Knowl Inf Syst (2008) 15:259–283
DOI 10.1007/s10115-007-0091-5

REGULAR PAPER

Optimal segmentation using tree models

Robert Gwadera · Aristides Gionis · Heikki Mannila

Received: 28 March 2007 / Accepted: 28 April 2007 / Published online: 28 July 2007

Abstract Sequence data are abundant in application areas such as computational biology, environmental sciences, and telecommunications. Many real-life sequences have a strong segmental structure, with segments of different complexities. In this paper we study the description of sequence segments using variable length Markov chains (VLMCs), also known as tree models. We discover the segment boundaries of a sequence and at the same time we compute a VLMC for each segment. We use the Bayesian information criterion (BIC) and a variant of the minimum description length (MDL) principle that uses the Krichevsky-Trofimov (KT) code length to select the number of segments of a sequence. On DNA data the method selects segments that closely correspond to the annotated regions of the genes.

Keywords Sequence segmentation · MDL · DNA segmentation · Sequence data mining

1 Introduction

We consider the problem of *segmenting* a sequence of symbols into contiguous homogeneous segments. The segmentation problem has many applications in areas such as computational biology, environmental sciences, and context recognition in mobile devices, see, e.g., [4,13,12]. The problem has been widely studied in different fields, for instance, in statistics it is known under the name *change-point detection*.

Many segmentation algorithms have been proposed in the data mining community, ranging from on-line to offline, from heuristic to optimal (typically involving a dynamic programming approach), and from combinatorial to probabilistic, see, e.g., [14,12,23].

We consider the sequence segmentation problem as a model selection process where we fit a *variable-length Markov chain* (VLMC) [5] to segments of the input sequence. We use the Bayesian information criteria (BIC) and a variant of the minimum description length (MDL)

R. Gwadera (✉) · A. Gionis · H. Mannila
HIIT, Basic Research Unit, Helsinki University of Technology and University of Helsinki, Helsinki, Finland
e-mail: gwadera@cis.hut.fi

principle that uses the Krichevsky-Trofimov (KT) code length [7] for: (i) fitting an optimal VLMC for each segment; and (ii) determining the optimal number of segments to partition the sequence.

A d-order VLMC is a Markov chain (MC) whose contexts (memory) are allowed to be of variable length. Such reduced models are also called *tree models*, since they can be conveniently represented by a context tree. The tree can range from a full tree in the case of an ordinary d-order full MC to an empty tree in the case of a 0-MC. The variable length memory of VLMCs has the potential of capturing complex phenomena that are present in real-life sequences. VLMCs provide a sparse representation of a sequence by reducing the number of parameters to be estimated. This flexibility of VLMCs is useful for the segmentation task: we can fit high order models to segments to maximize the likelihood, without being penalized for an exponential increase in the number of parameters (as in the case of ordinary MCs).

The fundamental question is whether the increased modeling power of VLMCs with respect to MCs really translates into a better segmentation performance. As it turns out VLMCs can provide more accurate segmentations than MCs and are also capable of recognizing partition points in cases where MCs fail.

We fit an optimal VLMC for each segment of the sequence in order to discover segments of differing contextual regularities corresponding to different tree structures, where an optimal tree is a trade-off between maximizing the maximum likelihood of the segment and the tree complexity.

The challenges in our approach are the following: (i) fitting an optimal VLMC to data is a non-trivial task because it involves selecting among an exponential number of trees; (ii) many real sources have short segments and the algorithm has to fit VLMCs from sparse data; and (iii) the standard dynamic programming algorithm has a quadratic time complexity.

We solve task (i) by adapting known pruning algorithms of VLMC to be used with the BIC and KT criteria. We address (ii) by using VLMCs of variable order (maximum depth of the tree) with respect to segment length such that we fit taller trees to longer segments and shorter trees to shorter segments. Finally, we solve (iii) by applying numerous optimization techniques.

We conducted experiments on synthetic data, on a variety of DNA sequences and on natural-language text sources. The results show that the method selects gene segments that closely correspond to the currently known (annotated) gene regions. Our segmentation system, called *TreeSegment*, and the data sources used in experiments are available at the authors' web pages.

The rest of this paper is organized as follows. In Sect. 2 we introduce the notion of tree models. Section 3 presents the details of the algorithm. In Sect. 4 we present our experimental results. Section 5 reviews related work and Sect. 6 is a short conclusion.

2 Tree models

2.1 Basic definitions

Let $\mathcal{A} = \{a_1, a_2, \ldots, a_m\}$ be an alphabet of cardinality $m = |\mathcal{A}|$. Let $s_1^n = s_1 s_2 \ldots s_n$ be a string over $\mathcal{A}$ and let $s_i^j = s_i s_{i+1} \ldots s_j$. A concatenation of strings u and v is denoted by uv. A string v is a suffix of s if there exists w such that $s = wv$.

Let $S_1^n = [S_1, S_2, \ldots, S_n]$ be a stationary ergodic stochastic process over alphabet $\mathcal{A}$, where $s_1^n = [s_1, s_2, \ldots, s_n]$ is its realization and $P(s_1^n) = P(S_1^n = s_1^n)$.

A *context tree* $\mathcal{T}$ is a set of strings such that no string $c \in \mathcal{T}$ is a suffix of another string $c' \in \mathcal{T}$. Each string $c = c_1^d \in \mathcal{T}$ can be visualized as a path from the root to a leaf consisting of d edges labeled by symbols $c_d c_{d-1} \cdots c_1$. We use λ to denote the empty context that corresponds to the root of the context tree.

Given a context tree $\mathcal{T}$, a parameter assignment $\theta(\mathcal{T})$ assigns to each suffix c' of a context $c \in \mathcal{T}$ (including to c itself) a vector of conditional probabilities $\theta(c') = [P(a_1|c'), P(a_2|c'), \ldots, P(a_m|c')]$, where $\sum_{a \in \mathcal{A}} P(a|c') = 1$.

We assign a probability $P(s|\mathcal{T}, \theta(\mathcal{T}))$ for the input sequence $s = s_1^n$ given the context tree $\mathcal{T}$ and the probability assignments $\theta(\mathcal{T})$ by

$$P(s|\mathcal{T}, \theta(\mathcal{T})) = \prod_{i=0}^{n-1} P(s_{i+1}|c_i), \tag{1}$$

where c_i is the longest suffix of s_1^i that belongs to $\mathcal{T}$.

A pair $(\mathcal{T}, \theta(\mathcal{T}))$ is called a tree model or a d-VLMC if the longest string in $\mathcal{T}$ has length d. In the case that all strings $c \in \mathcal{T}$ are of length d and $|\mathcal{T}| = m^d$ the tree model is called d-MC (full d-order Markov chain model). Given a tree model $(\mathcal{T}, \theta(\mathcal{T}))$ and an input sequence s_1^n, we can compute the probability of observing the sequence s_1^n given the model using (1).

Example Consider a binary context tree $\mathcal{T} = \{1, 00, 10\}$ and let $\theta(0) = [0.6, 0.4]$, $\theta(1) = [0.1, 0.9]$, $\theta(00) = [0.5, 0.5]$, $\theta(10) = [0.3, 0.7]$ and $\theta(\lambda) = [0.9, 0.1]$. Then from (1) for a sequence $s = 101000$ we obtain $P(s) = P(1|\lambda) \cdot P(0|1) \cdot P(1|10) \cdot P(0|1) \cdot P(0|10) \cdot P(0|00) = 0.1 \cdot 0.1 \cdot 0.7 \cdot 0.1 \cdot 0.1 \cdot 0.3 \cdot 0.5 = \frac{105}{10000000}$.

In the case that the tree $\mathcal{T}$ is known, but the parameter vector $\theta(\mathcal{T})$ is unknown, one can compute the *maximum likelihood estimator* (MLE) of the parameter vector denoted $\hat{\theta}(\mathcal{T})$. In particular, the MLE for a conditional probability $P(a|c)$ is:

$$\hat{P}(a|c) = \frac{N_n(c, a)}{N_n(c)},$$

where $N_n(c, a)$ and $N_n(c) = \sum_{a \in \mathcal{A}} N_n(c, a)$ are the number of occurrences of strings ca and c in s_1^n, respectively. For a given input sequence s_1^n we also use $ML = P(s_1^n|\hat{\theta}(\mathcal{T}))$ to denote the maximum likelihood of s_1^n.

For a tree $\mathcal{T}$ we define $d(\mathcal{T})$ to be the length of the longest context. If the tree $\mathcal{T}$ is understood then we use d instead. We also use $\mathcal{T}|_d$ to denote the tree that is a truncation of $\mathcal{T}$ to depth d.

In practice, given an input sequence s, a d-VLMC is built using a two-stage process. First a context tree of depth d is built from the sequence. Then the tree is pruned to obtain a variable-depth context tree that corresponds to contextual regularities in the input sequence. Figure 1 shows an example of a 2-MC (top) and a 2-VLMC (bottom) for an alphabet of size four. In the 2-VLMC case, the leaves $\{C, G\}A$ and $\{C, G, T\}C$, that are connected using a dashed edge to their parents, represent *virtual nodes*. A virtual node is created when a parent node loses between 2 and $m - 1$ children nodes as a result of pruning. The idea of virtual nodes is that they represent the context of the pruned children by merging the pruned children contexts together. Thus, the node $\{C, G\}A$ represents contexts CA and GA. Clearly, we do not create a virtual node if there is only one pruned child, as this would not change the total number of children.

We use $\mathcal{M}_\mathcal{T} = \{M(\mathcal{T}, \theta(\mathcal{T})) : \theta(\mathcal{T}) \in \Theta(\mathcal{T})\}$ to denote a set of all models sharing the same tree $\mathcal{T}$; here $\Theta(\mathcal{T})$ is the set of all valid parameter assignments to $\mathcal{T}$. If the input

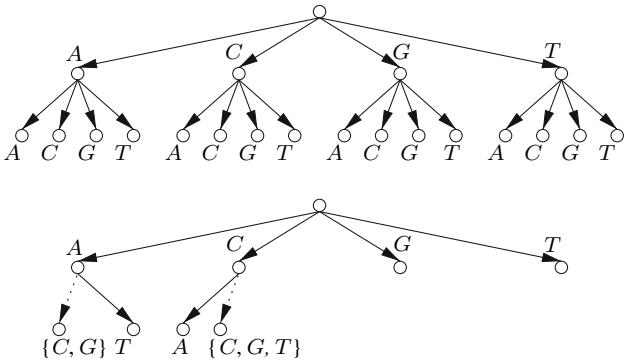

Fig. 1 An example of a 2-MC (*top*) and a 2-VLMC (*bottom*)

sequence has been generated by a VLMC, we denote by $\mathcal{T}_0$ the generating tree. Accordingly, θ_0 is the generating parameter vector and d_0 is the depth of the generating tree.

A *k-segmentation* of a sequence is a partition of the sequence in k consecutive segments. A *d-VLMC k-segmentation* is a segmentation by fitting a d-VLMC and having k segments. We also use the term *d-VLMC segmentation* to mean a segmentation that selects an optimal number of segments given a bound on the number of segments. We use the term BIC- or KT-segmentation with any of the above to specify the scoring function.

As an information criterion for model selection, we use a variant of the MDL principle that uses the KT code length. The MDL principle says that the best model of the process given the observed sequence is the one that gives the shortest description of the sequence, where the model itself is also a part of the description. For VLMCs, MDL has the following general form

$$\mathrm{MDL}_{\mathcal{T}}(s_1^n) = L_C(s_1^n | M(\mathcal{T}, \theta(\mathcal{T}))) + L_C(\mathcal{T}),$$

where $L_C(\cdot)$ is a real valued binary code length function of uniquely decodable binary code. Thus, $L_C(s_1^n | M(\mathcal{T}, \theta(\mathcal{T})))$ is the code length of the data given the model and $L_C(\mathcal{T})$ is the code length of the tree.

2.2 The Bayesian information criterion (BIC) and the Krichevski-Trofimov probability (KT)

In this section we review the known criteria for selecting an optimal context tree, where an optimal context tree is a trade-off between maximizing the maximum likelihood of the segment and the tree complexity.

BIC

For a d-MC the BIC has the following form [7]

$$\mathrm{BIC}_d(s_1^n) = -\log_2(ML_d(s_1^n)) + \frac{(m-1)m^d}{2}\log_2(n). \tag{2}$$

For a d-VLMC the BIC has the following form [7]

$$\mathrm{BIC}_{\mathcal{T}}(s_1^n) = -\log_2(ML_{\mathcal{T}}(s_1^n)) + \frac{(m-1)|\mathcal{T}|}{2}\log_2(n), \tag{3}$$

where

$$ML_{\mathcal{T}}(s_1^n) = P(s_1^d) \sum_{c\in\mathcal{T}, a\in\mathcal{A}} N_n(c,a) \left(\frac{N_n(c,a)}{N_n(c)}\right). \tag{4}$$

Thus, an optimal context tree of depth up to D, with respect to BIC, is defined as follows:

$$\hat{\mathcal{T}}_{\mathrm{BIC}}(s_1^n) = \min_{\mathcal{T}, d(\mathcal{T})\le D} \left(\mathrm{BIC}_{\mathcal{T}}(s_1^n)\right). \tag{5}$$

In coding terms BIC corresponds to the two-stage coding [10,8]. The likelihood terms ML_d and $ML_{\mathcal{T}}$ correspond to the code length of the data given the model while the additional terms $\frac{(m-1)m^d}{2}\log_2(n)$ and $\frac{(m-1)|\mathcal{T}|}{2}\log_2(n)$ correspond to the length of encoding of the parameters. In statistical terms, BIC has an interpretation as a maximum likelihood method. The first term measures the goodness of fit of the tree $\mathcal{T}$ to s_1^n, and the second term is the *penalty term* equal to the number of free parameters, which prevents BIC from overfitting.

KT

The KT probability [15] for a 0-MC binary sequence s_1^n is defined as the average probability of s_1^n over all possible parameter assignments $p = \theta \in [0,1]$ weighted by the Dirichlet distribution $\mathcal{D}(\mathbf{u})$ with parameters $\mathbf{u} = [\frac{1}{2}, \frac{1}{2}]$ (*Jeffrey's* prior) and it can be expressed as

$$\mathrm{KT}_0(s_1^n) = \int_0^1 p^{N_n(0)}(1-p)^{n-N_n(0)}\mathcal{D}(p|\mathbf{u})\mathrm{d}p, \tag{6}$$

where $N_n(0)$ is the number of zeros in s_1^n. In terms of Bayesian statistics, KT_0 corresponds to the *marginal likelihood* [17] of s_1^n. Equation (6) can be generalized to a multi-alphabet case ($m > 2$) by using the multinomial distribution in place of the binomial. It can be shown [15] that the integral has an exact solution

$$\mathrm{KT}_0(s_1^n) = \frac{\prod_{a\in\mathcal{A}}[\Gamma(N_n(a)+\frac{1}{2})]}{\Gamma\left(n+\frac{|A|}{2}\right)}. \tag{7}$$

The choice of prior parameter $\mathbf{u}$ in (6) is dictated by asymptotic properties [1] and has an effect as pseudo-counts in (7).

For VLMCs, KT can be expressed using the fact that all symbols corresponding to the same context $c \in \mathcal{T}$ form a memoryless subsequence of s_1^n, i.e, $P(s_1^n) = \prod_{c\in\mathcal{T}} \mathrm{KT}_0(s_1^n|c)$, where $s_1^n|c$ denotes a subsequence of s_1^n corresponding to context c. This leads to the following expression:

$$\mathrm{KT}_{\mathcal{T}}(s_1^n) = \frac{1}{|A|^k} \prod_{c\in\mathcal{T}, N_n(c)\ge 1} \mathrm{KT}_0(s_1^n|c). \tag{8}$$

KT is a minimizer of the *worst case average redundancy* $\overline{R}_n(\mathcal{T})$ for the model class determined by context tree $\mathcal{T}$, where $\overline{R}_n(\mathcal{T}) = \Theta\left(\frac{|\mathcal{T}|(m-1)}{2}\log(n)\right)$ [15].

In coding terms, KT corresponds to the mixture coding which consists of the encoding of s_1^n and the encoding of the tree [10,8]. Thus, the KT MDL estimator of an optimal context tree of depth up to D is defined as follows [28]:

$$\hat{\mathcal{T}}_{\mathrm{KT}}(s_1^n) = \min_{\mathcal{T}, d(\mathcal{T})\le D} \left(-\log_2(\mathrm{KT}_{\mathcal{T}}(s_1^n)) + L_C(\mathcal{T})\right). \tag{9}$$

In statistical terms, KT is a mixture distribution that measures the goodness of fit of the tree $\mathcal{T}$ to s_1^n in terms of the average probability in the model class $\mathcal{M}_{\mathcal{T}}$. In [27] a simple code is given that describes a context tree $\mathcal{T}$ using $L_C(\mathcal{T}) = \frac{m|\mathcal{T}|-1}{m-1}$ bits.

We now are ready to define the problem of optimal sequence segmentation using tree models.

2.3 Definition of the problem of optimal sequence segmentation using tree models

The problem of optimal sequence segmentation using tree models can be stated as follows. Given:

- $s = [s_1, s_2, \ldots, s_n]$: a finite sequence of categorical data over an alphabet $\mathcal{A}$, where $s_i^j = s_i s_{i+1} \ldots s_j$ over $\mathcal{A}$.
- K: the maximal number of segments.
- D: the maximal depth of the VLMC.
- $cost\left(s_i^j\right)$: a cost function for segment s_i^j.

find a vector of partition points $I = [i_1, i_2, \ldots, i_k]$, $1 \leq k \leq K-1$ such that

$$I = \arg \min_{[i_1, i_2, \ldots, i_k]} \left\{ \sum_{j=0}^{k} \text{cost}\left(s_{i_j}^{j_{j+1}-1}\right) + k \cdot B \right\},$$

where $1 \leq k \leq K-1$, $i_0 = 1$, $i_{k+1} = n$ and B is a border insertion penalty.

The cost function is either

$$\text{cost}\left(s_i^j\right) = \text{BIC}_{\hat{\mathcal{T}}_{\text{BIC}}}(s_i^j) \tag{10}$$

or

$$\text{cost}\left(s_i^j\right) = -\log_2\left(\text{KT}_{\hat{\mathcal{T}}_{\text{KT}}}(s_i^j)\right) + L_C(\hat{\mathcal{T}}_{\text{KT}}) \tag{11}$$

depending on the corresponding tree model selection method used. In experiments we used the size of the tree as the code length of the tree, i.e., we used $L_C(\hat{\mathcal{T}}_{\text{KT}}) = |\hat{\mathcal{T}}_{\text{KT}}|$.

2.4 Pruning the tree

Because it is infeasible in practice to enumerate all possible trees of a given maximum depth local search methods such as the *Context* algorithm [21] and the *context tree maximization* (CTM) algorithm [28] have to be used. The algorithms Context and CTM work in two stages. They first build a context tree of depth d and then they recursively prune the tree starting from the leaves and proceeding bottom-up.

Algorithm Context

Algorithm Context prunes a context tree as follows [5].

For every parent node w the algorithm considers every child node uw and marks it for pruning if $N_n(uw) < m$ or $\Delta_{uw} < K(n)$, where

$$\Delta_{uw} = \sum_{a \in \mathcal{A}} N_n(a, uw) \log_2\left(\frac{\hat{P}(a|uw)}{\hat{P}(a|w)}\right) \tag{12}$$

and $K(n)$ is a user defined threshold. If all children of w were marked for pruning then they are pruned and w becomes terminal. If at least two children were marked for pruning but there

is at least one non-marked child then the marked nodes are merged to create a virtual node, which represents the needed pruned contexts. In [21,22] the minimization of the stochastic complexity is used in place of (12) as a pruning criterion.

The context tree maximization (CTM)

We now present the original version of the CTM algorithm [28]. CTM finds a tree maximizing (8) by a local optimization in a recursive bottom-up way. For each node v in the tree CTM assigns two values: the maximum KT contribution to (8) of the contexts in the subtree rooted at v called $\mathrm{KT}_{\max}(v)$; and an indicator $I_{\max}(v)$ that marks nodes to be included in the maximizing tree. The algorithm proceeds bottom-up as follows:

1. **if** v is a leaf node **then** $\mathrm{KT}_{\max}(v) = \mathrm{KT}_0(s_1^n|v)$
2. **if** v is an internal node **then**

$$\mathrm{KT}_{\max}(v) = \max\left\{\mathrm{KT}_0(s_1^n|v), \prod_{a\in\mathcal{A}} \mathrm{KT}_{\max}(av)\right\} \tag{13}$$

3. **If** $\mathrm{KT}_0(s_1^n|v) < \prod_{a\in\mathcal{A}} \mathrm{KT}_{\max}(av)$ **then** $I_{\max}(v) = 1$ **else** $I_{\max}(v) = 0$.

After having visited all nodes, $\mathrm{KT}_{\max}(root)$ contains the maximized probability $\mathrm{KT}_{\hat{\mathcal{T}}_0}$ and $\hat{\mathcal{T}}_0$ has been marked by the indicators. To reconstruct $\hat{\mathcal{T}}_0$ one has to recursively read them off top-down starting from the root. In terms of pruning the value of $I_{\max}(v)$ has the following meaning: if $I_{\max}(v) = 0$ then all children are pruned at once (they are not part of $\hat{\mathcal{T}}_0$); while if $I_{\max}(v) = 1$ then the children are not pruned (they are a part of $\hat{\mathcal{T}}_0$). Comparing to *Context*, CTM has to visit all nodes in the tree.

3 TreeSegment: segmentation using tree models

In this section we present the TreeSegment algorithm that solves the problem of optimal segmentation using tree models as defined in Sect. 2.3. We start with discussing our pruning criteria.

3.1 Pruning according to Context algorithm to minimize the BIC of the tree

In this section we give a precise derivation for the threshold $K(n)$ of the Context algorithm that locally minimizes (3). There is a consensus in the literature [26,5] that $K(n)$ should be of the form $C\log(n)$; here we give a derivation for the value of C in detail.

We start with an example, illustrated in Fig. 2, which shows a tree rooted at a node w for alphabet $\mathcal{A} = \{A, C, G, T\}$. The tree undergoes a pruning scenario according to the Context algorithm. For simplicity we assume in this example that the virtual node is created after the

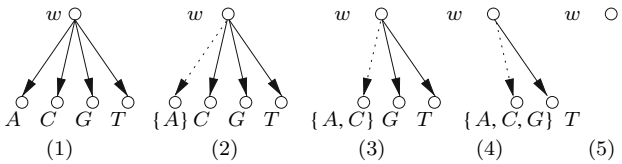

Fig. 2 VLMC pruning

first node is pruned. We number the trees (1)–(5) from the left to the right. Tree (1) shows the situation before the pruning algorithm starts. Tree (2) shows the situation after pruning node Aw to w, which results in creating a virtual node $\{A\}w$. Tree (3) shows the situation after pruning Cw to w, which results in updating the virtual node to represent context $\{A, C\}w$. Tree (4) shows the situation after pruning node Gw to w, which results in updating the virtual node to represent context $\{A, C, G\}w$. Finally, tree (5) shows the situation after the last node Tw has been pruned to w and w becomes terminal. Thus, in the presented scenario, the size of the tree $|\mathcal{T}|$ changes from m to 1 even though it does not decrease strictly monotonically after every pruning operation because of the need to create the virtual node. For simplicity we assume that every pruning operation decreases $|\mathcal{T}|$ by one. Let $\mathcal{T}_u$ be the tree $\mathcal{T}$ after the terminal node u has been pruned including a possible creation of a new virtual node.

Thus, we want to prune uw to w if and only if $\mathrm{BIC}_{\mathcal{T}_u}(s_1^n) < \mathrm{BIC}_{\mathcal{T}}(s_1^n)$, where $|\mathcal{T}| - |\mathcal{T}_u| = 1$, which leads to

$$\log_2\left(\frac{ML_{\mathcal{T}}(s_1^n)}{ML_{\mathcal{T}_u}(s_1^n)}\right) < \frac{(m-1)}{2}\log_2(n)$$

and from (12) we have

$$\sum_{a\in\mathcal{A}} N_n(a, uw)\log_2\left(\frac{\hat{P}(a|uw)}{\hat{P}(a|w)}\right) < \frac{(m-1)}{2}\log_2(n)$$

which finally gives us

$$K(n) = \frac{(m-1)}{2}\log_2(n).$$

3.2 Pruning according to CTM algorithm to minimize KT MDL of the tree

We modify the CTM algorithm (13) to locally minimize the KT MDL score (9) as follows:

$$\mathrm{MDL}(\mathrm{KT}_{\max}(v)) = \min\Bigg\{ -\log_2\left(\mathrm{KT}_0(s_1^n|v)\right) + 1, \sum_{a\in\mathcal{A}}\left(-\log_2\left(\mathrm{KT}_{\max}(av)\right) + |\hat{\mathcal{T}}_0(av)|\right)\Bigg\}, \tag{14}$$

where $|\hat{\mathcal{T}}_0(av)|$ is the size (number of contexts) of the optimal subtree rooted at node av. Thus, while (13) searches for a tree that maximizes the KT probability (14) searches for a tree that minimizes the corresponding KT MDL score. As in (13) (14) considers a local pairwise decision: parent versus children, i.e, whether the children should be part of the optimal tree or not that corresponds to pruning them off. Comparing to (13), (14) favors pruning the children by containing the term corresponding to the code length of the subtree rooted at v that is 1 bit if the children are pruned versus $\sum_{a\in\mathcal{A}}|\hat{\mathcal{T}}_0(av)|$ bits if the children are part of the optimal tree. As a result of it (14) produces a sparser tree than (13).

We also implemented a refinement of criterion (14) that considers all valid subsets of children for pruning instead of the two subsets consisting of all children (the parent node)

versus none of the children. The criterion is as follows:

$$\mathrm{MDL}(\mathrm{KT}_{\max}(v)) = \min_{\mathcal{X}} \Bigg\{ -\log_2 \left(\mathrm{KT}_0(s_1^n|v)\right) + 1, \sum_{a\in\mathcal{A}-\mathcal{X}} \left(-\log_2 \left(\mathrm{KT}_{\max}(av)\right) + |\hat{\mathcal{T}}_0(av)|\right) - \log_2 \left(\mathrm{KT}_0(s_1^n|\{\mathcal{X}\}v)\right) + 1 \Bigg\}, \tag{15}$$

where: $\mathcal{X}$ is a subset of children, where $|\mathcal{X}| = 0, 2, \ldots, |A| - 1$; v is the parent node; $\{\mathcal{X}\}v$ is the virtual node; and av is a child node.

Because of the need to consider all subsets of children criterion (15) is efficient only for small alphabet sizes and we found it useful for DNA in order to obtain a finer fitting of trees to s_1^n than by using criterion (14).

3.3 The segmentation algorithm

In this section we present the details of the algorithm TreeSegment. The standard optimal segmentation algorithm can be expressed by the following dynamic programming equation, due to Bellman [2]:

$$C[k, i] = \min_{k-1\le j\le i} \{C[k-1, j-1] + W[j, i]\}.$$

In the above equation, $C[k, i]$ is the optimal k-segmentation cost of the prefix s_1^i and $W[j, i]$ is the cost function (score) of the segment s_j^i, that is either the BIC (10) or the KT MDL (11) score as defined in Sect. 2.3. Computing $W[j, i]$ in time proportional to the length of the segment s_j^i leads to overall $O(n^3)$ running time, which is impractical for real-life sequences.

However, Algorithm 1 achieves a linear speedup by computing $W[j, i]$ in constant time (for a fixed alphabet size and depth of the tree). The main idea of the speedup is that for a fixed ending position i, a fixed-depth tree $\mathcal{T}_1$ is being built (inductively) starting at position i and proceeding backward for $j = i, i-1, \ldots, 0$. Thus, for every pair (j, i) $\mathcal{T}_1$ contains counts of all context strings that occur in segment s_j^i and for the next starting position $j-1$ $\mathcal{T}_1$ can be updated in constant time, since only one new context has to be added to it. After each updating of $\mathcal{T}_1$ it is copied to $\mathcal{T}_2$, which is pruned to obtain the score for s_j^i. Clearly, the cost of copying $\mathcal{T}_1$ to $\mathcal{T}_2$ and pruning of $\mathcal{T}_2$ is proportional to the size $|\mathcal{T}_1|$, which is also constant for fixed values of the parameters D and m. Since the computation of $W[j, i]$ can be done in a constant time for all pairs (j, i), the overall running time of algorithm TreeSegment is $\Theta(n^2)$. The space complexity of the algorithm is $\Theta(Kn)$.

We also find it very effective to compute the score $C[k, i]$ for values of i that are a multiple of a parameter Δ. Using this modification, we obtain a suboptimal solution, but the running time of the Algorithm is $\Theta((\frac{n}{\Delta})^2)$.

3.4 The maximum depth of the tree

Since we need to fit optimal trees to segments of varying length bounding the maximum depth of the tree is of a particular importance in TreeSegment for the following reasons: (i) it decreases the probability of overfitting while estimating the tree from a short sequence; and (ii) it reduces the unnecessary computational complexity of estimating a deeper tree. The only problem with the bound is that it may increases the probability of underestimation by restricting the context length. We use $D_{\max}(n) \le \log_m(n)$ as a bound, which follows from

Algorithm 1: Algorithm TreeSegment

Input: $\mathcal{A}, n, K, s_1^n, D, \Delta$
Output: $I = [i_1, i_2, \ldots, i_{k'}]$
begin

for $i = 1; i \leq n/\Delta; i = i + \Delta$ **do**
  $\mathcal{T}_1.init()$
  for $j = i; j \geq 0; j = j - \Delta$ **do**
    $\mathcal{T}_1.add(s^i_{j-\Delta+1})$;
    $N = i - (j - \Delta + 1) + 1$;
    if $N < m$ **then**
      $W[j] = \infty$;
    else
      $D_{\max}(N) = \max_{0 \leq d \leq D} \left(d \leq \log_m(N)\right)$;
      $\mathcal{T}_2 = \mathcal{T}_1|_{Dmax(N)}$;
      $\mathcal{T}_2.prune()$;
      $W[j] = score(\mathcal{T}_2)$;
    end
  end
  for $k = 2; k \leq K; k = k + 1$ **do**
    $C[k, i] = \min_{k-1 \leq j < i} C[k-1, j] + W[j+1]\}$;
  end
end
$BackTrack()$;
end

the fact that the if we assume that an occurrence of every context in a given position in the sequence is equally likely with probability $\frac{1}{m^d}$ then the length of the sequence has to be at least m^d to guarantee that on-average every context occurs at least once.

3.5 Border insertion penalties

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. Also, in MDL terms each partition point should be treated as an additional parameter and penalized appropriately. Based on our extensive experiments we selected the following penalties for the BIC and the KT scoring methods: $B_{\text{BIC}} = (K-1)\frac{\log_2(n)}{2}$ and $B_{\text{KT}} = \sum_{k=2}^{K} \log_2 \left(\frac{n}{k-1}\right)$, where K is the total number of segments. Clearly, B_{BIC} follows from the BIC as a parameter penalty. B_{KT} follows from MDL by observing that to encode the following partition points we need proportionally fewer bits, i.e, we need roughly $\log_2(n)$ bits for the first point, $\log_2(\frac{n}{2})$ for the second and so on.

4 Experiments

To evaluate results of segmentations obtained by TreeSegment we used the following distance measure $D_{\text{seg}}(A, B)$:

$$D_{\text{seg}}(A, B) = \max\{D(A, B), D(B, A)\}, \tag{16}$$

where

$$D(A, B) = \frac{1}{m} \sum_{a \in A} \min_{b \in B} \left\{ \frac{d(a, b)}{n} \right\},$$

and A and B represent the sets of partition points of two segmentations. The measure $D(A, B)$ captures the distance of each segmentation point in A to the closest segmentation point in B on average. The distance is measured as a fraction of the total length of the sequence. So, the measure $D(A, B)$ takes values between 0 and 1, where the value 0 means that the two segmentations A and B are identical, while the value 1 can be obtained only for segmentations with one segmentation points (and being at opposite ends). Thus, we compute the measures $D_{\text{seg}}(\text{BIC, SOURCE})$ and $D_{\text{seg}}(\text{KT, SOURCE})$, where SOURCE, BIC, and KT are the sets of partition points corresponding to a known segmentation of the source, the BIC segmentation, and the KT segmentation, respectively.

We conducted our experiments using the following sources: synthetic data in Sect. 4.1, DNA in Sect. 4.2 and text data in Sect. 4.3.

4.1 Generated data

In this section we use a synthetic sequence to show the advantage of VLMCs over MCs in segmentation. In short, VLMCs are advantages over MCs because if segments were generated using VLMCs then MCs may miss the partition points. This follows from the fact that by using tree models we can fit high order models to segments to maximize the likelihood, without being penalized for an exponential increase in the number of parameters (as in the case of ordinary MCs).

As an illustration of such a case consider the following example synthetic sequence S_{12} of length $2n$ over the alphabet $\mathcal{A} = \{A, C, G, T\}$ that was composed of two segments S_1 and S_2 of length n each generated as follows:

- S_2 was generated from a 2-VLMC model $M_2(T_2, \theta(T_2))$, where tree $\mathcal{T}_2$ has only 5 nodes at depth 2; and
- S_1 was generated from a 1-MC $M_1(\mathcal{T}_1, \theta(T_1))$, where $\mathcal{T}_1 = T_2|1$ and $\theta(T_1) = \theta(T_2|1))$, where $T_2|1$ is a truncation of tree T_2 upto depth 1. Thus, M_1 and M_2 are identical in terms of 1-MC and 0-MC.

The models generating S are presented in Fig. 3. Thus, since M_1 and M_2 are identical in terms of 1-MC and 0-MC MC segmentation methods will select M_1 as the underlying model for the whole sequence S to avoid being penalized for a full 2-MC for S_2, i.e., for useless

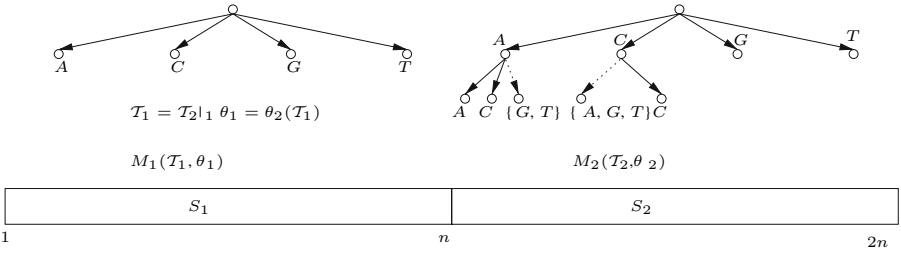

Fig. 3 An example sequence S_{12} that requires VLMC segmentation to discover the partition point. Segment S_1 was generated from model $M_1(\mathcal{T}_1, \theta(T_1))$ that is a truncation upto depth 1 of model $M_2(T_2, \theta(T_2))$ that generated segment S_2

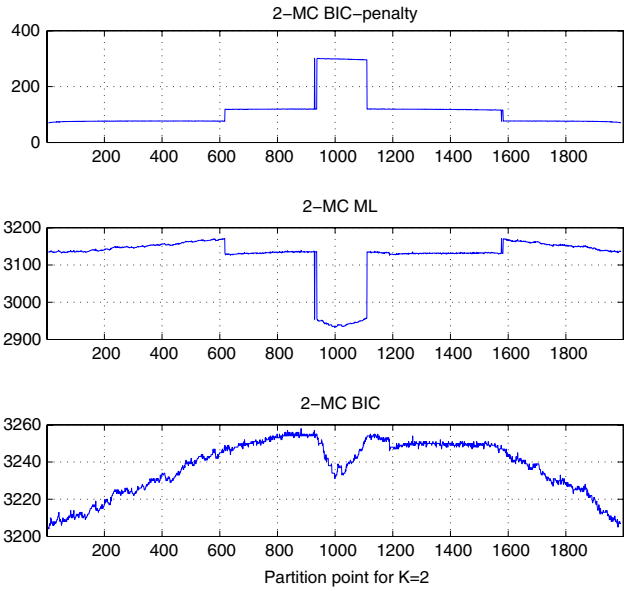

Fig. 4 BIC 2-MC method and the corresponding optimal segmentation cost for the 2-segmentation of the synthetic sequence S_{12} as a function of the partition point. *Top*: the BIC penalty. *Middle*: the minus maximum likelihood. *Bottom*: the BIC score

parameters corresponding to the non-existent contexts in the generating tree T_2. But a VLMC segmentation method will properly select T_2 for S_2 without paying for the the non-existent contexts and therefore it will properly select two segments with the partition point in the middle of S.

We now present figures that show the behavior of the optimal segmentation cost (3.3) for the 2-segmentation ($k = 2$) of S_{12} as a function of the partition point j for the BIC 2-MC, BIC 2-VLMC and KT 2-VLMC segmentation methods. Each figure in this section consists of 3 subplots, where each subplot presents a quantity of interest plotted as a function of the partition point in the 2-segmentation of S_{12}. Also, the upper two subplots depict quantities that when added up are equal to the quantity in the third subplot.

Figure 4 presents the BIC 2-MC case. We present this 2-segmentation case for comparison with the VLMC methods since we know that the algorithm selected 1-segmentation using 1-MC instead of the 2-segmentation. The plots of the BIC-penalty and ML are "choppy" because the model selection machinery is very poor in this case since there are only 3 trees (2-MC, 1-MC and 0-MC) available to fit to both segments. Also, this figure reveals the fundamental fact of the BIC segmentation namely the contribution of the penalty is not uniform with respect to the partition point placement. The reason is that the total BIC-penalty for the whole sequence is equal to $\frac{m^{k_1}(m-1)}{2}\log_2(x) + \frac{m^{k_2}(m-1)}{2}\log_2(n-x)$ where x is the partition point. Then clearly if $k_1 = k_2$ then the factor $\log_2(x(n-x))$ is the source of problem since $x(n-x)$ is a square function of x with the maximum at $x = \frac{n}{2}$.

Figure 5 presents the BIC 2-VLMC method. The area under the BIC-penalty curve is smaller for the 2-VLMC comparing to the 2-MC since here the BIC charges for only the relevant contexts. This example clearly shows the superiority of VLMCs over MCs in segmentation.

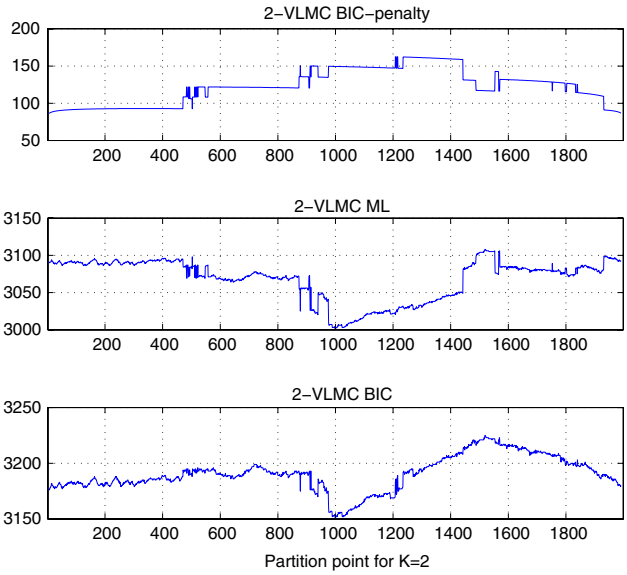

Fig. 5 BIC 2-VLMC method and the corresponding optimal segmentation cost for the 2-segmentation of the synthetic sequence S_{12} as a function of the partition point. *Top*: the BIC penalty. *Middle*: the minus maximum likelihood. *Bottom*: the BIC score

Figure 6 show the results for the KT 2-VLMC method and in particular it compares the MDL KT score (11) with the ML. A comparison of Figs. 5 and 6 reveals that KT gives a more uniform penalty then BIC but at the expense of a flatter score characteristic.

4.2 DNA sequences

In this section we present the results of experiments on DNA sequences. In particular, we consider DNA sequences containing genes and their flanking regions.

We distinguish the following structural regions in DNA [6,29]:

- *5′ UTR* and *3′ UTR* (untranslated regions)
- *CDS* (coding region) on the directed and complementary strand
- *intron*
- *5′ flanking* and *3′ flanking* regions,

where CDSs and UTRs are part of an *exon*. Thus, we consider a total of seven functional regions to be segmented by TreeSegment.

In our experiments, we start with studying viral genomes in Sect. 4.2.1, and then we consider eukaryotic genes in Sect. 4.2.2. We obtained our viral gene sequences from http://www.ncbi.nlm.nih.gov and the eukaryotic gene sequences from http://www.ensembl.org.

4.2.1 Viral genomes

The first experiment tests whether TreeSegment discovers any tree structure variation in the genomes with the simplest possible structure. For this purpose we selected complete viral genomes from a subset of ssRNA positive-strand viruses that contain exactly three segments: UTR5′, CDS and UTR3′ and segmented them. Our results revealed the following facts: (*i*)

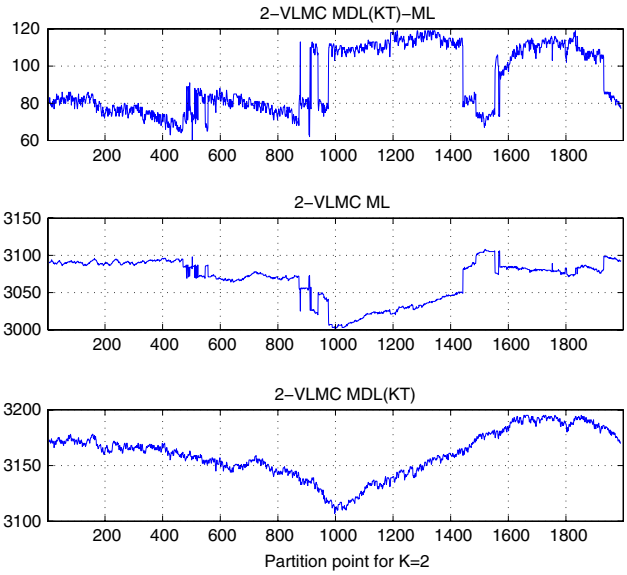

Fig. 6 KT 2-VLMC method and the corresponding optimal segmentation cost for the 2-segmentation of the synthetic sequence S_{12} as a function of the partition point. *Top*: the KT MDL score minus the maximum likelihood. *Middle*: the minus maximum likelihood. *Bottom*: the KT MDL score

for every genome TreeSegment delineates at most three segments, where for most of the genes it delineates exactly three segments; (ii) every CDS segment corresponds to a tree of depth $1 \leq D \leq 2$; (iii) the UTR segments correspond to trees of depth $D = 0$; and (iv) all the CDS segments have a common subtree of depth $D = 1$ consisting of contexts C and T.

We now show detailed results for Wisteria vein mosaic virus genome (accession point NC_007216) that is a member of the family of ssRNA positive-strand viruses. Figure 7 shows segmentation results and Fig. 8 shows a tree built from the CDS region.

In the following experiment we segmented *Bacteriophage lambda* virus genome that has a long history of being used as a test sequence for demonstrating new segmentation techniques [4,16]. The sequence mostly contains overlapping CDS segments from both DNA strands. Figure 9 shows results for *Bacteriophage lambda* that are consistent with [4,16].

4.2.2 Eukaryotic genes

We now consider a more difficult task of segmenting eukaryotic genes for which the TreeSegment needs to be capable of discovering differences in tree structures between introns and exons. Therefore, we first investigate those differences. For this purpose, we repeated the following experiment for many eukaryotic organisms. We first scanned the respective genomes and then extracted the corresponding exons and introns to separate sequences. Then we fitted context trees for those two kinds of sequences using algorithm Context. As an example we present results for *Caenorhabditis elegans* genome in Table 1.

The results show that there is a structural difference between the intron and exon trees, where the intron tree is bigger ($|\mathcal{T}| = 23$ versus $|\mathcal{T}| = 3$) and ($D = 3$ versus $D = 2$). Also the exon sequence has a higher CG content while the intron sequence has a higher AT content [9].

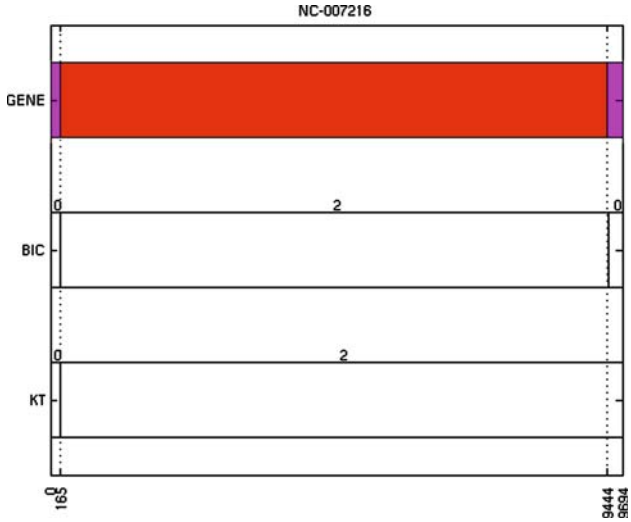

Fig. 7 Segmentations obtained for *Wisteria vein* mosaic virus genome. *Top*: annotated sequence. *Middle*: segmentation obtained by BIC–VLMC method. *Bottom*: segmentation obtained by KT–VLMC method. The numbers above each segment denote the depths of the corresponding trees

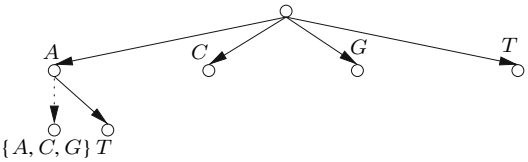

Fig. 8 CDS region of *Wisteria vein* mosaic virus NC_007216

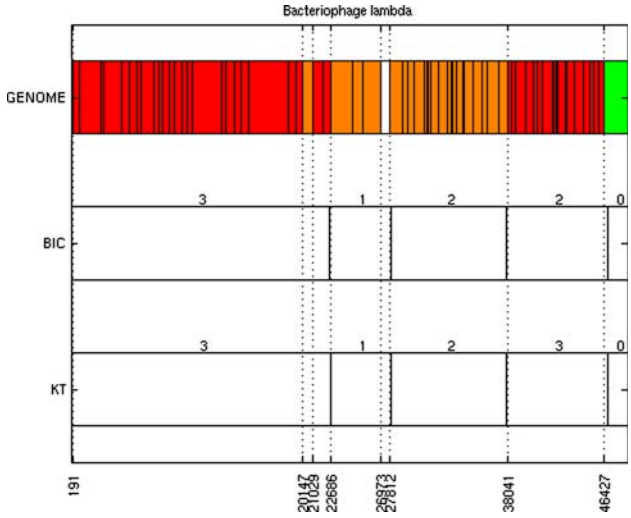

Fig. 9 Segmentations obtained for *Bacteriophage lambda* genome. *Top*: annotated sequence. *Middle*: segmentation obtained by BIC–VLMC method. *Bottom*: segmentation obtained by KT–VLMC method. The *numbers* above each segment denote the depths of the corresponding trees

Table 1 Comparison of intron and exon trees for *C. elegans*

Tree properties			Base content			
Tree	$\|T\|$	D	C	G	A	T
Intron	23	3	0.168	0.164	0.337	0.327
Exon	8	2	0.232	0.239	0.286	0.242

Table 2 Summary of segmentation results for DNA using BIC and KT segmentation methods, where D_{seg} is the distance measure and k is the number of segments

Gene		BIC		KT	
Name	k	D_{seg}	k	D_{seg}	k
Wisteria vein	3	0.0005	3	0.0065	2
Bacteriophage lambda	8	0.0149	5	0.0150	5
Drosophila melanogaster					
CG10045-RA	4	0.0144	3	0.0114	3
CG10045-RA+flanking	6	0.0309	6	0.0291	5
CG5407-RA	6	0.0383	7	0.0341	10
Caenorhabditis elegans					
F33E11.3	9	0.0243	8	0.0034	10
Y50D4C.3	9	0.0874	4	0.0336	12
Tetraodon nigroviridis					
GSTENT00014173001	7	0.0316	13	0.0567	8
Homo Sapiens					
ENST00000246662	14	0.0348	10	0.0590	8
Pan troglodytes					
Intron7	6	0.0944	4	0.0946	4

Given the discovered differences in tree structures, we segmented 10 example genes. The results are presented in Table 2. By comparing the D_{seg} distance measure for BIC and KT we can conclude that both methods perform comparably.

Below we present details of segmentations form Table 2. To check whether TreeSegment recognizes partition points between flanking regions and exons we segmented first *Drosophila melanogaster* gene CG10045-RA, and then we segmented a sequence composed of that gene and flanking regions of length 1,000. The results are shown in Fig. 10, where GENE=[UTR5′, Intron12, CDS2, UTR3′] and in Fig. 11, where GENE=[Flanking5′, UTR5′, Intron12, CDS2, UTR3′, Flnaking3′]. Clearly, after adding the flanking regions the origins of the first exon and the second exon have been properly recognized by the BIC and KT methods. Figure 12 shows segmentation of *Drosophila melanogaster* gene CG5407-RA.

Figure 13 shows segmentation of *Caenorhabditis elegans* gene F33E11.3. The gene structure is as follows: GENE=[CDS1, Intron12, CDS2, Intron23, CDS3, Intron34, CDS4, Intron45 and CDS5]. The BIC method merged two regions: Intron23, CDS3 into one region while the KT method recognized all gene regions.

Figure 14 shows segmentation of *Caenorhabditis elegans* gene Y50D4C.3. The gene structure is as follows: GENE=[CDS1, Intron12, CDS2, Intron23, CDS3, Intron34, CDS4, Intron45, CDS5]. The BIC method merged seven consecutive regions starting from CDS2

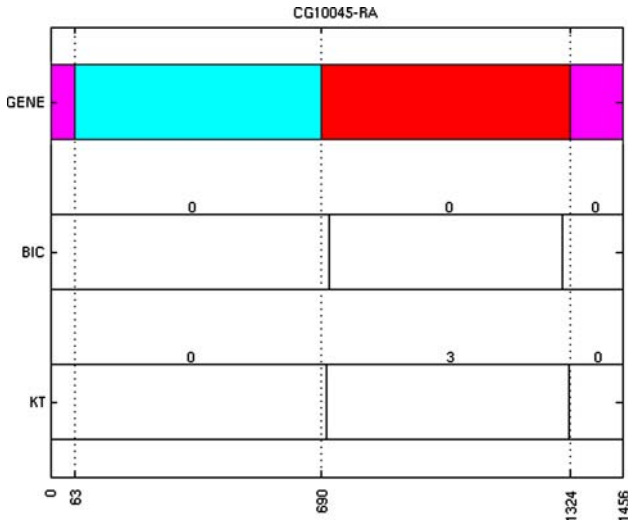

Fig. 10 Segmentations obtained for *Drosophila melanogaster* gene CG10045-RA. *Top*: annotated sequence. *Middle*: segmentation obtained by BIC–VLMC method. *Bottom*: segmentation obtained by KT–VLMC method. The *numbers* above each segment denote the depths of the corresponding trees

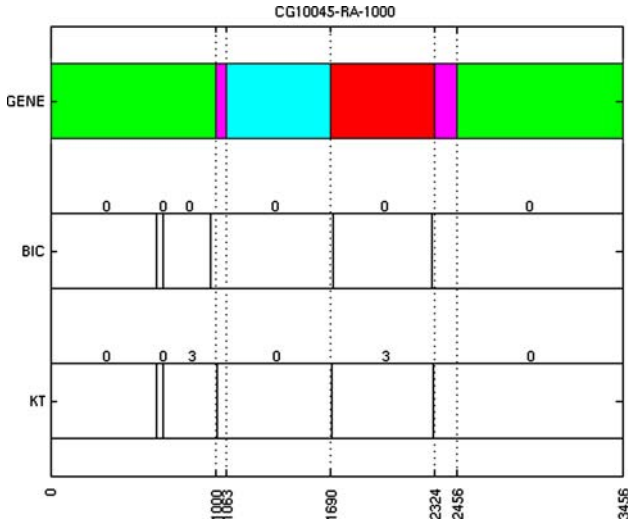

Fig. 11 Segmentations obtained for *Drosophila melanogaster* gene CG10045-RA + flanking regions. *Top*: annotated sequence. *Middle*: segmentation obtained by BIC–VLMC method. *Bottom*: segmentation obtained by KT–VLMC method. The *numbers* above each segment denote the depths of the corresponding trees

while the KT method merged only four regions from CDS2 to Intron34. Also the KT method produced more segments than the annotated segmentation, while the BIC method produced fewer segments.

Figure 15 shows segmentation of *Tetraodon nigroviridis* gene GSTENT00014173001 for which the BIC method seems to have recognized all gene segments while the KT method

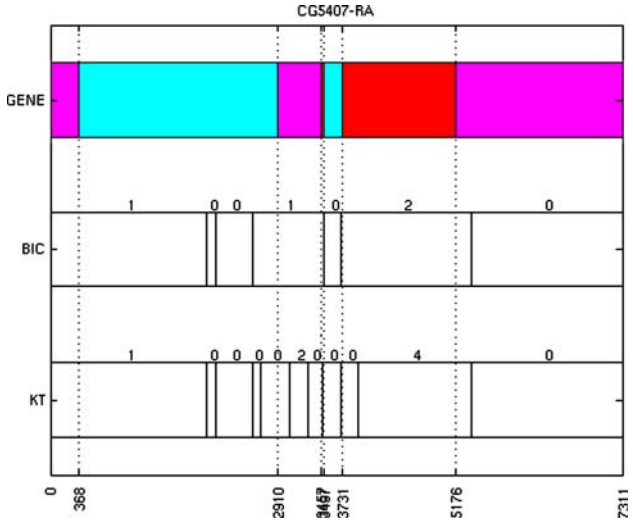

Fig. 12 Segmentations obtained for *Drosophila melanogaster* gene CG5407-RA. *Top*: annotated sequence. *Middle*: segmentation obtained by BIC–VLMC method. *Bottom*: segmentation obtained by KT–VLMC method. The it numbers above each segment denote the depths of the corresponding trees

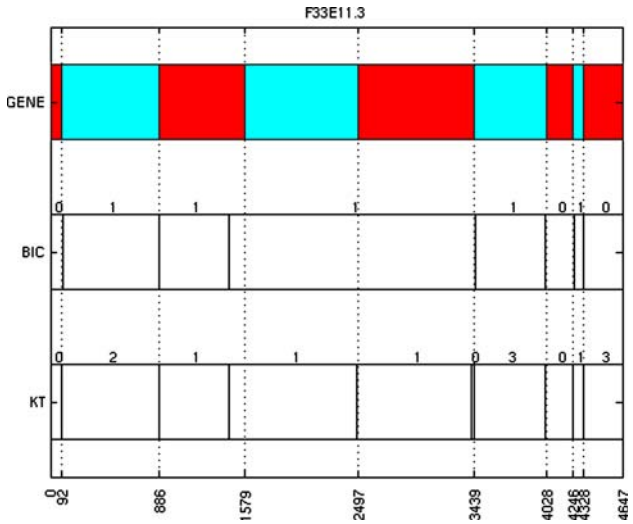

Fig. 13 Segmentations obtained for *Caenorhabditis elegans* gene F33E11.3. *Top*: annotated sequence. *Middle*: segmentation obtained by BIC–VLMC method. *Bottom*: segmentation obtained by KT–VLMC method. The *numbers* above each segment denote the depths of the corresponding trees

merged three gene regions. Also unlike in the case of gene Y50D4C.3 here the BIC method produced more segments than the KT method.

Figure 16 shows segmentation of *Homo Sapiens* gene ENST00000246662.

Figure 17 presents segmentation of intron7 of Pan troglodytes (chimpanzee) alpha-fetoprotein precursor (AFP) gene, where intron7 is know to contain distinct homogeneous segments [20]. The results are consistent with [20].

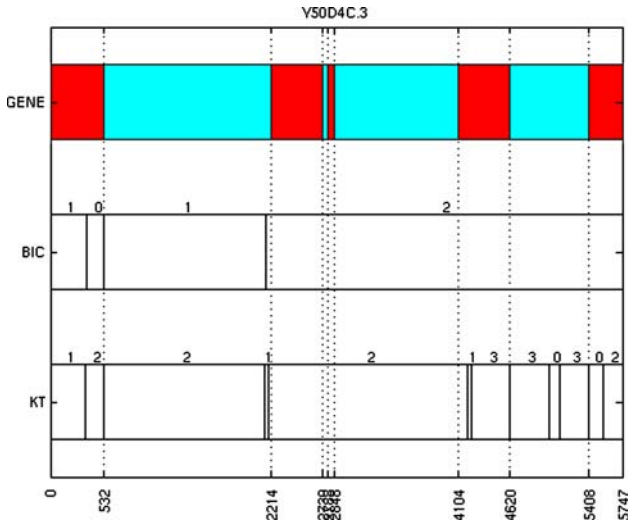

Fig. 14 Segmentations obtained for *Caenorhabditis elegans* gene Y50D4C.3. *Top*: annotated sequence. *Middle*: segmentation obtained by BIC–VLMC method. *Bottom*: segmentation obtained by KT–VLMC method. The *numbers* above each segment denote the depths of the corresponding trees

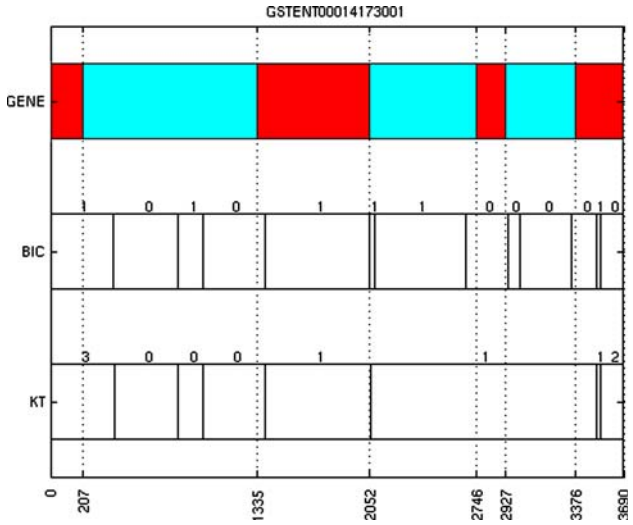

Fig. 15 Segmentations obtained for *Tetraodon nigroviridis* gene GSTENT00014173001. *Top*: annotated sequence. *Middle*: segmentation obtained by BIC–VLMC method. *Bottom*: segmentation obtained by KT–VLMC method. The *numbers* above each segment denote the depths of the corresponding trees

Concluding, the presented experiments on DNA sequences reveal two main properties of TreeSegment: (i) it tends to recognize boundaries exon–intron in cases where the corresponding segments are appropriately long; and (ii) it tends to merge consecutive smaller heterogeneous segments into one larger segment. Property (i) follows from the fact that long segments enable building appropriately large trees that may better fit to the segments in order to discover a finer difference between them. Property (ii) follows from the fact that the MDL

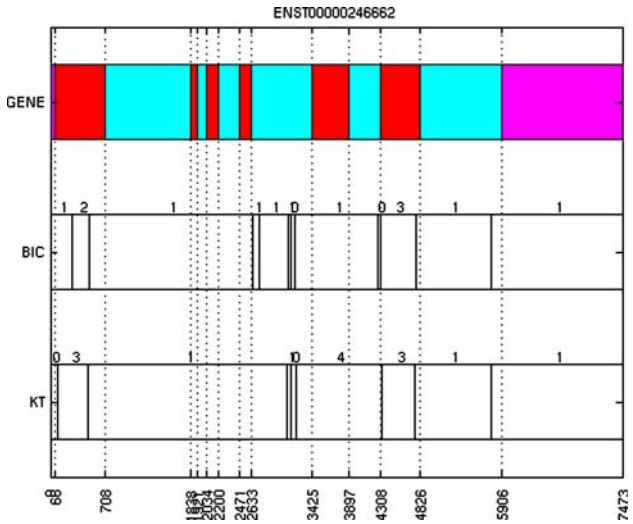

Fig. 16 Segmentations obtained for *Homo Sapiens* gene ENST00000246662. *Top*: annotated sequence. *Middle*: segmentation obtained by BIC–VLMC method. *Bottom*: segmentation obtained by KT–VLMC method. The *numbers* above each segment denote the depths of the corresponding trees

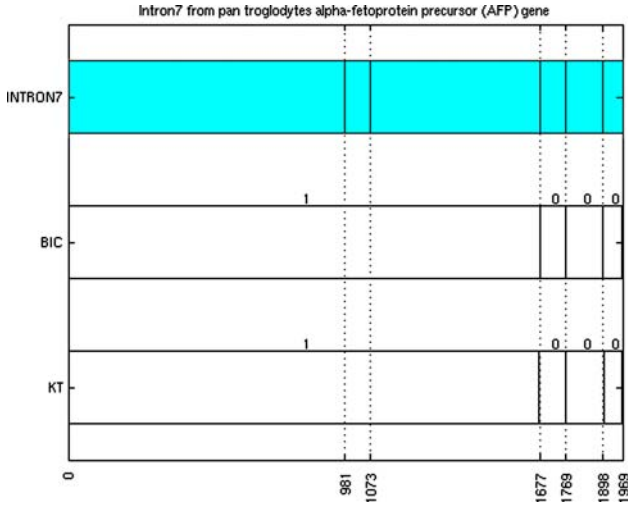

Fig. 17 Intron7 from *Pan troglodytes alpha-fetoprotein precursor (AFP)* gene (accession U21916). *Top*: known regions. *Middle*: segmentation obtained by BIC–VLMC method. *Bottom*: segmentation obtained by KT–VLMC method. The *numbers* above each segment denote the depths of the corresponding trees

criteria employed in TreeSegment favor a simpler model that spans a larger region instead of creating smaller segments to represent local fluctuations of the probabilistic behavior of the sequence.

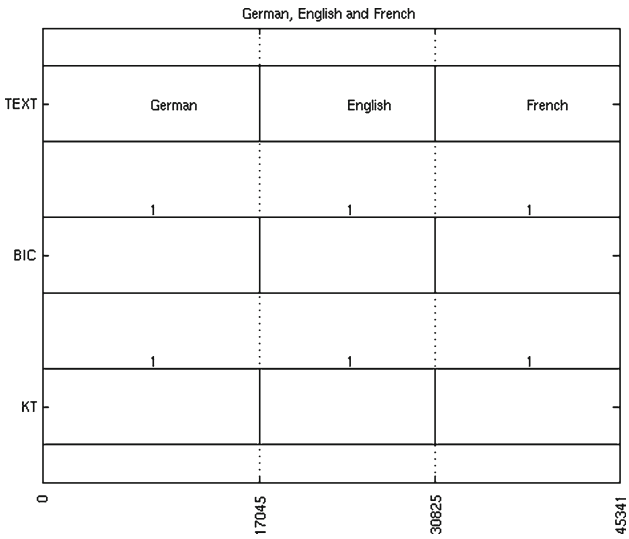

Fig. 18 Multilingual text separation: German, English and French. *Top*: segments corresponding to the three languages. *Middle*: segmentation obtained by BIC–VLMC method. *Bottom*: segmentation obtained by KT–VLMC method. The *numbers* above each segment denote the depths of the corresponding trees

4.3 Text

In this section we experiment with natural language text corpus. In Sect. 4.3.1 we segment a multilingual text and in Sect. 4.3.2 we experiment with corrupted context separation. We converted the original texts to an alphabet of cardinality 27, which is a set consisting of the English alphabet and the space character. Every space character in the converted text corresponds to a sequence of white spaces between normal characters in the original text.

4.3.1 Multilingual text separation

In this experiment we constructed a text sequence composed from three segments: German, English and French translations of Chapter 11 of *Robur the Conqueror* by Jules Verne. We obtained the text data from http://www.gutenberg.org. In order to convert the German and the French texts to the English alphabet we performed the following conversions. In the case of the German text we converted the *sharp s* to "ss" and the umlauts to "ae", "oe" and "ue", respectively. In the case of the French text we stripped off all accents. Figure 18 presents results of the segmentation that show a very accurate separation between the languages. As the figure shows both the BIC and the KT methods selected 1-VLMCs for each segment. As it turns out 0-MC can also separate the texts however not as accurately as 1-VLMC.

4.3.2 Corrupted context separation

In this experiment we constructed a text sequence composed from three segments: a scientific text segment of length 5,000, a reversed version of the first segment and a copy of the first segment. The purpose of this experiment was to test whether the algorithm was able to discover the corrupted context segment. This experiment presents the following two challenges

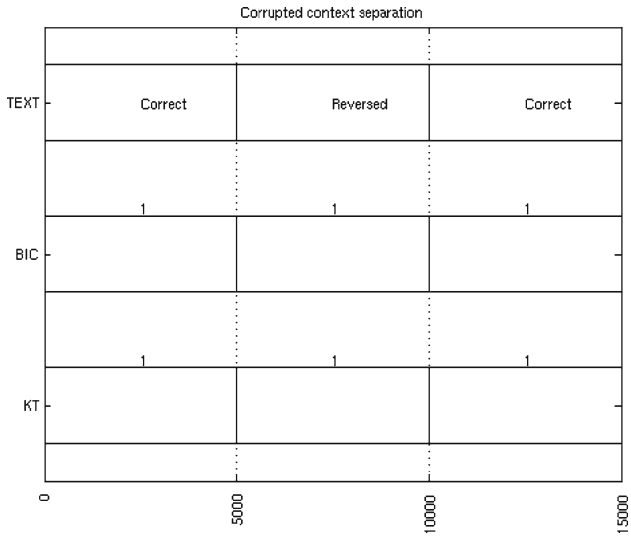

Fig. 19 Corrupted context separation. *Top*: original segment. *Middle*: segmentation obtained by BIC–VLMC method. *Bottom*: segmentation obtained by KT–VLMC method. The *numbers* above each segment denote the depths of the corresponding trees

for the algorithm: (i) all three segments have the same symbol composition such that the whole sequence is homogeneous in terms of 0-MC; and (ii) since the outer two segments are identical they correspond the same context tree making the tree prevalent across the sequence. As it turns out 1-MC is enough to separate the segments. Figure 19 presents results that show an almost perfect separation.

5 Related work

Tree models and the algorithm Context were introduced by Rissanen in [21]. Consistency results for tree models were provided by Weinberger et al. in [26] and by Bühlmann and Wyner in [5] who also defined the term VLMC. The CTM was introduced by Wilems et al. in [27,28]. In [15] Krichevsky and Trofimov introduced KT and derived its asymptotic properties. In [1], Barron et al. presented a comprehensive review of theoretical results on MDL in the context of coding and modeling. The Bayesian information criterion (BIC) was introduced by Schwarz in [24]. In [7], Ciszar and Talata proved consistency results for BIC and KT as estimators of the optimal context tree.

Segmentation algorithms have been central in the analysis of genomic sequences. In [17], Liu and Lawrence presented a Bayesian approach to DNA segmentation by assuming a 0-MC model and using the KT probability. The optimal number of segments was selected using Bayesian inference. Makeev et al. [18] studied a Bayesian approach to DNA segmentation by extending the idea from [17] by using heuristic border insertion penalties and filtration of boundaries.

Orlov et al. [19] presented a method for recognizing functional DNA sites and segmenting genomes. They developed a program "Complexity" for computing a context tree of a DNA sequence using the stochastic complexity [21,22] as a pruning criterion. Using their

program they analyzed DNA sequences of various functional classes (coding, non-coding and regulatory) and discovered that the DNA structure can be represented by trees.

The problem of DNA segmentation by model selection was posed by Li [16], where he considered a greedy top-down divide-and-conquer 0-MC segmentation approach to segment DNA by using the BIC and the Akaike information criterion but he did not consider gene segmentation. Also in [25], Szpankowski used 0-MC and the Shanon-Jensen distance to segment DNA.

As far as gene segmentation the distinctive statistical properties of gene functional regions are well documented in the literature, e.g., see [9,29,11]. In particular the detection of genes has been based on the non-uniform codon usage in protein coding segments and has been modeled by non-uniform Markov models and HMMs [6]. Bernaola et al. [3] proposed using entropic segmentation for finding borders between coding and non-coding DNA regions.

6 Conclusions

We presented a segmentation method that uses tree models to partition the input sequence into segments of differing contextual regularities that correspond to differnt tree structures. The MDL principle is used to guide the segmentation process by deciding the optimal tree model in each segment and by deciding the overall number of segments. In our experiments we demonstrated that VLMCs can provide more accurate segmentations than MCs and are also capable of recognizing partition points in cases where MCs fail. In the experiments on DNA we showed usefulness of our method for gene segmentation.

References

1. Barron A, Rissanen J, Yu B (1998) The minimum desiption length principle in coding and modeling. IEEE Trans Inf Theory 44(6):2743–2760
2. Bellman R (1961) On the approximation of curves by line segments using dynamic programming. Commun ACM 4(6):284
3. Bernaola-Galvan P, Grosse I, Carpena P, Oliver J, Roman-Roland R, Stanley H (2000) Finding borders between coding and noncoding dna regions by an entropic segmentation method. Phys Rev Lett 85(6):1342–1345
4. Braun J, Muller H (1998) Statistical methods for dna sequence segmentation. Statist Sci 13(2):142–162
5. Bühlmann P, Wyner A (1999) Variable length Markov chains. Ann Statist 27:480–513
6. Burge Ch, Karlin S (1997) Prediction of complete gene structures in human genomic dna. J Mol Biol 268:78–94
7. Csiszar I, Talata Z (2006) Context tree estimation for not necessarily finite memory processes, via bic and mdl. IEEE Trans Inf Theory 52(3):1007–1016
8. Grünwald P (2005) A tutorial introduction to the minimum description length principle. In: Advances in minimum description length: theory and applications. MIT Press
9. Guigo R, Fickett J (1995) Distinctive sequence features in protein coding genic non-coding, and intergenic human dna. J Mol Biol 253:51–60
10. Hansen M, Yu B (2001) Model selection and the principle of minimum description length. J Am Statist Assoc 96(454):746–774
11. Herzel H, Grosse I (1997) Correlations in dna sequences: the role of protein coding segments. Phys Rev Lett 55(1):800–810
12. Mannila H, Tikanmki J, Himberg J, Korpiaho K, Toivonen H (2001) Time series segmentation for context recognition in mobile devices. In: First IEEE international conference on data mining, pp 203–210
13. Kehagias Ath (2004) A hidden markov model segmentation procedure for hydrological and environmental time series. Stoch Environ Res Risk Assess (SERRA) 18(2):117–130
14. Keogh EJ, Chu S, Hart D, Pazzani MJ (2001) An online algorithm for segmenting time series. In: ICDM, pp 289–296

15. Krichevsky R, Trofimov V (1981) The performance of universal encoding. IEEE Trans Inf Theory IT-27(2):199–207
16. Li W (2001) DNA segmentation as a model selection process. In: International conference on research in computational molecular biology, pp 204–210
17. Liu S, Lawrence C (1999) Bayesian inference of biopolymer models. Bioinformatic 15:38–52
18. Makeev V, Ramensky V, Gelfand M, Roytberg M, Tumanyan V (2000) Bayesian approach to dna segmentation into regions with different average nucleotide composition. Lecture Notes in Computer Science, 2066:54–73, Computational Biology
19. Orlov Y, Potapov V, Filipov V (2002) Recognizing functional dna sites and segmenting genomes using the program "complexity". In: Proceedings of BGRS 2002, vol 3. Novosibirsk Insititute of Cytology and Genetics Press, pp 244–247
20. Henderson D, Boys R, Wilkinson D (2000) Detecting homogeneous segments in dna sequences by using hidden markov models. Appl Statist 49(2):269–285
21. Rissanen J (1983) A universal data compression system. IEEE Trans Inf Theory IT-29(5):656–664
22. Rissanen J (1999) Fast universal coding with context models. IEEE Trans Inf Theory 45(4):1065–1071
23. Salmenkivi M, Mannila H (2005) Using markov chain monte carlo and dynamic programming for event sequence data. Knowl Inf Systems 7(3):267–288
24. Schwarz G (1978) Estimating the dimension of a model. Ann Statist 7(2):461–464
25. Szpankowski W, Ren W, Szpankowski L (2003) An optimal DNA segmentation based on the MDL principle. In: IEEE computer society bioinformatics conference, pp 541–546
26. Weinberger M, Rissanen J, Feder M (1995) A universal finite memory source. IEEE Trans Inf Theory 41(3):643–652
27. Willems F, Shtarkov Y, Tjalkens T (1995) The context-tree weighting method: basic properties. IEEE Trans Inf Theory IT-41:653–664
28. Willems F, Shtarkov Y, Tjalkens T (2000) Context tree maximizing. In: Conference on information sciences and systems, pp 7–12
29. Zhang M (1998) Statistical features of human exons and their flanking regions. Hum Mol Genet 7(5): 919–932

Authors biography

Robert Gwadera received the M.S. degree in Electrical and Computer Engineering from Technical University of Gdansk, Poland in 1995. In 2003 he received the M.S. in Computer Sciences from Purdue University. He received his Ph.D. in Computer Science from Purdue University in 2005. He is currently a postdoctoral researcher in the Laboratory of Computer and Information Sciences, Helsinki University of Technology. His research interests are data mining, machine learning and databases.

Aristides Gionis received his Ph.D. from Stanford University in 2003, and he is currently a senior researcher at Yahoo! Research, Barcelona. His previous positions include being a senior researcher in the Helsinki Institute for Information Technology in Finland (2003-2006), as well as working for internships at Bell Labs, AT&T Labs, and Microsoft Research. His research areas are data mining, algorithms, and databases.

Heikki Mannila received his Ph.D. in Computer Science in 1985 from the University of Helsinki. After some time at the University of Tampere and various researcher positions, in 1989 he was appointed a professor of Computer Science at the University of Helsinki. He was a visiting professor in the Technical University of Vienna in 1993 and a visiting researcher at Max Planck Institute for Computer Science in Saarbruecken in 1995–1996. He moved to Microsoft Research in Redmond in 1998, came back to Finland to Nokia Research in 1999, where he stayed until the end of 2001. After that, Heikki Mannila was the research director of the basic research unit of Helsinki Institute for Information Technology in 2002–2004. Since 1999 he has been a professor of Computer Science at Helsinki University of Technology. Currently he is an academy professor (2004–2009). His research group is located partly in Helsinki University of Technology and partly in University of Helsinki. He received the ACM SIGKDD Innovation award in 2003. Heikki Mannila is the author of two books and over 150 refereed articles in Computer Science and related areas. The book Principles of Data Mining, with David Hand and Padhraic Smyth, is available also in Chinese and in Polish.