because the underlying physical process has stationary dynamics on the fast time scale of ultrasonic waves and the observable non-stationary slow-time scale evolution of fatigue damage is associated with the gradual changes in the material deformation (e.g., grain dislocation). In the context of early detection of fatigue damage in structural materials, statistically stationary load fluctuations may take place on the fast time scale. The resulting damage evolution and anomaly growth may occur on the slow-time scale only; in essence, the fatigue damage behavior can be safely assumed to be statistically invariant on the fast time scale [40,41]. It is also noted that PFSA-based *D*-Markov machine models are suitable for real-time execution on in situ microprocessors as reported in earlier publications [16–18].

Fig. 7(a) shows a picture of the experimental apparatus that is built upon a computer-instrumented and computer-controlled uniaxial fatigue testing machine. The apparatus is instrumented with ultrasonic flaw detectors and an optical traveling microscope; the details of the operating procedure of the fatigue test apparatus and its instrumentation and control system are reported in [40]. Tests have been conducted using center-notched 7075-T6 aluminum specimens (see Fig. 7(b)) under a periodically varying load, where the maximum and minimum (tensile) loads were kept constant at 87 MPa and 4.85 MPa at 12.5 Hz frequency. Each specimen is 3 mm thick and 50 mm wide, and has a slot of 1.58 mm × 4.5 mm at the center. The central notch increases the stress concentration factor that ensures crack initiation and propagation at the notch ends [40]. The ultrasonic sensing device is triggered at a frequency of 5 MHz at each peak of the cyclic load. The time epochs, at which data are collected, are chosen to be 1000 load cycles (i.e., ~80 s) apart. At the beginning of

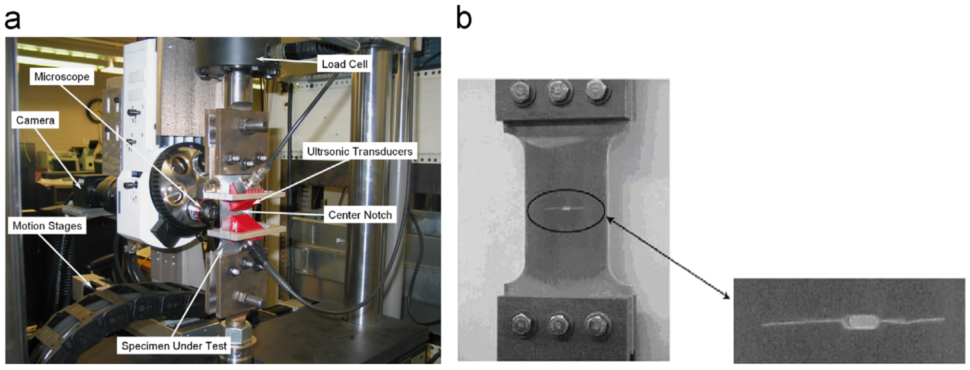

Fig. 7. Test apparatus and a test specimen in Example 4.3. (a) Computer-instrumented test apparatus. (b) Cracked 7075-T6 aluminum alloy specimen.

each time epoch, the ultrasonic data points have been collected for 50 load cycles (i.e., ~ 4 s) which produce a time series of 15,000 data points, i.e., 300 data points around each load peak. It has been observed that no significant changes occur in the fatigue crack behavior of the test specimen during the tenure of data acquisition at a given time epoch.

The set of time-series data at the nominal condition (i.e., time epoch τ_0) is first converted into a symbol sequence based on the maximum entropy partitioning (MEP) [13], where the alphabet size is obtained as $|\Sigma| = 6$ by following Eq. (3) with the selected parameters $\alpha = 0.5$ and $c = 0.16$; this partitioning is retained for all subsequent epochs, $\tau_1, \tau_2, \ldots$. It is noted that each partition segment is associated with a unique symbol in the alphabet and each symbol sequence characterizes the evolving fatigue damage.

Since the ground-truth PFSA are unknown in the experimental data sets, it is only possible to compare the predictive capabilities (e.g., entropy rates) of the PFSA after state splitting and state merging. With depth $D = 2$ (i.e., $|\Sigma|^D = 36$), the number of states of the PFSA model is reduced to 9 (i.e., $|Q| = 9$) by state merging. This PFSA of order 9 satisfies the convergence condition in Algorithm 4 with the threshold parameter $\eta_{mrg} = 0.010$.

Let $\widehat{\mathbf{P}}^0_{MAP}$ be the MAP state probability vector (see Eq. (12)) of the resulting probabilistic finite state automaton (PFSA) model at the epoch τ_0 and let $\widehat{\mathbf{P}}^k_{MAP}$ be the MAP state probability vector of the (possibly evolved) PFSA model at an epoch τ_k. Starting from the initial (zero-magnitude damage) at the epoch τ_0, the fatigue damage at an epoch τ_k is expressed in terms of $\widehat{\mathbf{P}}^0_{MAP}$ and $\widehat{\mathbf{P}}^k_{MAP}$ as a (scalar) damage divergence that is defined as

$$m_k \triangleq d(\widehat{\mathbf{P}}^k_{MAP}, \widehat{\mathbf{P}}^0_{MAP}) \tag{18}$$

where $d(\cdot,\cdot)$ is an appropriate metric. In this paper, $d(\cdot,\cdot)$ is chosen to be the standard Euclidean distance, while other choices can also be made (for example, see [12]).

The two-dimensional (optical microscope) images of the specimen surface, the corresponding profiles of ultrasonic sensor wave outputs, and the computed histograms of PFSA state vectors are respectively presented in the top, middle and bottom rows of Fig. 8, which display the gradual evolution of fatigue damage at time epochs, approximately at 1, 23, 34 and 47 kilocycles. The nominal condition at the time epoch τ_0 is chosen to be 1.0 kilocycles to ensure that the electro-hydraulic system of the test apparatus had come to a steady state and that no significant damage has occurred till that point, i.e., zero damage at the time epoch τ_0. The fatigue damage at subsequent time epochs, $\tau_1, \tau_2, \ldots \tau_k \cdots$, is then calculated with respect to the nominal condition at τ_0. The set consists of data collected at 56 consecutive epochs. The plates at the top of Fig. 8(a) and (b) show that no surface crack is yet visible under the optical microscope and the respective profiles of ultrasonic signals are largely similar. Nevertheless, over the span of 22 kilocycles (i.e., from the nominal condition of 1 kilocycles to 23 kilocycles) of load fluctuations, internal damage has already occurred in the specimen and is continually evolving, which is not detected by the optical microscope until ~ 34 kilocycles, but there is a clearly visible change in the respective histograms in the bottom rows of Fig. 8(a) and (b). The histogram of PFSA state probability in Fig. 8(b) at 23 kilocycles has been modestly distorted from the original histogram in Fig. 8(a) at the nominal condition of 1.0 kilocycles as a consequence of microstructural phenomena (e.g., possible reallocations of voids and inclusions) in the specimen. The cumulative effects of these irreversible changes are commonly called fatigue damage in the crack initiation phase in the fracture mechanics literature [42].

The plate at the top of Fig. 8(c) shows the appearance of a visible crack on the optical microscope image of the specimen surface in the vicinity of 34 kilocycles, while the respective profile of ultrasonic signal is clearly attenuated, accompanied by a significant change in the histogram of the PFSA state probability vector. This point is considered to be a transition of the crack initiation phase to the crack propagation phase in the fracture mechanics literature [42]; from this point onwards, the damage growth rate becomes significantly larger. Eventually, at ~ 47 kilocycles, the amplitude of the ultrasonic signal very rapidly decreases as seen in Fig. 8(d) when the surface crack is fully developed. It appears from Fig. 8 that the reduced-state PFSA (i.e., reduction from 36 states to 9 states) is adequate for representation of damage divergence (see Eq. (18)) in polycrystalline alloys. The effects of the PFSA model order reduction are explained below in more detail.

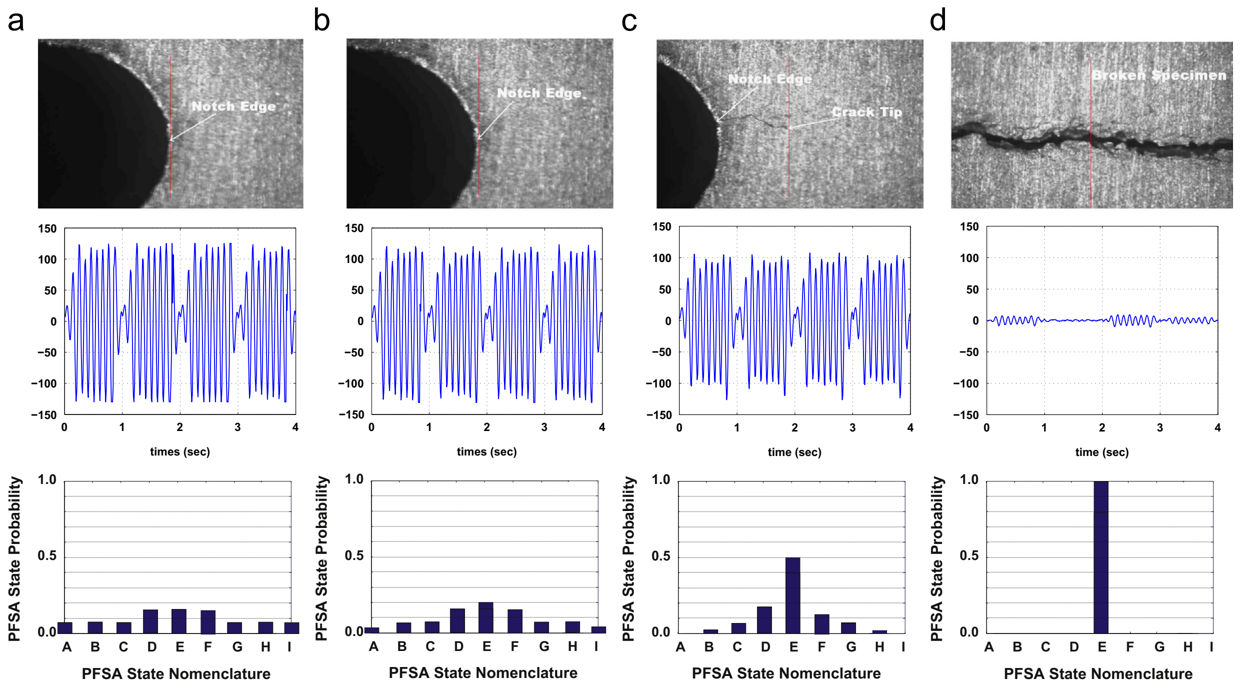

Fig. 8. Test results on a typical 7075-T6 aluminum alloy specimen investigated in Example 4.3: optical microscope images (top row); ultrasonic wave profiles under fatigue damage evolution (middle row); and histograms of PFSA state probability vector (bottom row). (a) Nominal: 1 kilocycles. (b) Internal damage: 23 kilocycles. (c) Surface crack: 34 kilocycles. (d) Full crack: 47 kilocycles.

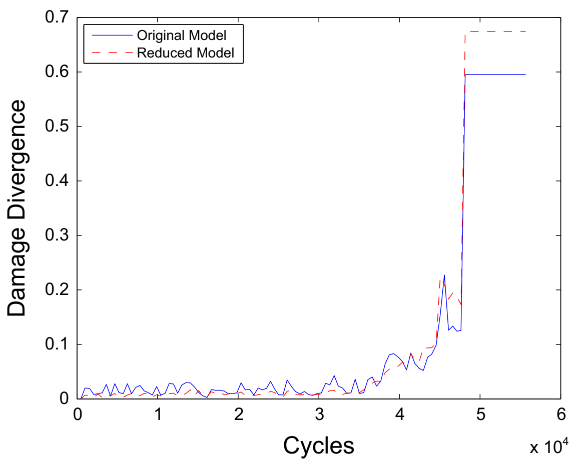

Fig. 9. Evolution of damage divergence in Example 4.3.

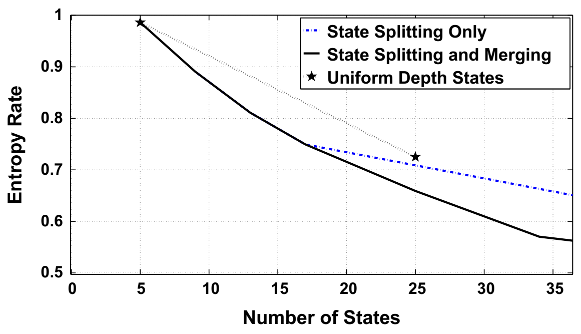

Fig. 10. Trade-off between entropy rate and # of states in Example 4.3.

Fig. 9 displays a comparison of a pair of damage evolution profiles over the entire period of 47 kilocycles. The plot in solid line corresponds to the original model of 36-state PFSA (i.e., $|\Sigma| = 6$ and $D=2$) and the plot in dashed line corresponds to the reduced-order model of 9-state PFSA as a consequence of state merging. These two profiles of damage divergence are observed to be very close to each other. It is apparent from Fig. 9 that the PFSA model order reduction by state merging not only does not compromise the quality of damage detection, but also the merging of the redundant states in the reduced-order 9-state model makes it more robust in the sense that small ripples are removed in the damage divergence profile, produced by the original 36-state model. A possible reason for the increase of robustness is better accuracy of the computed PFSA state probabilities (by frequency counting as seen in Eq. (12)) due to a significantly smaller number of PFSA states for the same data length of 15,000 data points.

Fig. 10 evaluates the efficacy of model order reduction by state merging in the PFSA-based *D*-Markov machine. Three Pareto optimal [43] curves, obtained as the convex hull of finitely many Pareto optimal points, are displayed in the space of entropy rate and the number of states. The top-most curve in Fig. 10 yields worst performance in terms of larger entropy rate, where each of the two points (marked by a $\star$) is obtained by uniform-depth *D*-Markov states with $D=1$ and $D=2$ i.e., for $|\Sigma| = 6$, the corresponding maximum state cardinalities of *D*-Markov machines are $|Q| = 6$ and $|Q| = 6^2 = 36$, respectively. The trade-off curve in the middle of Fig. 10 is obtained by using only the state splitting algorithm and yields superior performance relative to the uniform-depth *D*-Markov machine. The curve at the bottom in Fig. 10 is obtained by using a combination of state splitting and state merging. Although state merging may involve loss of information and thus increase the entropy rate of the PFSA, it yields a large decrease in the number of states (due to merging) that outweighs the shortcoming of modest increase in the entropy rate. Consequently, the Pareto optimal front for the state splitting and the state merging algorithm produces a significant improvement over the algorithm that implements state-splitting only.

5. Summary, conclusions, and future work

This paper presents the underlying concepts and algorithms for modeling dynamical systems as *probabilistic finite history automata* from symbol sequences as a special class of probabilistic finite state automata (PFSA). These PFSAs, called *D*-Markov machines [12], are structurally simple and computationally efficient to execute. While a larger depth *D* in the *D*-Markov machine is achieved by state splitting to provide a longer history resulting in a (possibly) better model prediction, it is accompanied by exponential growth of the number of PFSA states. The state merging algorithm focuses on order reduction in the *D*-Markov machine model. The underlying algorithms of state splitting and state merging in *D*-Markov machines are validated with three test examples. The first example presents a *D*-Markov machine model of a PFSA that is not a shift of finite type [3]. The second example develops *D*-Markov machines from the logistic map [28,39]. The third example presents the results of analysis of ultrasonic time series data, generated from laboratory experimentation, for detection of fatigue crack damage in a polycrystalline alloy.

The state merging algorithm, presented here, is suboptimal [12]. An important topic for future research is to investigate this issue and extend the general problem of approximation of a probability distribution by a PFSA, and more specifically by a *D*-Markov machine. In this regard, several key topics of future research are delineated as follows:

(1) Investigation of alternative methods of alphabet size selection by making use of tools of machine learning.
(2) Development of a rigorous framework for approximation of the probability distribution by making use of a general class of PFSA.
(3) Identification of a general relationship between the bound of modeling error and the number of states of the *D*-Markov machine.
(4) Investigation of other evaluation measures (e.g., perplexity [44]), used in speech recognition and natural language processing, for performance analysis of *D*-Markov machines.
(5) Investigation of numerical computation and robustness issues in the algorithms of *D*-Markov machines as needed for real-time in situ execution at the nodes of a sensor network.
(6) Extension of the proposed method to model symbolic sequences with limited data length by using Bayesian learning of transition probabilities. The Dirichlet distribution and the multinomial distribution are viable candidates for modeling the uncertainties resulting from (finite length) symbol strings [32].
(7) Elaborate validation of the proposed method using advanced experimental apparatuses of damage measurements (e.g., surface interferometer [41,45]).
(8) Investigation of a combined analytical and experimental procedure to demonstrate the usage of the proposed method for anomaly detection and pattern classification in diverse classes of physical processes.

Acknowledgments

The authors gratefully acknowledge the technical contributions of Mr. Patrick Adenis in this work.

References

[1] C. Daw, C. Fenney, E. Tracy, A review of symbolic analysis of experimental data, Rev. Sci. Instrum. 74 (February (2)) (2003) 915–930.
[2] H. Kantz, T. Schreiber, Nonlinear Time Series Analysis, 2nd ed. Cambridge University Press, Cambridge, UK, 2004.
[3] D. Lind, B. Marcus, Symbolic Dynamics and Coding, Cambridge University Press, Cambridge, UK, 1995.
[4] P. Dupont, F. Denis, Y. Esposito, Links between probabilistic automata and hidden Markov models: probability distributions, learning models and induction algorithms, Pattern Recognit. 38 (9) (2005) 1349–1371.
[5] G. Pola, P. Tabuada, Symbolic models for nonlinear control systems: alternating approximate bisimulations, SIAM J. Control Optim. 48 (2) (2009) 719–733.
[6] K. Deng, P. Mehta, S. Meyn, Optimal Kullback-Leibler aggregation via spectral theory of Markov chains, IEEE Trans. Autom. Control 71 (12) (2011) 2793–2808.
[7] E. Vidal, F. Thollard, C. de la Higuera, F. Casacuberta, R. Carrasco, Probabilistic finite-state machines—Part I and Part II, IEEE Trans. Pattern Anal. Mach. Intell. 27 (2005) 1013–1039.
[8] M. Vidyasagar, The complete realization problem for hidden Markov models: a survey and some new results, Math. Control Signals Syst. 23 (1–3) (2011) 1–65.
[9] P. Adenis, Y. Wen, A. Ray, An inner product space on irreducible and synchronizable probabilistic finite state automata, Math. Control Signals Syst. 23 (1) (2012) 281–310.
[10] P.F. Brown, P.V. deSouza, R.L. Mercer, V.J.D. Pietra, J.C. Lai, Class-based n-gram models of natural language, Comput. Linguist. 18 (4) (1992) 467–479.
[11] M. Mohri, F. Pereira, M. Riley, Weighted finite-state transducers in speech recognition, Comput. Speech Lang. 16 (1) (2002) 69–88.
[12] A. Ray, Symbolic dynamic analysis of complex systems for anomaly detection, Signal Process. 84 (7) (2004) 1115–1130.
[13] V. Rajagopalan, A. Ray, Symbolic time series analysis via wavelet-based partitioning, Signal Process. 86 (11) (2006) 3309–3320.
[14] S. Iyenger, P. Varshney, T. Damarla, A parametric copula-based framework for hypothesis testing using heterogeneous data, IEEE Trans. Signal Process. 59 (2011) 2308–2319.
[15] A. Sundaresan, P. Varshney, Location estimation of a random signal source based on correlated sensor observations, IEEE Trans. Signal Process. 59 (2011) 787–799.
[16] C. Rao, A. Ray, S. Sarkar, M. Yasar, Review and comparative evaluation of symbolic dynamic filtering for detection of anomaly patterns, Signal Image Video Process. 3 (2) (2009) 101–114.
[17] X. Jin, S. Sarkar, A. Ray, S. Gupta, T. Damarla, Target detection and classification using seismic and PIR sensors, IEEE Sens. J. 12 (2012) 1709–1718.
[18] S. Bahrampour, A. Ray, S. Sarkar, T. Damarla, N. Nasrabadi, Performance comparison of feature extraction algorithms for target

detection and classification, Pattern Recognit. Lett. 34 (16) (2013) 2126–2134.
[19] C. Bishop, Pattern Recognition, Springer, New York, NY, USA, 2006.
[20] T. Liao, Clustering of time series data—a survey, Pattern Recognit. 38 (2005) 1857–1874.
[21] C. Shalizi, K. Shalizi, Blind construction of optimal nonlinear recursive predictors for discrete sequences, in: AUAI '04: Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence, AUAI Press, Arlington, VA, USA, 2004, pp. 504–511.
[22] A. Paz, Introduction to Probabilistic Automata, Academic Press, New York, NY, USA, 1971.
[23] J. Hopcroft, R. Motwani, J. Ullman, Introduction to Automata Theory, Languages, and Computation, 2nd ed. ACM, New York, NY, USA, 2001, 45–138.
[24] M. Sipser, Introduction to the Theory of Computation, 3rd ed. Cengage Publishing, Boston, MA, USA, 2013.
[25] G. Mallapragada, A. Ray, X. Jin, Symbolic dynamic filtering and language measure for behavior identification of mobile robots, Trans. Syst. Man Cybern. Part B: Cybern. 42 (2012) 647–659.
[26] R. Carrasco, J. Oncina, Learning stochastic regular grammars by means of a state merging method, in: Grammatical Inference and Applications, Springer-Verlag, 1994, pp. 139–152.
[27] A. Clark, F. Thollard, PAC-learnability of probabilistic deterministic finite state automata, J. Mach. Learn. Res. 5 (2004) 473–497.
[28] C. Beck, F. Schlogl, Thermodynamics of Chaotic Systems: An Introduction, Cambridge University Press, Cambridge, U.K, 1993.
[29] P. Adenis, K. Mukherjee, A. Ray, State splitting and state merging in probabilistic finite state automata, in: American Control Conference, San Francisco, CA, USA, 2011, pp. 5145–5150.
[30] F. Thollard, P. Dupont, C. de la Higuera, Probabilistic DFA inference using Kullback-Leibler divergence and minimality, in: Seventeenth International Conference on Machine Learning, Morgan Kauffman, 2000, pp. 975–982.
[31] A. Berman, R. Plemmons, Nonnegative Matrices in the Mathematical Sciences, SIAM Publications, Philadelphia, PA, USA, 1994.
[32] Y. Wen, K. Mukherjee, A. Ray, Adaptive pattern classification for symbolic dynamic systems, Signal Process. 93 (2013) 252–260.
[33] T. Cover, J. Thomas, Elements of Information Theory, 2nd ed. Wiley, Hoboken, NJ, USA, 2006.
[34] L. Rabiner, A tutorial on hidden Markov models and selected applications in speech processing, Proc. IEEE 77 (2) (1989) 257–286.
[35] I. Chattopadhyay, Y. Wen, A. Ray, S. Phoha, Unsupervised inductive learning in symbolic sequences via recursive identification of self-similar semantics, in: Proceedings of American Control Conference, San Francisco, CA, USA, 2011, pp. 125–130.
[36] T. Niesler, P. Woodland, A variable-length category-based n-gram language model, in: IEEE Conference on Acoustics, Speech, and Signal Processing, vol. 1, May 1996, pp. 164–167.
[37] C. Kermorvant, P. Dupont, Improved smoothing for probabilistic suffix trees seen as variable order Markov chains, in: T. Elomaa, H. Mannila, H. Toivonen (Eds.), Machine Learning:ECML 2002, Lecture Notes in Computer Science, vol. 2430, Springer, Berlin, Heidelberg, 2002, pp. 185–194.
[38] P. Halmos, Naive Set Theory, Van Nostrand, Princeton, NJ, 1960, 26–29.
[39] K. Alligood, T. Sauer, J. Yorke, Complexity: Hierarchical Structures and Scaling in Physics, Springer-Verlag, York, NY, USA, 1996.
[40] S. Gupta, A. Ray, E. Keller, Symbolic time series analysis of ultrasonic data for early detection of fatigue damage, Mech. Syst. Signal Process. 21 (2) (2007) 866–884.
[41] D. Singh, S. Gupta, A. Ray, Symbolic dynamic analysis of surface deformation during fatigue crack initiation, Measur. Sci. Technol. 21 (3) (2010) 043003. (7 pp.).
[42] S. Suresh, Fatigue of Materials, 2nd ed. Cambridge University Press, Cambridge, UK, 1998.
[43] K. Miettinen, Nonlinear Multiobjective Optimization, Kluwer Academic, Norwell, MA, USA, 1998.
[44] M. Popel, D. Mareček, Perplexity of n-gram and dependency language models, in: P. Sojka, et al. (Eds.), TSD 2010, Lecture Notes in Artificial Intelligence, vol. 6231, Springer-Verlag, Berlin, Germany, 2010, pp. 173–180.
[45] D. Jha, D. Singh, S. Gupta, A. Ray, Fractal analysis of crack initiation in polycrystalline alloys using surface interferometry, Eur. Phys. Lett. 98 (4) (2012) 44006. (6 pp.).