

Segmenting multi-attribute sequences using Dynamic Bayesian Networks

Robert Gwadera*, Janne Toivola and Jaakko Hollmén,

Laboratory of Computer and Information Science, Helsinki University of Technology, Finland

Abstract

Discovering dependencies between attributes in multi-attribute event sequences (multi-sequences), also known as synchronized multi-stream sequences, is an important problem in many domains, including monitoring systems and molecular biology. Many real-life multi-sequences have a segmental structure, with segments of differing complexities of attribute dependencies, which reflects a changing nature of the dependencies over time and space. In this paper we propose a new approach for discovering dependencies in multi-sequences which considers a possible segmental nature of such dependencies and tries to describe the multi-sequences in probabilistic terms using Dynamic Bayesian Networks (DBN). To accurately quantify such changing dependencies, we segment the multi-sequence by fitting an optimal DBN for each segment. We use the Bayesian Information Criterion (BIC) to select an optimal DBN structure and the number of segments of the multi-sequence.

1. Introduction

Discovering dependencies in multi-attribute event sequences (multi-sequences), also known as synchronized multi-stream sequences, is an important problem in many domains, including monitoring systems and molecular biology. Usually such streams are generated by monitoring systems (sensor networks) where an event corresponds to a configuration of values obtained from different sensors. As an example, consider an intensive care unit in a hospital, where a patient is continuously monitored for values of attributes like blood pressure, temperature, etc. Another class of monitoring applications includes spatio-temporal data, where the attributes correspond to sensors located in geographical locations. In molecular biology, multi-attribute sequences include gene expression levels measured for many genes simultaneously [11]. The assumption that the streams are synchronized does not restrict practical applications since streams can be synchronized by introducing a new symbol [12] or by interpolation [15].

Different approaches have been considered for discovering dependencies in multi-sequences including dependency in terms of significant occurrences of complex inter and intra-stream patterns [12], dependency in terms of correlation between entire streams [15], dependency in terms of probabilistic relationships in Dynamic Bayesian Networks (DBN) [9, 11, 4] and dependency in terms of Boolean Networks [14]. In particular, DBNs have been used for inferring gene regulatory networks from gene expression time series. However all the presented approaches assumed a single model of dependence over the entire multi-sequence implying that the multi-sequence is time homogeneous in terms of such dependencies.

Many real-life multi-sequences have a segmental structure, with segments of differing complexities of attribute dependencies, which reflects a changing nature of the dependencies over time and space, see, e.g., [7].

In this paper we propose a new approach for discovering dependencies in multi-sequences, which instead of assuming a single dependency model for the entire multi-sequence, considers a possible segmental nature of such dependencies and tries to describe the multi-sequence in terms of DBNs. Thus, we discover dependencies between attributes in terms of probabilistic relationships in DBNs, where the attributes correspond to vertices (random variables) and directed edges correspond to probabilistic dependencies between the attributes. To accurately quantify such changing dependencies, we segment the multi-sequence into non-overlapping and homogeneous segments by fitting an optimal DBN for each segment, where an optimal DBN structure is a trade-off between maximizing the model fit in terms of the likelihood of observed data and the model complexity in terms of its graph structure. We use the Bayesian Information Criterion (BIC) to select an optimal DBN structure and the number of segments of the multi-sequence. Thus, for every segment we obtain a set of inter-stream and intra-stream dependencies.

In [5] a single-sequence segmentation algorithm that uses tree models was presented. This paper builds on [5] by presenting a novel algorithm for segmenting multi-sequences. The most similar approach to our approach for analyzing multi-sequences seems to be *bi-clustering*

*Contact author. E-mail: gwadera@cis.hut.fi

[10], that computes possibly overlapping associations (*bi-clusters*) between subsets of rows and subsets of columns of a given data matrix. Thus, in *bi-clustering* terms a multi-sequence can be treated as a matrix, where the rows correspond to the attributes and the columns correspond to the evolution of the values of the attributes. However, there are the following main differences between our method and *bi-clustering*: (1) our method considers only contiguous columns while *bi-clustering* considers subsets of non-contiguous columns; and (2) by fitting DBNs to segments our method finds dependencies between elements corresponding to different columns while *bi-clustering* finds dependencies between elements from the same columns.

It may appear that in order to segment a multi-sequence one could merge the streams and fit Markov Chains (MC) to the merged single-sequence instead of fitting DBNs to the multi-sequence. However there are many drawbacks of such an approach. For example, the combined alphabet is exponential in the number of streams.

The challenges in our approach are the following: (i) fitting an optimal DBN to data is NP-hard [2]; (ii) the standard optimal segmentation algorithm that uses dynamic programming has a quadratic time complexity; and (iii) many real sources may have short segments and the algorithm has to reliably fit DBNs from sparse data. We address (i) and (ii) by proposing a graph structure model space that is a reasonable trade-off between complexity and size. We address (iii) by proposing a limit on the minimum segment size.

The paper is organized as follows: Section 2 reviews the foundations on Bayesian Networks, Section 3 states the problem of multi-attribute sequence segmentation using Dynamic Bayesian Networks, Section 4 presents the details of the algorithm, Section 5 presents our experimental results and Section 6 presents conclusions.

2. Dynamic Bayesian Networks

In this section we review the foundations on Dynamic Bayesian Networks.

2.1. Notation

- $\mathcal{A} = \{a_1, a_2, \dots, a_{|\mathcal{A}|}\}$ is an alphabet.
- $\mathcal{X} = \{X_1, X_2, \dots, X_I\}$ is a set of random variables where each X_i takes values from \mathcal{A} . We use x_i or $X_i = x_i$ to denote X_i has value $x_i \in \mathcal{A}$. We use \mathbf{x} or $\mathcal{X} = \mathbf{x}$ to denote \mathcal{X} is in configuration \mathbf{x} . $X_t^{(i)}$ denotes a random variable corresponding to attribute i (stream i) and time instance t . In general we use capital letters to denote random variables and lower case letters to denote their instantiations.

- $\mathcal{S} = \{s^{(1)}, s^{(2)}, \dots, s^{(I)}\}$ is a multi-attribute sequence (multi-sequence) as a set of input streams each of length n , where $I = |\mathcal{S}|$. $s^{(i)} = [s_1^{(i)}, s_2^{(i)}, \dots, s_n^{(i)}]$ is the i -th attribute sequence (i -th stream), where $s_t^{(i)} \in \mathcal{A}$. We use $\mathcal{S}_{a:b}$ and $s_{a:b}^{(i)}$ to denote a contiguous subsequence between a and b from \mathcal{S} and $s^{(i)}$ respectively. We also use $S_t^{(i)}$ to refer to the random variable corresponding to stream i and time t .
- D is the maximum in-degree of a node in a DBN and d_i is the in-degree of node X_i .
- $\mathbf{Pa}(X_i)$ denotes the *sorted set* of parents of X_i , where the parents are sorted in increasing order with respect to their subscripts. $\mathbf{pa}(X_i)$ denotes a configuration of the variables in $\mathbf{Pa}(X_i)$. We also use $\mathbf{Pa}(X_t^{(i)})$ and $\mathbf{pa}(X_t^{(i)})$ in reference to $X_t^{(i)}$.
- c_i is the number of configurations of $\mathbf{Pa}(X_i)$. Since all variables take values from \mathcal{A} , $c_i = |\mathcal{A}|^{d_i}$.
- G : Bayesian network structure as a *directed acyclic graph* (DAG). $\Theta = \{\theta_1, \theta_2, \dots, \theta_m\}$ is the set of all parameters. $\theta_i = [\theta_{i,1}, \theta_{i,2}, \dots, \theta_{i,c_i}]$ is a parameter vector associated with X_i , where $\theta_{i,j} = [\theta_{i,j,1}, \theta_{i,j,2}, \dots, \theta_{i,j,|\mathcal{A}|}]$ and $\theta_{i,j,k} = P(X_i = k | \mathbf{Pa}(X_i) = j)$. $N_n(i, j, k)$ is the number of occurrences of $X_i = k$, $\mathbf{Pa}(X_i) = j$ in a multi-sequence of length n and $N_n(i, j) = \sum_{k=1}^{|\mathcal{A}|} N_n(i, j, k)$. To denote *maximum likelihood* (ML) estimators of parameters we use $\hat{\Theta}$ and $\hat{\theta}_{i,j,k}$ respectively.
- Given a graph $G(E, V)$ and the parameter set Θ we use $DBN(G, \Theta)$ to denote the corresponding DBN. We use k -DBN to denote a DBN of order k meaning the maximum length of the context of the model. We also use $G.E = [\mathbf{Pa}(X_t^{(1)}), \mathbf{Pa}(X_t^{(2)}), \dots, \mathbf{Pa}(X_t^{(I)})]$ to denote a vector of parent sets of all child nodes, where \emptyset means an empty set of parents.

2.2. Definition of Dynamic Bayesian Networks

A Bayesian Network (BN) for a set of I discrete random variables $\mathcal{X} = \{X_1, X_2, \dots, X_I\}$ consists of two components: (1) a DAG in which the nodes are in a one-to-one correspondence with variables in \mathcal{X} and each node X_i receives directed edges from its set of parent nodes $\mathbf{Pa}(X_i)$; (2) a set of probability distributions $\Theta = \{\theta_1, \theta_2, \dots, \theta_I\}$ associated with each variable X_i , where each θ_i is represented by a conditional probability table (CPT) [8].

Each node X_i in a BN is conditionally independent of its non-descendants given its parents and the joint probability

for a given configuration $\mathcal{X} = \mathbf{x}$ is equal to:

$$P(\mathcal{X} = \mathbf{x}|G, \Theta) = \prod_i^I P(x_i|\mathbf{pa}(X_i), \theta_i). \quad (1)$$

In general, conditional independence between arbitrary nodes in a BN can be determined by the *d-separation* criterion, where 'd' means 'directed' [13].

From (1) it is clear that the likelihood in a BN is decomposable into independent contributions of individual variables and their parents. This property is of fundamental importance in learning the graph of BNs using model selection criteria introduced in Section 2.4 since (2) can be maximized independently for each variable. This makes the criteria *decomposable* and results in a more efficient optimal model search method [2].

Dynamic Bayesian Networks can be defined as a special case of BN that are dedicated to sequential data. The graph structure and parameters that form the static interpretation of a system is identical to BN. However the variables in a DBN denote a state of a system that satisfies the Markov property: the state of the system at time t depends only on a limited number of contiguous past states. The length of the contiguous past states is called the order of the Markov property. Also, a directed edge from a node X_i to a node X_j can be regarded as causal relationship meaning X_i causes X_j .

Given a $DBN(G, \Theta)$ and a multi-sequence $\mathcal{S}_{1:n}$ of length n the likelihood of the multi-sequence is equal to:

$$L(\mathcal{S}_{1:n}|G, \Theta) = \prod_{t=1}^n \prod_{i=1}^I P(s_t^{(i)}|\mathbf{pa}(S_t^{(i)})). \quad (2)$$

2.3. Parameter learning

In the segmentation algorithm we first select model structure and then estimate conditional probabilities of the nodes from a segment of the input multi-sequence. This corresponds to the case of known structure and full observability. Thus, given a graph G and an input multi-sequence $\mathcal{S}_{1:n}$ we use the *maximum likelihood(ML)* method to compute conditional probabilities for each node as

$$\hat{\theta}_{i,j,k} = P(S_t^{(i)} = k|\mathbf{Pa}(S_t^{(i)}) = j) = \frac{N_n(i, j, k)}{N_n(i, j)}. \quad (3)$$

Using the decomposability of (1) we obtain the following formula for the log ML of $\mathcal{S}_{1:n}$:

$$\begin{aligned} \log_2(L(\mathcal{S}_{1:n}|G, \hat{\Theta})) &= \sum_{i=1}^I \sum_{\mathbf{pa}(S_t^{(i)}), s_t^{(i)}} N_n(\mathbf{pa}(S_t^{(i)}), s_t^{(i)}) \cdot \\ &\log_2 \left(\frac{N_n(\mathbf{pa}(S_t^{(i)}), s_t^{(i)})}{N_n(\mathbf{pa}(S_t^{(i)}))} \right) \end{aligned} \quad (4)$$

where $N_n(\mathbf{pa}(S_t^{(i)}), s_t^{(i)})$ is the number of occurrences of symbol $s_t^{(i)}$ with the context $\mathbf{pa}(S_t^{(i)})$ in $\mathcal{S}_{1:n}$.

2.4. Structure learning

The problem of learning a BN structure can be defined as follows. Given a set of nodes $\mathcal{X} = \{X_1, X_2, \dots, X_I\}$ and an I -attribute input sequence $\mathcal{S}_{1:n}$, find a graph G that *best fits* the input multi-sequence. The approach to this problem is to use a scoring function, that evaluates the quality of the fit for a given graph and then to search for an optimal graph over a space of graphs.

In the following sections we first review known criteria (scoring functions) for selecting an optimal DBN graph, where an optimal DBN structure is a trade-off between maximizing the model fit in terms of the likelihood of observed data and the model complexity in terms of its graph structure. Then we review the search methods.

Bayesian Information criterion (BIC)

For a BN graph G BIC has the following formula [3]

$$\begin{aligned} BIC_G(\mathcal{S}_{1:n}) &= -\log_2(L(\mathcal{S}_{1:n}|G, \hat{\Theta})) \\ &+ \sum_{i=1}^I \frac{|\mathcal{A}|^{d_i} (|\mathcal{A}| - 1)}{2} \log_2(n). \end{aligned} \quad (5)$$

Thus, an optimal graph with respect to BIC, is defined as follows:

$$\hat{G}_{BIC}(\mathcal{S}_{1:n}) = \min_{G \in \mathcal{G}} (BIC_G(\mathcal{S}_{1:n})). \quad (6)$$

In statistical terms, the first term in (5) measures the goodness of fit of G to $\mathcal{S}_{1:n}$, and the second term is the *penalty term* equal to the number of free parameters, which prevents BIC from overfitting.

2.4.1 Search Methods

The problem of finding a graph that optimizes the BIC score is NP-hard because of the exponential search space [2]. Therefore the following two main approaches are used in practice: (1) bounding the in-degree by a small value of D ; and (2) using heuristic local search algorithms that include greedy search, greedy search with restart and simulated annealing [6]. The drawback of the heuristic methods is that they may fail to find a global optimum.

3. Multi-attribute sequence segmentation using Dynamic Bayesian Networks

Definition of the problem of multi-attribute sequence segmentation using Bayesian Networks can be stated as follows. Given:

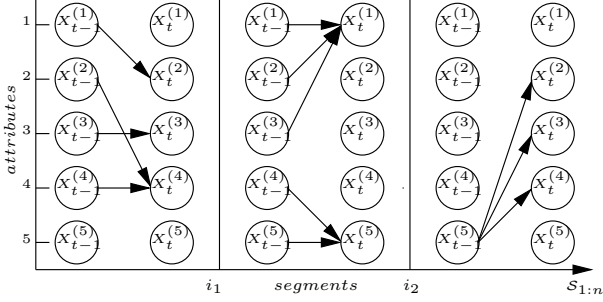


Figure 1. Example segmentation of a 5-attribute sequence using Dynamic Bayesian Networks. The figure presents three segments and the corresponding probabilistic models for each segment.

- $S_{1:n}$: a finite multi-sequence.
- K : the maximal number of segments.
- $\mathcal{G} = \{G_1, G_2, \dots, G_{|\mathcal{G}|}\}$: a set of DBN graph structures (structure space).
- $cost(\mathcal{S}_{a:b})$: a cost function for segment $\mathcal{S}_{a:b}$.

find a vector of partition points $\mathbf{i} = [i_1, i_2, \dots, i_k]$, $1 \leq k \leq K - 1$ such that

$$\mathbf{i} = \arg \min_{[i_1, i_2, \dots, i_k]} \left\{ \sum_{j=0}^k cost(\mathcal{S}_{i_j:i_{j+1}-1}) + k \cdot I \cdot B \right\},$$

where $1 \leq k \leq K - 1$, $i_0 = 1$, $i_{k+1} = n$ and B is a border insertion penalty and the cost function in terms of BIC is computed as $cost(\mathcal{S}_{a:b}) = BIC_{\hat{G}_{BIC}}(\mathcal{S}_{a:b})$.

Figure 1 shows an example result of 5-attribute sequence segmentation using DBNs. The figure presents three segments and the corresponding probabilistic models for each segment.

4. Algorithm for Multi-attribute sequence segmentation using DBNs

In this section we present details of our algorithm for multi-sequence segmentation using DBNs.

4.1. Proposed structure space and search method

For the sake of efficiency we define the following structure space \mathcal{G} for possible DBNs: (1) the in-degree of each node is bounded by $D = 3$; (2) there are only inter-slice connections, i.e. only connections between time $t - 1$ and

t ; and (3) the memory of the DBN is one time lag. Thus the set of all parents in \mathcal{G} consists of variables $X_{t-1}^{(i)}$ and the set of all child nodes consists of variables $X_t^{(i)}$. In order to obtain an optimal solution we use the exhaustive search over the space of models.

We chose $D = 3$ as a trade-off between model complexity and algorithm efficiency. In general, in order to cut the search space significantly one should choose $D \ll I$, which easily follows from the following fact. Let $\Pi = \{\pi_1, \pi_2, \dots, \pi_{|\Pi|}\}$ be the set of all possible subsets of parents of cardinality up to D for a child node in \mathcal{G} . If $I = D$ then clearly $|\Pi| = \sum_{i=0}^I \binom{I}{i} = 2^I$, while for $D \leq 0.5 \cdot I$, which is the goal, $|\Pi| = \sum_{i=0}^D \binom{I}{i} \leq 2^I \frac{D}{I}$.

As a further restriction of the structure space, we could restrict the set of possible parents for each node $X_t^{(i)}$ to a particular subset of all parents.

4.2. Optimal segmentation algorithm

The standard optimal segmentation algorithm can be expressed by the following dynamic programming equation, due to Bellman [1]:

$$C[k, b] = \min_{k-1 \leq a \leq b} \{C[k-1, a-1] + W[a, b]\}.$$

In the above equation, $C[k, b]$ is the optimal k -segmentation cost of the prefix $S_{1:b}$ and $W[a, b]$ is the cost function (score) of the segment $\mathcal{S}_{a:b}$, that is the BIC (6) score as defined in Section 2.4.

Since computing $W[a, b]$ in time proportional to the length of the segment $\mathcal{S}_{a:b}$ would lead to overall $\mathcal{O}(n^3)$ running time, we adapt an optimization technique that was proposed in [5]. The idea of the speedup is to keep appropriate counts for computing the scores for a fixed ending position b and proceeding backward for $a = b, b-1, \dots, 0$. Thus, given the counts for a segment $\mathcal{S}_{a:b}$, they can be updated in constant time for the next starting position $a-1$ since only one new configuration has to be added. The technique achieves $\mathcal{O}(n^2)$ by computing $W[a, b]$ in constant time.

Given the proposed structure space \mathcal{G} , we compute $W[a, b]$ in one pass as follows. For every $\pi_m \in \Pi$ and for each node $X_t^{(i)}$, we keep counts of the number of occurrences of $X_t^{(i)} = k$ given $\pi_m(X_t^{(i)}) = j$ for the current segment $\mathcal{S}_{a:b}$ that we call $N_n(m, i, j, k)$. Then given the counts we compute the BIC (6) score in a constant time for a fixed I .

Since the computation of $W[a, b]$ can be done in a constant time for all pairs (a, b) , the algorithm runs in time $\mathcal{O}(n^2)$ times the complexity of computing $W[a, b]$. The space complexity of the algorithm is equal to $\mathcal{O}(Kn)$ plus the space complexity for computing $W[a, b]$. Thus, the overall time complexity of algorithm is $\mathcal{O}(n^2 \cdot I \cdot 2^I \frac{D}{I} \cdot |\mathcal{A}|^{\frac{D}{2}} \cdot |\mathcal{A}|)$ and the space complexity is $\mathcal{O}(Kn + 2^I \frac{D}{I} \cdot |\mathcal{A}|^{\frac{D}{2}} \cdot |\mathcal{A}|)$.

4.3. Minimum segment size

Given the proposed structure space of the algorithm, a fundamental question is what the minimum segment size should be to reliably learn a structure. We view this problem in terms of guaranteeing that the CPTs can be estimated reliably. Therefore, since members of the proposed structure space are 1-DBNs with the in-degree bounded by D we use $\Omega(|\mathcal{A}|^{D+1})$ as a bound, which follows from the fact that if we assume that an occurrence of every configuration in a given position in the multi-sequence is equally likely with probability $\frac{1}{|\mathcal{A}|^{D+1}}$ then the length of the multi-sequence has to be at least $|\mathcal{A}|^{D+1}$ to guarantee that on-average every configuration occurs at least once.

4.4. Border insertion penalty

The border insertion penalty can be understood in terms of the *Hidden Markov Model* (HMM) as a transition probability between hidden states of the generating source, where segments correspond to the hidden states [5]. In the case of a multi-sequence, a partition point corresponds to a change of state of every individual stream. Thus, using the single-sequence penalty from [5], we obtain the following multi-sequence penalty for the BIC method: $B_{BIC} = (k - 1) \cdot I \cdot \frac{\log_2(n)}{2}$, where k is the optimal number of segments that was found by the algorithm and I is the number of streams.

5. Experiments

In this section we generated a 10-attribute sequence $\mathcal{S}_{1:300}$ and then segmented it using our algorithm. The sequence was generated by concatenating three segments: $\mathcal{S}_{1:100}$, $\mathcal{S}_{101:200}$ and $\mathcal{S}_{201:300}$ that were generated individually from the following models: $DBN_1(G_1, \Theta_1)$, $DBN_2(G_2, \Theta_2)$, $DBN_3(G_3, \Theta_3)$, where the structures of individual graphs and parameters are presented in Table 1. Thus, we used the same set of parameters Θ for all DBNs, i.e., $\Theta_1, \Theta_2, \Theta_3 \in \Theta$.

Figure 2 shows the generated 10-attribute sequence $\mathcal{S}_{1:300}$, where the vertical lines correspond to the boundaries between the generated segments $\mathcal{S}_{1:100}$, $\mathcal{S}_{101:200}$ and $\mathcal{S}_{201:300}$.

Figure 3 presents the result of the segmentation of \mathcal{S} using our algorithm and includes both the segment boundaries and the optimal DBN graphs. For the sake of clarity of the presentation we omit the nodes and only draw the edges. However, the endpoints of the directed edges of the corresponding DBN graphs indicate the node numbers. Thus, the x -axis denotes a partition point location and the y -axis denotes the stream (attribute) number. The vertical lines correspond to segment boundaries and the directed edges

Table 1. Structures and parameters used for generating 10-attribute sequence $\mathcal{S}_{1:300}$ in Figure 2.

$G_1.E = \{\{5\}, \emptyset, \emptyset, \emptyset, \{5\}, \emptyset, \emptyset, \emptyset, \emptyset, \{5\}\}$
$G_2.E = [\emptyset, \emptyset, \{4\}, \{4\}, \emptyset, \emptyset, \{8\}, \{8\}, \emptyset, \emptyset]$
$G_3.E = [\emptyset, \{2\}, \emptyset, \emptyset, \{2, 9\}, \emptyset, \emptyset, \emptyset, \{9\}, \emptyset]$
$P(X_t^{(i)} = 1) = 0.5$
$P(X_t^{(i)} = 0 \text{Pa}(X_t^{(i)}) = 0) = 0.9$
$P(X_t^{(i)} = 1 \text{Pa}(X_t^{(i)}) = 1) = 0.9$
$P(X_t^{(i)} = 0 \text{Pa}(X_t^{(i)}) = [0, 0]) = 0.9$
$P(X_t^{(i)} = 1 \text{Pa}(X_t^{(i)}) = [1, 1]) = 0.9$
$P(X_t^{(i)} = 0 \text{Pa}(X_t^{(i)}) = [0, 1]) = 0.5$
$P(X_t^{(i)} = 1 \text{Pa}(X_t^{(i)}) = [1, 0]) = 0.5$

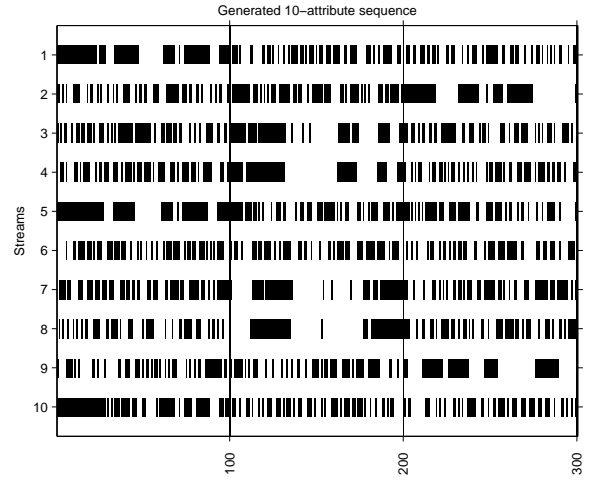


Figure 2. Generated 10-attribute sequence $\mathcal{S}_{1:300}$ according to the probabilistic model in Table 1. Vertical lines denote boundaries between the generated segments $\mathcal{S}_{1:100}$, $\mathcal{S}_{101:200}$ and $\mathcal{S}_{201:300}$.

from a y -coordinate i_1 to another y -coordinate i_2 within a given segment denote an edge from $X_{t-1}^{(i_1)}$ to $X_t^{(i_2)}$ in the optimal DBN that describes the respective segment, where $0 \leq i_1, i_2 \leq I$.

By comparing Figure 2 with Figure 3 given the structures and parameters (Table 1) of the generating models we can state two findings: (1) the generating models indeed generated the multi-sequence; and (2) the algorithm discovered the correct segment boundaries and the original generating models within corresponding segments.

The parameters in Table 1 were constructed in such way that a parent node generates contiguous runs of ones and ze-

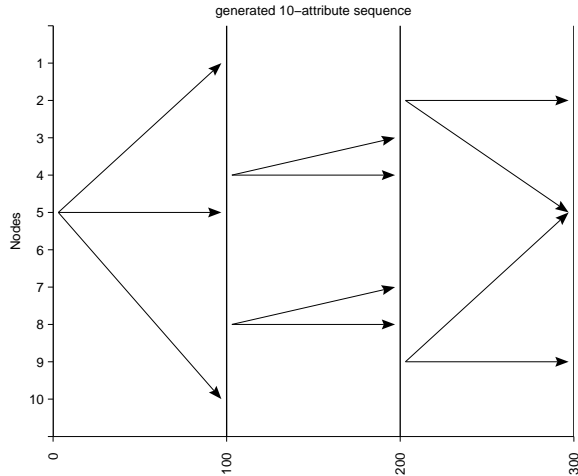


Figure 3. Results of segmentation of the generated 10-attribute sequence $\mathcal{S}_{1:300}$ presented in Figure 2. Vertical lines denote segment boundaries and the directed edges denote DBN edges.

ros in its child stream with probability $P(1|1) = P(0|0) = 0.9$ depending whether its current state is 0 or 1. Now we consider each generating model case. In $DBN_1(G_1, \Theta_1)$ $X_{t-1}^{(5)}$ causes $X_t^{(1)}$ and $X_t^{(10)}$ to output the contiguous runs of ones and zeros. In $DBN_2(G_2, \Theta_2)$ there are two independent pairs of streams $X_t^{(3)}$ with $X_t^{(4)}$ and $X_t^{(7)}$ with $X_t^{(8)}$, where in the first pair $X_t^{(4)}$ causes $X_t^{(3)}$ and in the second pair $X_t^{(8)}$ causes $X_t^{(7)}$ to output the runs of ones and zeros. In $DBN_3(G_3, \Theta_3)$ there are two independent parents $X_{t-1}^{(2)}$ and $X_{t-1}^{(9)}$ that cause $X_t^{(5)}$ to generate the runs of ones or zeros whenever $X_{t-1}^{(2)}$ and $X_{t-1}^{(9)}$ have the same values equal 1 or 0 respectively. In other words, if sequences $X_{t-1}^{(2)}$ and $X_{t-1}^{(9)}$ are in phase then they generate contiguous runs of zeros or ones in $X_t^{(5)}$. Thus, $DBN_3(G_3, \Theta_3)$ works as a “noisy XNOR gate” with respect to $X_{t-1}^{(2)}$ and $X_{t-1}^{(9)}$ as inputs and $X_t^{(5)}$ as the output. The term *noisy* corresponds to the fact that the gate produces an output with a given probability depending on input values instead of a deterministic operation.

6. Conclusions

We presented a novel approach to discovering dependencies in multi-attribute sequences that considers a possible segmental character of such dependencies and tries to describe the multi-sequence in terms of DBNs.

References

- [1] R. Bellman. On the approximation of curves by line segments using dynamic programming. *Communications of the ACM*, 4(6):284, 1961.
- [2] D. Chickering. Learning Bayesian networks is NP-Complete. In D. Fisher and H. Lenz, editors, *Learning from Data: Artificial Intelligence and Statistics V*, pages 121–130. Springer-Verlag, 1996.
- [3] I. Csiszar and Z. Talata. Context tree estimation for not necessarily finite memory processes, via bic and mdl. *IEEE Transactions on Information Theory*, 52(3):1007–1016, 2006.
- [4] N. Friedman, I. Nachman, and D. Peér. Learning bayesian network structure from massive datasets: The “sparse candidate” algorithm. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI99)*, pages 206–215, 1999.
- [5] R. Gwadera, A. Gionis, and H. Mannila. Optimal segmentation using tree models. In *Sixth IEEE International Conference on Data Mining*, pages 244–253, 2006.
- [6] D. Heckerman, D. Geiger, and D. Chickering. Learning bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, (20):197–243, 1995.
- [7] J. Himberg, K. Korpiaho, H. Mannila, J. Tikanmäki, and H. Toivonen. Time series segmentation for context recognition in mobile devices. In *First IEEE International Conference on Data Mining*, pages 203–210, October 2001.
- [8] M. Jordan. *Learning in Graphical Models*. The MIT Press, 1998.
- [9] S. Liang, S. Fuhrman, and R. Somogyi. Reveal: a general reverse engineering algorithm for inference of genetic network architectures. In *Pacific Symposium on Biocomputing*, pages 18–19, 1998.
- [10] S. Madeira and A. Oliveira. Biclustering algorithms for biological data analysis: a survey. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*.
- [11] K. Murphy and S. Mian. Modeling gene expression data using dynamic bayesian networks. Technical report, Computer Science Division, University of California, Berkeley, CA, 1999.
- [12] T. Oates and P. Cohen. Searching for structure in multiple streams of data. In *13th International Conference on Machine Learning*, pages 346–354, 1996.
- [13] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988.
- [14] S. K. T. Akutsu, S. Miyano. Identification of genetic networks from a small number of gene expression patterns under the boolean network model. In *Proc. Pacific Symposium on Biocomputing*, pages 17–18, 1999.
- [15] Y. Zhu and D. Shasha. Statstream: Statistical monitoring of thousands of data streams in real time. In *28th VLDB Conference*, 2002.