# Mining poly-regions in DNA

## Panagiotis Papapetrou*

Department of Information and Computer Science,
Aalto University 00076, Finland
E-mail: panagiotis.papapetrou@tkk.fi
*Corresponding author

## Gary Benson

Departments of Biology and Computer Science,
Boston University, MA 02215, USA
E-mail: gbenson@bu.edu

## George Kollios

Computer Science Department,
Boston University, MA 02215, USA
E-mail: gkollios@cs.bu.edu

**Abstract:** We study the problem of mining poly-regions in DNA. A poly-region is defined as a bursty DNA area, i.e., area of elevated frequency of a DNA pattern. We introduce a general formulation that covers a range of meaningful types of poly-regions and develop three efficient detection methods. The first applies recursive segmentation and is entropy-based. The second uses a set of sliding windows that summarize each sequence segment using several statistics. Finally, the third employs a technique based on majority vote. The proposed algorithms are tested on DNA sequences of four different organisms in terms of recall and runtime.

**Keywords:** poly-regions; burstiness; sliding windows; recursive segmentation; majority vote; nucleosomes.

**Reference** to this paper should be made as follows: Papapetrou, P., Benson, G. and Kollios, G. (xxxx) 'Mining poly-regions in DNA', *Int. J. Data Mining and Bioinformatics*, Vol. x, No. x, pp.xxx–xxx.

**Biographical notes:** Panagiotis Papapetrou received this BSc in Computer Science from the University of Ioannina, Greece, in 2003, and the MA and PhD Degrees in Computer Science from Boston University in 2006 and 2009, respectively. Since September 2009, he is a postdoctoral researcher with the data mining group of the Department of Information and Computer Science at Aalto University, Finland. He is Member of the ALGODAN Academy of Finland center of excellence and also member of HIIT. His research interests include data mining, knowledge discovery in databases, sequential and temporal pattern mining, human motion analysis, efficient similarity-based retrieval, and bioinformatics.

Gary Benson is an Associate Professor at Boston University in the Department of Computer Science, the Department of Biology, and the Bioinformatics graduate program. He is Director of the NSF funded BU Bioinformatics IGERT graduate training grant and Executive Editor of the annual Web Server issue of Nucleic Acids Research. He received his PhD in 1992 in Computer Science from the University of Maryland and was a Postdoctoral Fellow at the University of Southern California from 1992 to 1994. His area of expertise is pattern matching and pattern detection, especially as it applies to DNA sequence analysis and DNA repeats.

George Kollios received his Diploma in Electrical and Computer Engineering in 1995 from the National Technical University of Athens, Greece; and the MSc and PhD Degrees in Computer Science from Polytechnic University, New York in 1998 and 2000 respectively. He is currently an Associate Professor in the Computer Science Department at Boston University in Boston, Massachusetts. His research interests include spatio-temporal databases and data mining, database security, multimedia indexing, and approximation algorithms in data management. He is currently an Associate Editor for the ACM Transactions on Database Systems. He is a member of ACM and IEEE Computer Society.

## 1 Introduction

In cells, DNA forms long chains made up of four chemical units known as nucleotides: adenine (A), guanine (G), cytosine (C), and thymine (T). In these DNA chains or sequences, a number of important, known functional regions, at both large and small scales, contain a high occurrence of one or more nucleotides. We will refer to these as *poly-regions* (for example, a region that is rich in nucleotide A, will be called poly-A). Such regions include:

- *Isochores*: Multi-megabase regions of genomic sequence that are specifically GC-rich or GC-poor. GC-rich isochores exhibit greater gene density. Human ALU and L1 retrotransposons appear preferentially in isochores with composition that approaches their own.
- *CpG islands*: Regions of several hundred nucleotides that are rich in the dinucleotide CpG which is generally under-represented (relative to overall GC content) in eukaryotic genomes. The level of methylation of the cystine (C) in these dinucleotide clusters has been associated with gene expression in nearby genes.
- *Protein binding regions*: Within these domains, tens of nucleotides long, dinucleotide, or base-step composition, can contribute to DNA flexibility, allowing the helix to change physical conformation, a common property of protein-DNA interactions.

Despite the importance of poly-regions, their algorithmic identification and study has received only limited attention. To the best of our knowledge, there has been

yet no formal definition of poly-regions, in their most general form. Moreover, most methods that have been applied (or developed) so far are designed for limited types of poly-regions and target specific compositions (mainly $G + C$-rich or CpG islands).

The main contributions in this paper include:

- a formal definition of the problem of discovering poly-regions of items or patterns in a sequence
- an exact algorithm that uses a set of sliding windows over the sequence
- two approximate algorithms for detecting poly-regions: the first one is entropy-based and uses recursive segmentation techniques and the second one is based on the majority vote
- the application of an efficient arrangement mining algorithm by Papapetrou et al. (2009) to extract the complete set of frequent arrangements of these poly-regions
- an extensive experimental evaluation of our algorithms by testing their efficiency on the Dog genome
- an analysis of some standard types of poly-regions that have been detected on exons, introns, and nucleosomes in four different genomes: Dog, Chicken, Mouse, and Yeast.

## 2 Related work

Most approaches for identifying DNA regions of specific compositions use DNA segmentation techniques. One family of DNA segmentation algorithms employs statistical methods based on:

- The Maximum Likelihood Estimation (MLE) of the segments where the MLE is computed for the segments, given a restriction on their minimum length (Fu and Curnow, 1990). In the same setting, several dynamic programming approaches have been developed, such as Bement and Waterman (1977) and Auger and Lawrence (1989).
- The hidden Markov chain model, where Churchill (1989, 1992) and Perina et al. (2009) use HMM's to model the segmentation of DNA sequences and predict the locations of possible segments in mitochondrial and phage genomes, assuming that different segments can be classified into a finite number of states, i.e., *poly-A*, or $A + T$-rich.
- The Variable-Length Markov Chain Model (VLMC) by Gwadera et al. (2008) where the segment boundaries of a sequence are discovered by computing a VLMC for each segment using the Bayesian Information Criterion (BIC) and a variant of the Minimum Description Length (MDL) principle; in DNA the proposed method selects segments that closely correspond to the annotated regions of the genes.
- The walking Markov model, where Ficket et al. (1992) examined the base composition of human and $E.coli$ genomes and analyse the phenomenon of

strand symmetry, i.e., each base has the same number of occurrences on each strand, and notice the poor fit of Markov models and observe that there is less local homogeneity than necessary for most existing segmentation models,

- Bayesian models such as Ramensky et al. (2000), where the Bayesian estimator is used as a measure of homogeneity.

Another family of DNA segmentation algorithms includes those that work in a hierarchical manner (top-to-bottom) employing recursive segmentation of DNA sequences, where at each stage a split point is chosen based on a specific criterion, e.g., the Jensen-Shannon Divergence (Grosse et al., 2002; Zhang et al., 2005). Such algorithms have been proposed in Bernaola-Galvan et al. (1996, 2000), Grosse et al. (2002), Zhang et al. (2005) and Arvey et al. (2009) and their main focus was to find domains in DNA that are homogeneous in base composition or more specifically in C+G content. Moreover, in Li et al. (2002) and Li (2001), it is shown that there are many other applications of the recursive segmentation algorithm to the analysis of DNA sequences, such as detection of isochores (large homogeneous C+G domains), CpG islands (small homogeneous CG domains), etc. Another recursive segmentation approach is presented in Oliver et al. (1999), where the DNA sequence is divided into compositionally homogeneous domains by iterating a local optimisation procedure at a given statistical significance. Once the DNA sequence is partitioned into domains, a global measure of Sequence Compositional Complexity (SCC), accounting for both the sizes and compositional biases of all the domains in the sequence, is derived.

There have been studies on similar problems, called "change-point problems" that have been applied to DNA sequence segmentation Carlstein et al. (1994), Braun and Mueller (1998) and Braun et al. (2000)). The basic form of the multiple change point problem assumes that there exists a set of points in a sequence where the distribution of the sequences changes. Thus, each grouping of consecutive literals (that will form a segment) will arise from a different distribution. The methodology they follow can be broken down into first determining how many change-points exist in a sequence, and then finding their locations. Also, in Szpankowski et al. (2003), a study on change-points (transitions between homogeneous and inhomogeneous regions of DNA) is carried out, and rigorous methods of information theory are employed to quantify structural properties of DNA sequences.

In addition, DNA amplifications, i.e., mutations that increase the copy number of a specific DNA segment are frequently observed in a variety of human cancers and have been recently studied in Myllykangas et al. (2006, 2008). The goal of was to classify human cancers based on their amplification patterns and explore the biological and clinical fundamentals behind their amplification-pattern based classification. Nonetheless, in this paper we provide a more general formulation that is not limited to repeats of this type but can cover any type of highly occurring pattern along a large DNA sequence.

A sliding window approach with fixed size window has been applied on the human genome by Venter (2001) and Larsen et al. (1992) to detect $G + C$-rich regions and CpG islands. Also, in Olivera et al. (2002), a reliable segmentation method is used to partition the longest contigs in the human genome into Long Homogeneous Regions (LHGRs), thereby revealing the isochores.

Finally, a statistical approach for evaluating the significance of 'burstiness' in DNA sequences has been studied by Haiminen et al. (2008) and others. This work, however, was limited to detecting unusually bursty episodes of size 2. Assessing the statistical significance of poly-regions is a very challenging topic and of great biological interest, though it is not one within the scopes of this paper.

All the aforementioned methods have been designed and tuned for a specific problem setting, i.e., to detect poly-regions of pre-defined or fixed compositions. Though there have been large amounts of work related to finding such regions, there has yet not been any general formulation covering all types of regions where there exists a high occurrence of some pattern. In this paper, our goal is to approach the problem using a more general setting.

## 3 Problem formulation

A *sequence* $\mathcal{T} = \{t_1,\ t_2,\ \ldots,\ t_m\}$ is an ordered set of items, where $t_i \in \Sigma$, $\forall i \in [1, m]$. Since we are studying DNA sequences, $\Sigma$ corresponds to the alphabet consisting of the four nucleotide types that appear in DNA, i.e., $\Sigma = \{A,\ C,\ G,\ T\}$. A *poly-region* is a triplet $P = \{\mathcal{X},\ start,\ end\}$ that corresponds to a segment of $\mathcal{T}$, where there is a 'high occurrence' of pattern $P.\mathcal{X}$, starting at position $P.start$ and ending at position $P.end$ in $\mathcal{T}$. There are two types of patterns $P.\mathcal{X}$ that are being studied in this paper: in the first case, $P.\mathcal{X}$ corresponds to a set of items from $\Sigma$, whereas in the second case, $P.\mathcal{X}$ is a sequence of items from $\Sigma$. Based on the pattern type, two types of poly-regions are considered, formally defined as follows:

- *Poly-region of Type I*: $P = \{\mathcal{I},\ start,\ end\}$, where $\mathcal{I} \subseteq \Sigma$ is a set of items, with $t_{start} \in \mathcal{I}$ and $t_{end} \in \mathcal{I}$. $|\mathcal{I}|$ corresponds to the number of items in $\mathcal{I}$. Examples of poly-regions of Type I are: poly-{A} (known as poly-As), poly-{A,C} (known as poly-{A+C}s), etc.
- *Poly-region of Type II*: $P = \{\mathcal{S},\ start,\ end\}$, where $\mathcal{S} = s_1,\ s_2,\ \ldots,\ s_{|\mathcal{S}|}$ is a sequence of items, with each $s_i \in \Sigma$, $t_{start} = s_1$ and $t_{end} = s_{|\mathcal{S}|}$. $|\mathcal{S}|$ corresponds to the size of pattern $\mathcal{S}$, i.e., the length of the sequence. An example of a poly-region of Type II is: poly-{CG} (also known as *CpG-island*).

A poly-region where the size of pattern $P.\mathcal{X}$ is $k$, is called *$k$-poly-region*. Given a Type $I$ poly-region $P = \{\mathcal{I},\ start,\ end\}$, with $|\mathcal{I}| = k$, the frequency $f_i$ of each item $t_i \in \{t_{P.start},\ \ldots,\ t_{P.end}\}$ is the number of occurrences of $t_i$ in that region over the length of the region. Hence, the total frequency of $\mathcal{I}$ is defined as follows:

$$f_{P.\mathcal{I}} = \frac{\sum_{i=1}^{|\mathcal{I}|} f_i}{|\mathcal{I}|}. \tag{1}$$

The *density* of a type I poly-region $P$ is $d$, if $f_{\mathcal{I}} \geq d$ and $f_i \geq \frac{d}{2k},\ \forall i \in \mathcal{I}$. This means that the sum of the individual frequencies of each item should be at least $d$ and each individual frequency should be at least $\frac{d}{2k}$. For example, a poly-{A+C} of size 20 should have at least $20\frac{d}{4}$ $A$'s, at least $20\frac{d}{4}$ $C$'s, and the sum of $A$'s and $C$'s should be at least $20d$.

In the case of a Type $II$ poly-region $P = \{\mathcal{S},\ start,\ end\}$, the frequency of $\mathcal{S}$ is defined as

$$f_{P.\mathcal{S}} = \frac{|\mathcal{S}| * N_o(P,\ T)}{P.end - P.start + 1}. \tag{2}$$

where $N_o(P,\ T)$ is the number of non-overlapping occurrences of sequence $P.\mathcal{S}$ in $\{t_{P.start},\ \ldots,\ t_{P.end}\}$. Also, the *density* of a type II poly-region $\mathcal{P}$ is $d$, if $f_{P.\mathcal{S}} \geq d$.

Given a density threshold $min\_density$, a poly-region of density $d$ is said to be *dense*, if $d \geq min\_density$. In Figure 1, we can see four examples of poly-regions:

1 is a poly-A region, with $P = \{\{A\},\ 5,\ 14\}$ with density $80\%$

2 is a poly-{A,C} region, with $Q = \{\{A,\ C\},\ 20,\ 29\}$, where each one has a frequency of $40\%$

3 is a poly-ApC, with $= \{\{AC\},\ 32,\ 39\}$ and density $75\%$

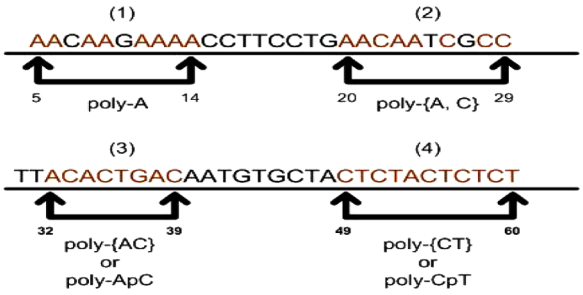
4 is a poly-CpT, with $P = \{\{CT\},\ 49,\ 60\}$ and density $91\%$.

Given two poly-regions $P$ and $Q$, both of the same type (either I or II), with $P = \{\mathcal{X},\ P.start,\ Q.start\}$, and $Q = \{\mathcal{Y},\ P.end,\ Q.end\}$, the *merging* of $P$ and $Q$ is a new poly-region $P'$, with

$$P' = \{\mathcal{X},\ min\{P.start, Q.start\},\ max\{P.end, Q.end\}\}. \tag{3}$$

Notice that merging is only allowed when $P$ and $Q$ are of the same type and $P.\mathcal{X} = Q.\mathcal{Y}$. Also, a poly-region $P = \{\mathcal{X},\ start,\ end\}$ is said to be contained in another poly-region $Q = \{\mathcal{Y},\ start,\ end\}$, if $Q.start \leq P.start$, $Q.end \geq P.end$, and $P.\mathcal{X} = Q.\mathcal{Y}$. A dense poly-region $P$ with density $d_P$ is *maximal*, if there exists no poly-region $Q$ with density $d_Q$ such that $d_Q \geq min\_density$ and $P$ is contained in $Q$.

**Figure 1** Example of two poly-regions (see online version for colours)




Finally, a poly-region can be seen as an *event interval*, which (based on Papapetrou et al. (2005, 2009)) is a triple $(e_i,\ t^i_{start},\ t^i_{end})$, where $e_i$ is an event label, $t^i_{start}$ is the start position of the event and $t^i_{end}$ is the end position in the DNA sequence. A set of event intervals, ordered by their start time, is called an *event interval sequence* or *e-sequence*. Thus, a set of poly-regions of a DNA sequence $\mathcal{T}$ corresponds to an e-sequence where each event is a poly-region and the event start and end points are

the start and end points of that poly-region. A more detailed analysis on the above terminology and concepts is given in Section 5.

Our goal is to first find the complete set of poly-regions given an input DNA sequence and then apply an efficient algorithm for mining frequent arrangements of temporal intervals to discover arrangements of poly-regions that occur frequently

1 in the sequence

2 among different segments of the sequence.

For (2), the Hybrid-DFS proposed by Papapetrou et al. (2005, 2009) applies directly, whereas for (1) an approach similar to that described in Mannila and Toivonen (1996) for mining frequent episodes over a sequence of instantaneous events can be employed.

*Problem statement*: Given a sequence $\mathcal{T} = \{t_1,\ t_2,\ \ldots,\ t_m\}$, a density constraint $d$, a minimum poly-region size $min\_poly$, a maximum poly-region size $max\_poly$ and a support threshold $min\_sup$, are goal is to:

1 Discover the complete set $\mathcal{P}_S$ of maximal poly-regions of type I and II in $\mathcal{T}$, where each region has density of at least $d$ and size $\in [min\_poly, max\_poly]$, and then

2 given $\mathcal{P}_S$, define a set of segments of $\mathcal{T}$, and based on a support threshold $min\_sup$, extract the complete set $\mathcal{F}$ of arrangements of poly-regions of type I and II that occur frequently in those segments.

## 4 Extracting poly-regions

Since we are studying DNA, $\Sigma = \{A,\ C,\ G,\ T\}$. In this setting, we are going to cover the following cases of poly-regions:

- Poly-regions $P = \{\mathcal{I},\ start,\ end\}$ of Type I, with: $|\mathcal{I}| = 1$ (giving a total of $K_1 = |\Sigma|$ poly-regions), and $2 \leq |\mathcal{I}| \leq 3$ with all items in $\mathcal{I}$ are different from each other (giving a total of $K_2 = \frac{(|\Sigma|+1)|\Sigma|}{2}$ poly-regions)
- Poly-regions $P = \{\mathcal{S},\ start,\ end\}$ of Type II, where $2 \leq |\mathcal{S}| \leq 3$. Notice that if $|\mathcal{S}| = 2$, the two nucleotides should be different from each other, and if $|\mathcal{S}| = 3$, the case where all three nucleotides are the same is excluded (giving $K_3 = |\Sigma|\ (|\Sigma| - 1) + |\Sigma|\ (|\Sigma|^2 - 1$ poly-regions).

### *4.1 Recursive segmentation*

The idea of recursive segmentation based on a measure of divergence has been used in earlier works, such as Bernaola-Galvan et al. (1996), Grosse et al. (2002) and Zhang et al. (2005). Li et al. (2002) describes how it can be applied to DNA for detecting $G + C$-rich regions and $CpG$ islands. In this section, we present a general method for poly-regions of Type I and II. Our method uses recursive segmentation

and also applies an alphabet reduction technique to efficiently handle poly-regions of both types.

Before proceeding to a detailed description of the algorithm, let us first give some basic definitions. Let $\mathcal{T} = \{t_1,\ t_2,\ \ldots\ , t_m\}$ be the input sequence and $\mathcal{T}_{l,r} = \{t_l,\ \ldots\ ,\ t_r\}$ be a segment of $\mathcal{T}$, i.e. the subsequence of $\mathcal{T}$ starting at $t_l$ and ending at $t_r$, for $1 \leq l, r \leq m$. A segmentation of $\mathcal{T}$ is denoted as $\mathcal{T}_{seg} = \{n_1,\ n_2,\ \ldots, n_{M-1}\}$, where each $n_i$ is an index of a point in $\mathcal{T}$. Trivially, $\mathcal{T}_{seg}$ defines $M$ segments, where each segment starts at point $t_{n_{j-1}}$ and ends at point $t_{n_j}$, with the first segment starting at point $t_1$ and ending at point $t_{n_1}$ and the last segment starting at point $t_{n_{M-1}}$ and ending at point $t_m$. Given a segment $\mathcal{T}_{l,r}$, $fr(\mathcal{T}_{l,r}) = \{fr_i,\ i = 1, \ldots, t\}$ denotes the set of frequencies of each distinct item in $\mathcal{T}_{l,r}$, where each item frequency is the number of occurrences of this item in that segment.

Given a sequence $\mathcal{T}$, with $t$ distinct items, $H_{\mathcal{T}} = -\sum fr_i \log_2 fr_i$, $for\ i = 1, \ldots, t$ is the Jensen-Shannon Entropy of $\mathcal{T}$, where $fr_i$ is the frequency of item $i$ in $\mathcal{T}$. Assume that $\mathcal{T}$ is split into two segments $\mathcal{T}_{1,r}$ and $\mathcal{T}_{r+1,m}$. Then, the Jensen-Shannon Divergence of the two segments is defined as

$$D(n) = H_{\mathcal{T}} - \left(\frac{r}{m} H_{\mathcal{T}_{1,r}} + \frac{m-r}{m} H_{\mathcal{T}_{r+1,m}}\right) \tag{4}$$

### 4.1.1 The algorithm in detail

Starting with the original sequence $\mathcal{T}$, the algorithm looks for that position $r \in [2, |\mathcal{T}| - 1]$ in $\mathcal{T}$ that maximises the JSD value of the two segments $\mathcal{T}_{1,r}$ and $\mathcal{T}_{r+1,m}$. The same process is applied recursively to each segment until a halting condition is satisfied. In our case, the halting condition requires that each segment should be of length between $min\_poly$ and $max\_poly$. Thus, the splitting continues while there are segments of size larger than $max\_poly$. In the case where a segment is of size less than $min\_poly$, the recursion halts without reporting that segment.

The algorithm, as described above, can efficiently detect regions of high occurrence of a single nucleotide. However, if we are interested in poly-regions of more than one nucleotides or poly-regions of Type II, the above process may fail. To achieve an efficient segmentation for both types of poly-regions, a preprocessing step is applied, which has been suggested in Li et al. (2002) for the detection of isochores. When looking for poly-regions of two nucleotides, say poly-$\{W, Y\}$, the original sequence is transformed to a new sequence as follows: each $W$ and $Y$ nucleotide is replaced by literal $X$, whereas the rest are replaced by a literal taken from $\Sigma$ (each time a different literal is chosen and when all literals of $\Sigma$ have been used, we start over). For example, if $S = ACAAAGCGA$ and we are looking for poly-regions of $A$, $S$ will be converted to $S' = XAXXXCGTX$, given that $\Sigma = \{A,\ C,\ G,\ T\}$. The same idea is followed when looking for all poly-regions of Type I. As for poly-regions of Type II, the input pattern is detected in the sequence and all the literals that are part of the pattern are changed to $X$, whereas the other ones are replaced as in the case of Type I poly-regions. The benefit of this replacement is the following: at each step of the segmentation, two regions are under consideration, say $r_1$ and $r_2$. If $r_1$ is of high occurrence of the desired pattern and in $r_2$ (which is the rest of the sub-sequence under consideration) all literals are different, the entropy difference between $r_1$ and $r_2$ will be maximised.

The steps of the recursive segmentation algorithm are given below:

- Given an input sequence $\mathcal{T}$, for each type of poly-region, $\mathcal{T}$ is converted to $\mathcal{T}'$ as described above.
- Given $\mathcal{T}'$, $JSD(\mathcal{T}_{1,r}, \mathcal{T}_{r+1,m})$ is calculated for each $r \in [2, m-1]$.
- Let $n$ be the index of $\mathcal{T}'$ where $JSD$ is maximised. $\mathcal{T}'$ is segmented, and position $n$ is reported. If the halting condition is satisfied for a segment, the segmentation process terminates for that segment, otherwise it proceeds recursively.
- When the above process is completed, a segmentation $\mathcal{T}_{seg} = \{n_1,\ n_2,\ \ldots, n_{M-1}\}$ of $M$ segments is generated. Each of these segments is a candidate poly-region. Next, a linear scan is performed on $\mathcal{T}_{seg}$. Each segment is checked whether it satisfies the density constraint and it is further expanded both ways until the density constraint is violated. Finally, when a poly-region is found it is reported.

A major weakness of such approach is that the poly-region pattern we are looking for needs to be pre-defined before we pre-process the original sequence. Then, for each additional pattern we need to start from scratch and pre-process the sequence again. Thus, we cannot get all poly-regions by one single pass, but for each specific poly-region pattern we need to perform one full recursive segmentation. Also, due to its recursive nature and as confirmed by the experimental evaluation, this approach is approximate, i.e., there may be a small fraction of false dismissals.

### *4.1.2 Complexity*

Every time the sequence is split into two subsequences. The number of splits is $O(\log(|\mathcal{T}|/(max\_poly - min\_poly)))$, where $|\mathcal{T}|$ is the size of the original sequence. Since on each recursion each segment is read once and at the final step we just perform a linear scan, the total runtime of each run of the algorithm is $O(|\mathcal{T}| \log |\mathcal{T}|)$. Now, given that the alphabet size is $\Sigma$, the number of times the algorithm is run is $K'$, the total runtime of the algorithm is $O(K'|\mathcal{T}| \log |\mathcal{T}|)$, and since $K'$ is a constant (and $K' << |\mathcal{T}|$), this becomes $O(|\mathcal{T}| \log |\mathcal{T}|)$.

## *4.2 Sliding windows*

The key idea behind this approach is to use a set of sliding windows over the input sequence. Each sliding window keeps statistics of a segment that mainly include the number of occurrences of each candidate element (meaning each item or sequence of the poly-regions we are looking for) in that segment. Combining these statistics efficiently produces the complete set of poly-regions in the sequence.

More formally, the proposed algorithm is given a sequence $\mathcal{T}$, a density factor $d$, a minimum window size $min\_poly$ and a maximum window size $max\_poly$. The first step is to define a set of sliding windows $\mathcal{W}$. Let $\mathcal{W} = \{w^1,\ w^2,\ \ldots,\ w^n\}$, where $w^i$ corresponds to sliding window $i$ and $n = |\mathcal{W}| = max\_poly - min\_poly + 1$. Each sliding window $w^i$ is a triplet $\{\mathcal{C}^i,\ w^i_{start},\ w^i_{end}\}$, where $\mathcal{C}^i$ is a set of statistics for $w^i$, $w^i_{start}$ is an index to the starting position of $w^i$ on $\mathcal{T}$ and $w^i_{end}$ is an index to the ending position of $w^i$ on $\mathcal{T}$. $\mathcal{C}^i$ is a set of $t$ counters $\{C_1,\ C_2,\ \ldots,\ C_t\}$,

with $t = K_1 + K_3$ or $t = K'_1 + K'_3$ if reverse complements are excluded. The value of each counter is the number of occurrences of the corresponding item/sequence in the window. Moreover, the piece of $\mathcal{T}$ covered by $\mathcal{W}$ is stored at each time instance. Given this setting, at any time, we can extract the top $k$ frequent items in each window.

Also, we keep a list $L$ of the poly-regions discovered so far. Each record in $L$ corresponds to a poly-region label and points to a list of all the poly-regions discovered so far with this label. Upon discovery of a new poly-region we insert it into $L$ based on its label.

### 4.2.1 The algorithm in detail

The algorithm has three phases: the Initialisation Phase, the Sliding Phase and the Merging Phase. During the first phase, $\mathcal{W}$ is initialised; this phase is completed as soon as the first $max\_poly$ characters of the sequence are read. Then the algorithm proceeds with the Sliding Phase, where $\mathcal{W}$ slides across the sequence until it reaches the end of the sequence. Before inserting each new poly-region into $L$ the Merging Phase is activated, to identify any old poly-region that can be absorbed by the new one. More details on the three Phases are given below:

- *Initialisation phase*: The first $min\_poly$ characters are read and window $w^1$ is created. This is in fact the window of the smallest size in $\mathcal{W}$. The counters of $w^1$ are updated based on what has been read so far. For each new character $s_j$, a window $w^i$, for $i = 2, \ldots, n$, of size $min\_poly + i - 1$ is created starting at character $s_1$ and ending at character $s_j$. The counters of each window $w^i$ are updated based on the counters of the previous window (i.e., $w_{i-1}$. Let $C_j^{i-1},\ for\ j = 1, \ldots, t$ denote the counters of the $(i-1)$th window. Then $C_j^i = C_j^{i-1},\ for\ j = 1, \ldots, t$. This process is repeated until $j = max\_poly$. Every time a new window is created and all the counters are updated, the window is checked for items that satisfy the density constraint. If so, it constitutes a poly-region and is added into $L$ after applying the Merging Phase. Upon completion of the current phase, $\mathcal{W}$ has been fully created. Notice that in this phase, no sliding is performed on the windows.
- *Sliding phase*: During this phase, $\mathcal{W}$ keeps sliding to the right and for every new item $s_i$, the corresponding counters are updated, i.e., for each $w^i$ in $\mathcal{W}$, $C_{s_i} = C_{s_i} + 1$. Since each window in $\mathcal{W}$ is moved one position to the right, the counter of the element that is no longer in the window has to be decreased by one, i.e., for each $w^i$ in $\mathcal{W}$, $C_{S_{start}} = C_{S_{start}} - 1$. Finally, the start and end pointers of each window are updated accordingly. After a slide is performed and all counters are updated, each window is checked for having any itemset or sequence satisfying the density threshold. Starting with the window of maximum size, if element $c$ is found to satisfy the density threshold, then this window is reported as a poly-region of $c$. Since we are only looking for maximal windows, the counter of $c$ is not checked any more in the rest of the windows in the current instance of $\mathcal{W}$. Finally, each poly-region is added into $L$ after applying the Merging Phase.
- *Merging phase*: For each new window $w^j$, before it is inserted into $L$, the corresponding record of $L$ is scanned for a window $w^i$ such that the start

points of $w^i$ and $w^j$ coincide and $w^i$ is contained in $w^j$. Trivially, if such window exists, it will be one of the last $max\_poly - min\_poly + 1$ inserted in that record. Before the insertion of $w^j$ in $L$, $w^i$ is removed. Also, since the windows inserted into $L$ are ordered by their start time, if a window is reached, with start point smaller than that of $w^j$, then the process stops and inserts $w^j$ in $L$.

Notice that at each step we do not need to check all the windows. Instead we can start with the window of maximal size and prune some of the smaller windows. More specifically, the value of each counter in a large window is an upper bound for the value of the corresponding counters in the smaller windows in $\mathcal{W}$. Let the number of elements of type $c$ (either itemsets or sequences) in $w^i$ be $N_c^i$. Then $c$ is dense in $w^i$, if $\frac{N_c^i}{|w^i|} \geq d$. Hence, the maximum size of the window were these elements (of type $c$) can fit and fulfill the density constraint is $\frac{N_c^i}{d}$. Based on this observation, we can start with the maximum window and then apply the bound on each counter. This indicates which windows of the lower levels should be searched for a candidate poly-region for each item. Consider Figure 1(2) for example, and let $d = 50\%$. Suppose that $max\_poly = 10$, and currently the maximum window in $\mathcal{W}$ is the DNA sequence segment shown in the Figure and notice that $C_c = 4$. Then the maximum window in $\mathcal{W}$, where item $C$ can be dense, is of size $\frac{C_c}{d} = 8$. Thus, in order to look for a poly-region of nucleotide $C$, we should skip $w^9$. The described method produces a set $\mathcal{P}_S$ of poly-regions for the input sequence $\mathcal{T}$.

### *4.2.2 Complexity*

Based on the previous analysis, it can be seen that at any time instance, the number of windows under consideration is $M = max\_poly - min\_poly + 1$. Moreover, for each window we keep $t$ counters, which yields a total of $tM$ counters. Also, for each set of windows $\mathcal{W}$ we store the piece of the sequence that is covered by the maximum window. Thus, the space complexity is $O(|\Sigma|M + max\_poly)$. Each element is read once and then stored in $\mathcal{W}$. At each slide, in the worst case $M$ windows are accessed. For each window, the value of $t$ counters is checked and the last element of each window is removed. Therefore, for each slide a total of $Mt$ counters are accessed. Also, when a window is determined to constitute a poly-region, at most $M$ records are accessed in the list $L$ to check whether it overlaps with an existing poly-regions. The above analysis yields a time complexity of $O(|\mathcal{T}|M)$. Since in practice $max\_poly,\ min\_poly\ << |S|$, the algorithm is linear.

## *4.3 Majority vote*

Another efficient approach is described in this section that employs the idea of the majority vote, first used in Misra and Gries (1982) for finding repeated items in a sequence. The same concept was later used in Golab et al. (2003) for finding frequent items over sliding windows. Our goal is to improve the performance of the sliding window algorithm by having only a single sliding

window $w$ along with:

- a set of *primary counters* $\mathcal{C}_p$
- a set of *secondary counters* $\mathcal{C}_s$.

The primary counters are used to indicate regions that are candidate poly-regions. If a candidate poly-region is detected, then the set of secondary counters is examined to check if it actually is a poly-region.

In particular, the algorithm uses $t_1 = 3 + K_2$ primary counters and $t_2 = K_1 + K_3$ secondary counters, along with a set of buffers holding the literal corresponding to each primary counter. All counters are initially set to zero, the first literal of the input sequence is read and stored under the right buffer, and the corresponding primary counter is increased by one. Each time a new literal is read, the sequence index is increased by one. If the new literal matches one stored under a buffer, then the corresponding primary counter is increased by one, otherwise it is decreased by one; if a primary counter reaches zero, the literal currently in its buffer is replaced by the new one. In any other case, we move on to the next literal in the sequence. When an element (either itemset or sequence) is identified in the sequence the corresponding secondary counters are updated so that, at any time during the sequence scan, each secondary counter is equal to the number of occurrences of the corresponding element in the window. This process continues until the whole sequence is read. In the case of Type I poly-regions of a set of literals, all literals in the set are considered to be the same during the scan. As for poly-regions of Type II, they can be seen as a single literal. For example, consider the sequence $ACACCAC$. In this case, the first literal is $AC$, the next is $CA$, the next $ACA$, then $CA$ and so on. This explains the need for more than one primary counters: 1 counter for the single items, $K_2$ counters for the itemsets, 1 counter for poly-regions of Type II with $|\mathcal{X}| = 2$ and one for poly-regions of Type II with $|\mathcal{X}| = 3$, yielding a total of $t_1$ primary counters.

The benefit of this approach is that the behaviour of the primary counters can imply high occurrence of a set of items or subsequence in a specific region of the sequence. In fact, we have two cases:

- if a primary counter increases rapidly, then there is high occurrence of the corresponding literal stored in the buffer implying the existence of a poly-region
- if a primary counter decreases rapidly, then the corresponding literal in the buffer does not occur frequently in that region.

Instead another literal might be in majority in the region, which will constitute a poly-region. However, this might not be the case since decrease on a primary counter only implies that the corresponding literal in the buffer is not in majority in that area and does not necessarily imply majority of another literal.

Let $w$ be the sliding window, and $c_i^S$ be the value of counter $i$ at the beginning of $w$, and $c_i^E$ be the value of the counter at the end of $w$. The following lemmas hold, based on the previous analysis for each primary counter.

**Lemma 4.1:** *If* $\exists\ C_i \in \mathcal{C}_p$ *such that* $\Delta C_i > 0$ *and* $\Delta C_i \geq |w|(2d-1)$*, where* $d$ *is the density constraint, then* $w$ *corresponds to a poly-region.*

*Proof:* Since $w$ corresponds to a poly-region $\mathcal{P}$ of say element $c$, the number of occurrences of $c$ in $P$ is $N_c \geq |w|d$. The counter will be increased by one at least $|w|d$ times, and it will be decreased by one at most $|w|(1-d)$ times. Thus, the total change of the counter in a poly-region can be at least $|w|(2d-1)$.

**Lemma 4.2:** *If* $\exists\ C_i \in \mathcal{C}_p$ *such that* $\Delta C_i < 0$ *and* $|\Delta C_i| \geq |w|(2d-1)$*, then* $w$ *is a candidate poly-region.*

*Proof:* Lemma 4.2 is proved by an argument similar to that for Lemma 4.1. However, in this case, since the fact that the counter decreases does not necessarily mean that the same literal appears consecutively. Thus, $w$ corresponds to a candidate poly-region.

**Lemma 4.3:** *If for all counters* $C_i \in \mathcal{C}_p$, $|\Delta C_i| < |w|(2d-1)$*, then* $w$ *cannot be a poly-region.*

*Proof:* Straightforward, from the above Lemmas.

The algorithm applies the above lemmas each time $w$ slides to the right. Every time a poly-region is discovered by Lemma 1, it is added into the set of poly-regions. When a candidate poly-region is discovered by Lemma 4.2, the set of secondary counters in $w$ are invoked to check whether it actually corresponds to a poly-region and if not it is discarded.

### *4.3.1 The algorithm in detail*

Let $w$ be the sliding window, and $c_i^S$ be the value of counter $i$ at the beginning of $w$, and $c_i^E$ be the value of the counter at the end of $w$. Also, let the change on a counter be $\Delta C_i = c_i^E - c_i^S$. The main steps of the algorithm are the following:

- If for all primary counters in $\mathcal{C}_p$, $\Delta C_i < |w|(2d-1)$, slide to the right.
- If $\exists\ C_i \in \mathcal{C}_p$ such that $\Delta C_i > 0$ and $\Delta C_i < |w|(2d-1)$, then $w$ is reported as a poly-region.
- If $\exists\ C_i \in \mathcal{C}_p$ such that $\Delta C_i < 0$ and $\Delta C_i < |w|(2d-1)$, then $w$ is a candidate poly-region. Each of the secondary counters is checked. If for a set of secondary counters $C' \subseteq \mathcal{C}_s$, $\Delta C^j = C_j^E - C_j^S \geq |w|d\ (\forall j \in \mathcal{C}')$, then $w$ is reported as a poly-region of $\mathcal{C}'$.
- Steps 1–3 are repeated until the whole sequence is scanned.

Finally, we get a set of poly-regions of size $|w|$. However, according to the problem formulation, the poly-regions should be of size $min\_poly$ to $max\_poly$. To capture all these regions, we set $|w| = \frac{min\_win}{2}$ and when a poly-region is detected, it is expanded as much as possible in order to detect all the maximal legal poly-regions in the range of $[min\_win,\ max\_poly]$ in that area, keeping in mind that a valid poly-region should start and end with specific literals (a poly-A should start and end with an A). This step is the most costly one of this method. Notice that once a poly-region of size $|w|$ is discovered, the expansion should make sure that

the final poly-region will:

- include $w$
- be of size $\in [min\_win,\ max\_poly]$
- start and end with the appropriate literals.

To satisfy the third condition efficiently, an index of each literal in $\Sigma$ is built at the beginning of the algorithm, such that for each literal in $\Sigma$ we get to know the positions where it occurs in the sequence. This requires a single scan, and the indices are stored in $|\Sigma|$ arrays, one for each literal. Also, for each array we keep a pair of pointers that move according to $w$: while $w$ scans the sequence the pointers slide over the indices so that they include the positions where each literal occurs in that part of the sequence currently under $w$. To satisfy the second condition we need to expand $w$ both ways. Since $|w| = \frac{min\_win}{2}$, we need to check $r = max\_poly - |w|$ positions to the right and to the left. Since $w$ has to be fully contained in the larger poly-region a maximum of $r^2$ checks is needed: we have $r$ candidate positions on the left and $r$ positions on the right and we check their combinations. Notice that since the maximum poly-region size is bounded by $\max\_poly$, we can skip some of the above checks, i.e., we first check the poly-region that starts at point $w_{start} - r$ and ends at $w_{end}$, then the poly-region starting at point $w_{start} - r + 1$ and ending at $w_{end}$, and so on. If one of those windows is a valid poly-region we check for any possible merging with any other region found in this step and then report the new poly-region.

### 4.3.2 Complexity

In terms of space complexity, the algorithm is efficient, since it only needs to keep two pointers (one to the start and one to the end point of the window $w$), a total of $t_1 + t_2$ counters, a set of $t_1 + t_2$ buffers, and $|S|$ index values. Regarding time complexity, one sequence scan is needed to make the indices, and for each small poly-region (of size $|w|$) we need to check at most $O(r^2)$ expansions. This gives a total cost of $O(|r^2||S|)$. Notice that $r = max\_poly - |w|$ and since in practice $max\_poly,\ min\_poly << |S|$, the algorithm is linear.

## 5 Discovering frequent arrangements of poly-regions

In this section we show have we can extract frequent arrangements of poly-regions in a DNA sequence. We use an existing approach for mining frequent arrangement of temporal intervals Papapetrou et al. (2005, 2009). Assuming that the set of poly-regions in a given DNA sequence $\mathcal{T}$ have been discovered using one of the three algorithms described above. Now, each of these poly-regions can be seen as an interval-based event, i.e., an event that has a time duration. Thus, the discovered set of poly-regions can be mapped to an interval-based sequence (i.e., e-sequence). An arrangement is a set of events that are temporally correlated. For more details, the reader can refer to Papapetrou et al. (2005) and Papapetrou et al. (2009).

The algorithm uses a sliding window $w$ of size $win$ to scan the whole e-sequence. $w$ is initially placed at the beginning of the e-sequence and includes the first

$win$ event intervals (in our case poly-regions) of $\mathcal{T}$. The window keeps sliding to the right (one event interval per slide) until it reaches the end of $\mathcal{T}$, i.e., its right end includes the last event interval of $\mathcal{T}$, for the first time. Based on this formulation, a total of $\mathcal{W} = |S| + win - 1$ overlapping windows is defined over the sequence. The frequency of an arrangement $\mathcal{A}$ is defined as the fraction of windows in which $\mathcal{A}$ occurs. Thus, given $\mathcal{A}$ and a window of size $win$, the frequency of $\mathcal{A}$ is: $freq(\mathcal{A}, win) = \frac{|\{w|\mathcal{A} \ occurs \ in \ w\}|}{|\mathcal{W}|}$. Notice that we could also apply spatial (i.e., place limits on the distance between two poly-regions) and structural constraints (i.e., apply regular expression constraints to the extracted patterns) during the mining process so as to focus on certain types of patterns.

## 6 Experimental evaluation

In our experiments we analyse the performance of the proposed algorithms in terms of recall and runtime. We further investigate the types of poly-regions that occur in different DNA locations (introns, exons, and nucleosomes), and detect frequent arrangements of poly-regions in these different region types. All experiments were performed on a 2.8Ghz Intel Pentium 4 dual-processor machine with 64GB of main memory, running Linux with kernel 2.4.20. The algorithms have been implemented in C++ and their runtime has been measured with the output turned off.

### *6.1 Data sets*

We study four different genomes: Dog, Yeast, Chicken, and Mouse. DNA data has been obtained from NCBI. [1] For our experiments we have investigated introns and exons within the 39 chromosomes (including the $X$ chromosome) of the dog genome ($Canis \ familiaris$). Additional DNA data has been obtained from $http://genie.weizmann.ac.il/pubs/nucleosomes06$ that includes DNA sequences around all nucleosome regions of: the Yeast in vivo (119 nucleosomes) and in vitro (204 nucleosomes), the Chicken in vivo (177 nucleosomes) and the Mouse in vitro (87 nucleosomes) genomes with an explicit annotation of the nucleosome positions on the chromosomes.

### *6.2 Performance analysis*

The three proposed algorithms have been compared in terms of runtime and recall considering the following factors:

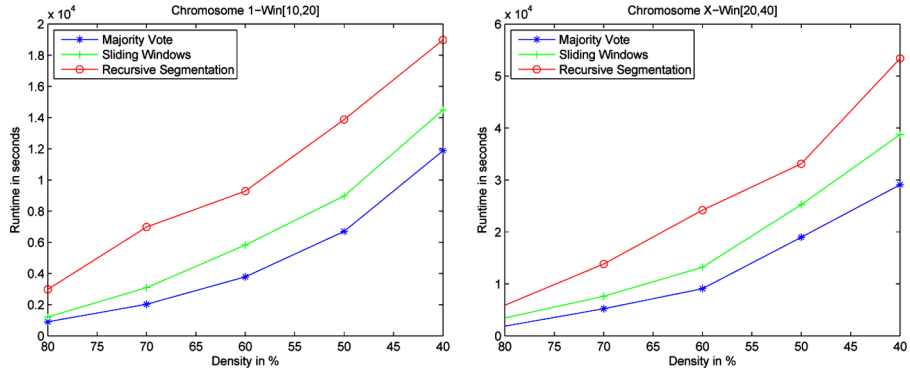
- size of the input sequence
- density of the poly-regions
- size of the minimum and maximum windows.

Recall corresponds to the percentage of poly-regions that could be retrieved by each algorithm.

Regarding runtime, the basic observation is that the third algorithm (majority vote-based) outperforms the rest. The sliding window approach is quite fast,

outperforming the recursive segmentation approach. In Figure 2, we show the performance of each algorithm with respect to the density constraint, which varies from 40% to 80%, for Chromosomes 1 (approximately 127 million bases) and $X$ of the $Canis\ Familiaris$ respectively. For Chromosome 1, the window range is $[10, 20]$, whereas for Chromosome $X$, the window range is $[20, 40]$ for the Sliding Window method and $|w| = 40$ for the Majority Vote. Notice that the runtime of both (Sliding Window and Majority Vote) is affected by the selection of the window size; however since $|w| << |S|$ (as also discussed in the corresponding complexity analysis sections) this affect is negligible.

**Figure 2** Runtime comparison of the three algorithms for Chromosomes 1 (left figure) and $X$(right figure), and the window range is $[20, 40]$ for the Sliding Window approach and $|w| = 40$ for the majority vote (see online version for colours)




Regarding recall, the sliding window approach achieved to find the complete set of poly-regions. The recursive segmentation was proved to be less accurate managing to find almost 80% (on average) of the total poly-regions. This was totaly expected for both cases: the nature of the recursive segmentation is such that split points might be chosen inside some poly-regions. This can happen mainly at the first segmentations where the segments are relatively huge. As a result, these poly-regions are not going to be included in the final segmentation. In the case of the majority vote, the chosen window size might skip some poly-regions, due to its size and depending on the value of the density constraint. For example, let $S = \ldots AACAA \ldots$, $d = 80\%$, $w = 3$ and $max\_win = 6$; due to the value of $d$, a poly-region will be reported only when all three literals in $w$ are the same. Thus, the poly-region of literal $A$ of size 5 shown in $S$ will be skipped. The experimental evaluation however, showed that if the size of $w$ is chosen to be $min\_win/2$, the percentage of false negatives will be less than 11%. Table 1 presents some results regarding the recall of the algorithms showing that the majority vote method performs significantly better than the recursive segmentation.

### *6.3 Our findings*

In this section, we study the types of poly-regions identified in DNA. We consider three different types of DNA regions: exons (coding regions), introns (non-coding regions), and nucleosomes.

**Table 1** Recall of the three algorithms for chromosomes 1, 38 and $X$ of the *Canis Familiaris* the window size for the Majority Vote is the maximum value in the range used for the sliding window approach

| *Chrom.* | *Poly-region size* | *Sliding windows* | *Recursive segm.* | *Majority vote* |
|---|---|---|---|---|
| 1 | [10, 20] | 49325 (100%) | 38223 (77%) | 45582 (92%) |
| 1 | [18, 64] | 26332 (100%) | 23245 (88%) | 24765 (94%) |
| 38 | [10, 20] | 11285 (100%) | 8195 (85%) | 9980 (91%) |
| 38 | [18, 64] | 8221 (100%) | 6948 (72%) | 7988 (97%) |
| X | [10, 20] | 1793112 (100%) | 1291762 (72%) | 1615344 (93%) |
| X | [18, 64] | 696261 (100%) | 598455 (85%) | 626762 (91%) |

### *6.3.1 Poly-regions in exons and introns*

Table 2 shows a summary of poly-regions detected in exons of Chromosome 1 of the Dog genome. A much larger number of poly-regions have been discovered in non-coding regions, but these regions are not biologically very interesting as opposed to exons and thus we are not including our findings in this paper. Similar results have been obtained for the rest of the chromosomes, but due to space limitations they are omitted. The minimum density constraint was set to $80\%$ and the poly-region size varied between $10$ and $60$ nucleotide bases. We examined a total of $360457$ exons with an average size between $147$ and $186$ nucleotides, and $194373$ introns with an average size between $5096$ and $27521$ nucleotides. The main observation is that introns show a significantly larger accumulation of poly-regions than the exons, especially poly-As, poly-Cs, poly-Ts, poly-CTs and poly-TGs. On the other hand, exons have a high concentration of poly-As, poly-Ts and poly-TGs. Among all poly-regions of Type $II$ with $\mathcal{S} = 3$, only poly-AATs, poly-ATTs, poly-TATs and poly-ATAs show a significant occurrence in exons whereas in introns we can also have poly-CCTs, poly-CTTS poly-GAAs and poly-GTTs.

**Table 2** Types of poly-regions with density $\geq 80\%$ in exons of chromosome 1 of the Dog genome

| *Poly-region type* | *Percentile over all regions* (%) | *Percentile among exons* (%) |
|---|---|---|
| A | 0.42 | 21.29 |
| T | 0.45 | 23.74 |
| TG | 2.25 | 56.69 |
| A+C | 2.96 | 75.19 |
| A+G | 4.38 | 79.29 |
| A+T | 10.11 | 83.65 |
| C+T | 4.40 | 77.87 |
| G+T | 2.82 | 74.46 |
| A+C+G | 16.64 | 94.38 |
| A+C+T | 27.51 | 96.70 |
| A+G+T | 25.35 | 96.90 |

### 6.3.2 Poly-regions in nucleosomes

Our algorithms have also been applied to nucleosome regions of the Yeast (in vivo and in vitro), Chicken (in vivo) and Mouse (in vitro) genomes. The extracted poly-regions for the Yeast in vivo are shown in Table 3. The main observation is that nucleosome regions show a larger accumulation of poly-regions than exons; especially poly-regions of Type $I$ with $|\mathcal{I}| \geq 2$, are present in almost every nucleosome. We also noticed a high occurrence of poly-CAs and poly-TGAs in the Mouse in vitro genome, which is not true for the other genomes we examined. Similar observations have been obtained for the rest of the genomes.

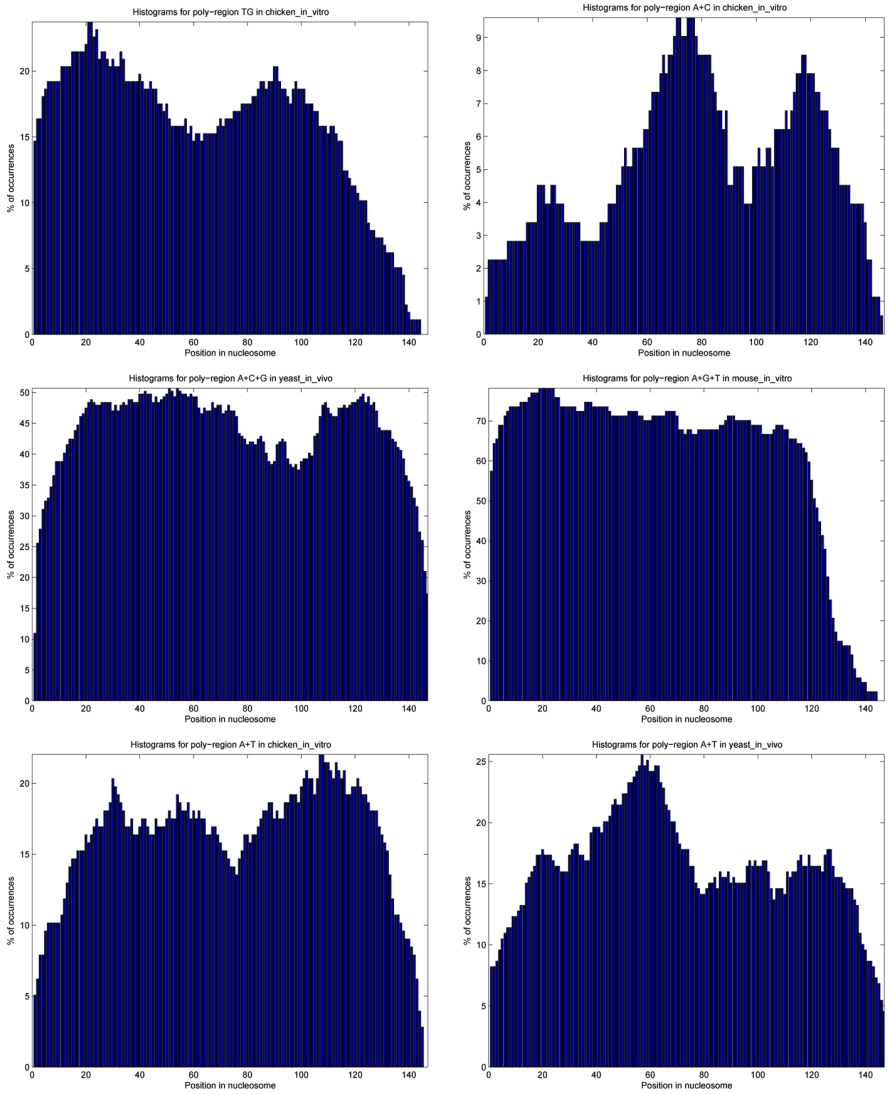
**Table 3** Types of poly-regions with density $\geq 80\%$ in the Yeast in vivo

| *Poly-region Type* | *Percentile over all regions* (%) | *Percentile among nucleosomes* (%) |
|---|---|---|
| A | 0.49 | 23.62 |
| TG | 1.79 | 56.78 |
| A+C | 2.85 | 88.94 |
| A+G | 3.92 | 91.46 |
| A+T | 7.75 | 97.49 |
| C+T | 3.20 | 87.94 |
| G+T | 2.69 | 82.41 |
| A+C+G | 14.38 | 99.50 |
| A+C+T | 31.17 | 100.00 |
| A+G+T | 30.64 | 100.00 |

In Figure 3, we see the histograms for each different frequent poly-region type for the $4$ organisms of the nucleosome data set. The $x$-axis corresponds to the actual position on the nucleosome and the $y$-axis represents the percentage of nucleosomes where this poly-region occurs in that specific position. We observed that in the majority of nucleosome regions,

- There is a high occurrence of poly-regions of Type I and size $2$ with frequencies of approximately $20\%$. As for poly-regions of size $3$, their positioning is random, which is expected due to the fact that the alphabet size is only $4$.
- In some cases there is a sharp drop off towards the end of the nucleosomes. This is true especially for poly-regions of Type I and sizes $2$ and $3$: poly-$A+C+G$, poly-$A+G+T$ and poly-$TG$ in Mouse in vitro, poly-$A+T$ in the chicken in vitro.
- Some signs of periodicity are detected in a few histograms, for example in the poly-$C+G$ in the Yeast in vivo and poly-$A+C$ in the Chicken in vitro.
- The only *poly-di-nucleotide region* (poly-region of Type II and size $2$) that appears in nucleosomes is $TG$. In the Chicken in vivo genome, there is a high concentration of $TG$s at the beginning of the nucleosomes.

**Figure 3** Histograms of frequent poly-regions in the nucleosome areas of 4 organisms. The $x$-axis corresponds to the actual nucleosome position and the $y$-axis represents the percentage of nucleosomes where this poly-region occurs in that position (see online version for colours)




### *6.3.3 Extracting temporal arrangements*

Finally, an efficient mining algorithm has been applied to the extracted poly-regions, as described in Section 4, to detect frequent temporal relations between them. Specifically, the algorithm has been applied to the poly-regions of Chromosomes 1, 2 and $X$ of the Canis Familiaris. An interesting number of frequent patterns has been extracted. In all three cases we detected a great number of overlaps and contains between poly-As and poly-Ts (in exons) as well as poly-Cs and poly-Gs (in introns). Figure 4 gives a sample of the frequent arrangements that have been

extracted from the exons of Chromosomes 1, 2 and $X$ of the Dog Genome. The most significant observation is the arrangement involving a follow relation of poly-TAs and poly-$\{C + G\}$s. Similar but fewer arrangements have been discovered in the nucleosomes (of the Yeast, Mouse and Chicken genomes). A sample of the highest scoring arrangements is shown in Figure 5.

**Figure 4** A sample of the extracted set of frequent arrangements of poly-regions in introns of Chromosomes 1, 2 and $X$ of the dog. The poly-region size varied between 10 and 40 nucleotides (see online version for colours)

| Chromosome | Arrangement | Support in % | Arrangement | Support in % |
|---|---|---|---|---|
| Chromosome 1 | A TG | 35% | A T | 66% |
| | A + C C | 21% | C A C | 19% |
| Chromosome 2 | A T | 25% | T G | 33% |
| | G T | 42% | C + G T A | 26% |
| Chromosome X | A + T C+G | 33% | C T A | 35% |

**Figure 5** A sample of the extracted set of frequent arrangements in nucleosome regions of the Yeast, Mouse and Chicken genomes. The poly-region size varied between 10 and 40 nucleotides (see online version for colours)

| Arrangement | Support in % | Arrangement | Support in % |
|---|---|---|---|
| T G | 36% | A+T A+G | 7% |
| T+G A | 19% | TG G+C | 11% |
| C+G A | 14% | T T | 21% |
| T G | 31% | TG TG | 24% |

## 7 Conclusion

We have formally defined the problem of detecting regions of elevated occurrence of a literal or set of literals in a sequence and proposed three efficient algorithms to solve it. The first algorithm employs a set of sliding windows over the input sequence that maintains some statistics per segment and combines them to extract the

complete set of poly-regions. The second algorithm is based on the idea of recursive segmentation, whereas the third achieves linear running time by employing a majority vote-based approach with a minimal number of false negatives. We further applied an efficient arrangement mining algorithm to extract the complete set of frequent temporal arrangements of the extracted regions and provided an extensive experimental evaluation of our algorithms by testing their efficiency on four different genomic regions of organisms. In our experiments, we extensively study the types of poly-regions and arrangements that occur frequently in different DNA regions (coding, non-coding, and nucleosomes). An interesting direction for future research would be to assess the significance of bursty sequences by defining meaningful null models and test statistics (based on, e.g., Haiminen et al., 2008).

## Acknowledgements

Panagiotis Papapetrou was partially supported by the Academy of Finland through the Algorithmic Data Analysis (Algodan) Centre of Excellence. Gary Benson was partially supported by the National Science Foundation through grant IIS-1017621. George Kollios was partially supported by the National Science Foundation through grant IIS-0812309.

## References

Arvey, A.J., Azad, R.K., Raval, A. and Lawrence, J.G. (2009) 'Detection of genomic islands via segmental genome heterogeneity', *Nucleic Acids Res.* Vol. 37, No. 16, pp.5255–5266.

Auger, I.E. and Lawrence, C.E. (1989) 'Algorithms for the optimal identification of segment neighborhoods', *Bulletin of Mathematical Biology*, Vol. 51, pp.39–54.

Bement, T.R. and Waterman, M.S. (1977) 'Locating maximum variance segments in sequential data', *Mathematical Geology*, Vol. 9, pp.55–61.

Bernaola-Galvan, P., Grosse, I., Carpena, P., Oliver, J.L., Roman-Roldan, R. and Stanley, H.E. (2000) 'Finding borders between coding and noncoding DNA regions by an entropic segmentation method', *Physical Review Letters*, Vol. 85, No. 6, pp.1342–45.

Bernaola-Galvan, P., Roman-Roldan, R. and Oliver, J.L. (1996) 'Compositional segmentation and long-range fractal correlations in DNA sequences', *Physical Review E*, Vol. 53, pp.5181–5189.

Braun, J.V., Braun, R.K. and Mueller, H.G. (2000) 'Multiple change-point fitting via quasi-likelihood, with application to DNA sequence segmentation', *Biometrica*, Vol. 87, pp.301–314.

Braun, J.V. and Mueller, H.G. (1998) 'Statistical methods for DNA segmentation', *Statistical Science*, Vol. 13, pp.142–162.

Carlstein, E., Mueller, H.G. and Siegmund, D. (1994) 'Change-point problems', *Lecture Notes and Monograph Series*, Vol. 23, No. 2.

Churchill, G.A. (1989) 'Stochastic models for heterogeneous DNA sequences', *Bulletin of Mathematical Biology*, Vol. 51, No. 1, pp.79–94.

Churchill, G.A. (1992) 'Hidden markov chains and the analysis of genome structure', *Computes and Chemistry*, Vol. 16, No. 2, pp.107–115.

Ficket, J.W., Torney, D.C. and Wolf, D.R. (1992) 'Base compositional structure of genomes', *Genomics*, Vol. 13, pp.1056–1064.

Fu, Y-X. and Curnow, R.N. (1990) 'Maximum likelihood estimation of multiple change points', *Biometrica*, Vol. 77, pp.563–573.

Golab, L., DeHaan, D., Demaine, E.D., Lopez-Ortiz, A. and Munro, J.I. (2003) 'Identifying frequent items in sliding windows over on-line packet streams', *IMC '03: Proceedings of the ACM SIGCOMM Conference on Internet Measurement*, pp.173–178.

Grosse, I., Galvan, P.V., Carpena, P., Roldan, R.R., Oliver, J. and Stanley, H.E. (2002) 'Analysis of symbolic sequences using the Jensen-Shannon divergence', *Physical Review E*, Vol. 65, p.041905.

Gwadera, R., Gionis, A. and Mannila, H. (2008) 'Optimal segmentation using tree models', *Knowledge and Information Systems*, Vol. 15, pp.259–283.

Haiminen, N., Mannila, H. and Terzi, E. (2008) 'Determining significance of pairwise co-occurrences of events in bursty sequences', *BMC Bioinformatics*, Vol. 9, pp.336.

Larsen, F., Gundersen, G., Lopez, R. and Prydz, H. (1992) 'CpG islands as gene markers in the human genome', *Genomics*, Vol. 13, pp.1095–1107.

Li, W., Bernaola-Galvan, P., Fatameh, H. and Grosse, I. (2002) 'Applications of recursive segmentation to the analysis of DNA sequences', *Computes and Chemistry*, Vol. 26, No. 2, pp.491–510.

Li, W. (2001) 'New stopping criteria for segmenting DNA sequences', *Phys. Rev. Letters*, Vol. 86, pp.5815–5818.

Mannila, H. and Toivonen, H. (1996) 'Discovering generalized episodes using minimal occurences', *Proc. of ACM SIGKDD*, Portland, Oregon, pp.146–151.

Misra, J. and Gries, D. (1982) 'Finding repeated elements', *Sci. Comput. Program.*, Vol. 2, No. 2, pp.143–152.

Myllykangas, S., Himberg, J., Bohling, T., Nagy, B., Hollmen, J. and Knuutila, S. (2006) 'DNA copy number amplification profiling of human neoplasms', *ONCOGENE*, Vol. 25, No. 55, pp.7324–7332.

Myllykangas, S., Tikka, J., Bohling, T., Knuutila, S. and Hollmen, J. (2008) 'Classification of human cancers based on DNA copy number amplification modeling', *BMC Medical Genomics*, Vol. 1, No. 1, p.15.

Olivera, J.L., P-Carpena, Roman-Roldanc, R., Mata-Balaguera, T., Mejyas-Romeroa, A., Hackenberga, M. and Bernaola-Galvan, P. (2002) 'Isochore chromosome maps of the human genome', *Gene*, Vol. 300, pp.117–127.

Oliver, J., Romn-Roldn, R., Prez, J. and Bernaola-Galvn, P. (1999) 'Segment: identifying compositional domains in DNA sequences', *Bioinformatics*, Vol. 15, pp.974–979.

Papapetrou, P., Kollios, G., Sclaroff, S. and Gunopulos, D. (2005) 'Discovering frequent arrangements of temporal intervals', *Proc. of IEEE ICDM*, Houston,Texas, pp.354–361.

Papapetrou, P., Kollios, G., Sclaroff, S. and Gunopulos, D. (2009) 'Mining frequent arrangements of temporal intervals', *Knowledge and Information Systems*, Vol. 21, pp.133–171.

Perina, A., Cristani, M., Xumerle, L., Murino, V., Pignatti, P.F. and Malerba, G. (2009) 'Fully non-homogeneous hidden markov model double net: a generative model for haplotype reconstruction and block discovery', *Artif. Intell. Med.*, Vol. 45, Nos. 2–3, pp.135–150.

Ramensky, V.E., Markeev, V., Roytberg, M. and Tumanyan, V. (2000) 'DNA segmentation through the Bayesian approach', Vol. 7, pp.215–231.

Szpankowski, W., Ren, W. and Szpankowski, L. (2003) 'An optimal DNA segmentation based on the MDL principle', *IEEE Computer Society Bioinformatics Conference (CSB)*, Stanford, California, pp.541–557.

Venter, J.C. (2001) 'The sequence of the human genome', *Science*, Vol. 291, pp.1304–1351.

Zhang, C-T., Gao, F. and Zhang, R. (2005) 'Segmentation algorithm for DNA sequences', *Physical Review E*, Vol. 72, p.041917.

## Note

[1]http://www.ncbi.nlm.nih.gov