

Discovering Frequent Poly-Regions in DNA Sequences

Panagiotis Papapetrou[†] Gary Benson^{††} George Kollios[†]

[†]Department of Computer Science, ^{††}Departments of Biology and Computer Science
Boston University

panagpap@cs.bu.edu, gbenson@bu.edu, gkollios@cs.bu.edu

Abstract

The problem of discovering arrangements of regions of high occurrence of one or more items of a given alphabet in a sequence, is studied, and two efficient algorithms are proposed. The first one is entropy-based and uses an existing recursive segmentation technique to split the input sequence into a set of homogeneous segments. The key idea of the second approach is to use a set of sliding windows over the sequence. Each sliding window keeps a set of statistics of a sequence segment that mainly includes the number of occurrences of each item in that segment. Combining these statistics efficiently yields the complete set of regions of high occurrence of the items of the given alphabet. After identifying these regions, the sequence is converted to a sequence of labeled intervals (each one corresponding to a region). An efficient algorithm for mining frequent arrangements of event intervals is applied to the converted sequence to discover frequently occurring arrangements of these regions. The proposed algorithms are tested on various DNA sequences producing results with significant biological meaning.

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, is called poly-A). Such regions include:

- Isochores (multi-megabase regions of genomic sequence) which are specifically GC-rich or GC-poor. GC-rich isochores exhibit greater gene density.
- CpG islands (regions of several hundred nucleotides rich in the dinucleotide CpG). The level of methylation of the cystine (C) in these dinucleotide clusters has been associated with gene expression in nearby genes [12, 11, 13].
- Protein binding regions (tens of nucleotides long), where dinucleotide, or base-step composition, can contribute to

DNA flexibility, allowing the helix to change physical conformation, a common property of protein-DNA interactions [24, 19, 14, 18].

Despite the importance of "poly" regions, their algorithmic identification and study has received only limited attention. One family of segmentation algorithms employ statistical methods based on: (1) the Maximum Likelihood Estimation (MLE) of the segments, which is computed for the segments, given a restriction on their minimum length [12]. A dynamic programming approach has been introduced in [3] that computes the global maximum, whereas [2] proposed an extension where there is no restriction on the segment size, (2) the hidden Markov chain model, proposed in [8, 9] to model the segmentation of DNA sequences and predict the locations of possible segments in mitochondrial and phage genomes, (3) the walking Markov model, which is a continuously varying stochastic process. Also, [10] examined the base composition of human and *E.coli* genomes and analyzed the phenomenon of strand symmetry.

Simultaneously, there have been studies on similar problems, called "change-point problems", that were applied to DNA sequence segmentation [7, 6, 5]. 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. They first determine how many change-points exist in a sequence and then find their locations.

Another family of DNA segmentation algorithms includes those that work in a hierarchical manner. In particular, they employ a 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 [13, 19]. Such algorithms have been proposed in [4, 13, 19] 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 [15] 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. Last but not least, a sliding window approach with fixed size window has been applied to the human genome [18, 14] to detect *G+C*-rich regions and CpG islands.

To the best of our knowledge, most current approaches target specific compositions (mainly $G+C$ -rich or CpG islands). Furthermore, there have been no studies on any relations that may occur between these regions. In this paper, we propose two general approaches to finding any type of “poly”-regions in DNA, and apply an efficient mining algorithm to extract frequent patterns between those regions.

In this paper: (1) we formally define the problem of detecting regions of high occurrence of a literal or set of literals in a sequence, (2) we propose two efficient algorithms to solve the problem, (3) we further apply an efficient arrangement mining algorithm to extract the complete set of frequent arrangements of these regions, (4) we provide experimental evaluation of our algorithms by testing their efficiency on the dog genome.

2 Problem Formulation

A sequence $\mathcal{S} = \{s_1, s_2, \dots, s_m\}$ is an ordered list of items, where each s_i belongs to an alphabet Σ . In the case of DNA sequence, each s_i corresponds to a nucleotide base and thus $\Sigma = \{A, C, G, T\}$, where A stands for *Adenine*, C for *Cytosine*, G for *Guanine* and T for *Thymine*.

A *High region* or *H-region* is a segment of \mathcal{S} , where there is a “high occurrence” of one or more items of Σ . Let $H^{d,k} = \{\mathcal{I}, p_{start}, p_{end}\}$ denote an *H-region* of k items with density d , that starts at item $s_{p_{start}}$ and ends at item $s_{p_{end}}$. \mathcal{I} is a set of items that works as a label for the *H-region* and is determined by the k dense items in the *H-region*. Also, $s_{p_{start}} \in \mathcal{I}$ and $s_{p_{end}} \in \mathcal{I}$. An *H-region* $H^{d,k}$ has density d , if each of the k items occurs in the region with a frequency of at least d/k %. d is the minimum % in the segment of the items in \mathcal{I} . Given a density threshold $min_density$, an *H-region* $H^{d,k}$ is *dense* if $d \geq min_density$. Consider for example the two *H-regions* in Figure 1; (1) $H^{80,1} = \{\{A\}, 5, 14\}$: in this case we have a region of “high occurrence” of nucleotide A with density 80%, (2) $H^{80,2} = \{\{A, C\}, 20, 29\}$: in this case we have a region of “high occurrence” of nucleotides A and C , where each one has a density of 40%. Notice that the density is meaningful for small values of k . In the case of DNA, k should either be 1 or 2, i.e. if a region has a high occurrence of all four nucleotides (or of three) then the occurrences are rather random and have no particular meaning. Note that the terms “poly”-region and *H-region* are synonyms and are used interchangeably in this paper.

Given two *H-regions* $H_1^{d,k} = \{\mathcal{I}^1, p_{start}^1, p_{end}^1\}$, $H_2^{d,k} = \{\mathcal{I}^2, p_{start}^2, p_{end}^2\}$, the *merging* of $H_1^{d,k}$ and $H_2^{d,k}$ is a new *H-region* $H_{12}^{d,k} = \{\mathcal{I}^{12}, p_{start}^{12}, p_{end}^{12}\}$, with $p_{start}^{12} = \min\{p_{start}^1, p_{start}^2\}$, $p_{end}^{12} = \max\{p_{end}^1, p_{end}^2\}$, and $\mathcal{I}^{12} = \mathcal{I}^1 \cup \mathcal{I}^2$. Notice that merging is only allowed when $\mathcal{I}^1 = \mathcal{I}^2$. Also, an *H-region* $H_1^{d,k} = \{\mathcal{I}^1, p_{start}^1, p_{end}^1\}$ is said to be contained in another *H-region* $H_2^{d,k} = \{\mathcal{I}^2, p_{start}^2, p_{end}^2\}$, if $p_{start}^2 \leq p_{start}^1$, $p_{end}^2 \geq p_{end}^1$, and $\mathcal{I}^1 = \mathcal{I}^2$. A dense *H-region* $H_1^{d_1,k}$ is *maximal*, if there exists no *H-region* $H_2^{d_2,k}$ such that $d_2 \geq min_density$ and $H_1^{d_1,k}$ is contained in $H_2^{d_2,k}$.



Figure 1. Example of two *H-regions*.

Problem Statement: Given a sequence $\mathcal{S} = \{s_1, s_2, \dots, s_m\}$, a density constraint d , a minimum window size min_win , a maximum window size max_win and a support threshold min_sup , we want: (1) to discover the complete set $\mathcal{H}_{\mathcal{S}}$ of maximal *H-regions* in \mathcal{S} , where each region $H^{d,k} = \{\mathcal{I}, p_{start}, p_{end}\}$ has density of at least d and size $|H^{d,k}| = p_{end} - p_{start} + 1 \in [min_win, max_win]$, and then (2) given $\mathcal{H}_{\mathcal{S}}$ and a support threshold min_sup , extract the complete set \mathcal{F} of frequent arrangements of *H-regions* in $\mathcal{H}_{\mathcal{S}}$.

3 Extracting H-regions

We present two approaches for extracting the set of *H-regions* in a sequence of items that belong to a given alphabet. The first approach is entropy-based and uses an existing recursive segmentation technique to split the input sequence into a set of homogeneous segments applying measures of divergence (in our case the *Jensen Shannon Entropy*) during the segmentation, whereas the second one implements a set of sliding windows over the sequence.

3.1 Recursive Segmentation

The idea of recursive segmentation based on a measure of divergence has been used in earlier works, [4, 13, 19], and [15] describes how it can be applied to DNA for detection of $G+C$ -rich regions and CpG islands. In this section we present an approach that applies the standard recursive segmentation algorithm used previously targeting any type of “poly”-region. The main difference in our approach is that the recursive segmentation does not use the standard s_0 [4] stopping criterion; instead, the recursion stops when the size of a segment drops below max_win .

The input sequence is recursively segmented, ensuring that the homogeneity difference (in our case the entropy) between the segments is maximized. To define the homogeneity difference between two segments, an appropriate measure λ is used. There is a variety of measures that can be used for the segmentation process, like the quadratic divergence (QD) [19], the Jensen-Shannon Divergence (JSD) [13], the Gini-Index, etc. In this work, we use the *Jensen-Shannon divergence*.

The target of the segmentation is a set of regions, where, in each region, the *Jensen-Shannon Entropy* is maximized. To achieve that, the input sequence is recursively segmented and each time a split point is chosen where the JSD value between the two segments is maximized, i.e. the distributions of the items in the two segments have maximal JSD value from each other. The recursive segmentation stops when it reaches a segment of size from min_win to max_win . The final seg-

mentation includes a set of regions of the desired size that are candidates for being H -regions. Through a sequential scan, each segment is checked whether it satisfies the density constraint and it is further expanded both ways until the density constraint is violated.

To improve the efficiency of the segmentation we apply a preprocessing step, which has been proposed in [15] for the detection of isochores. When looking for H -regions of two nucleotides, say nucleotide A and G , the original sequence is transformed to a new sequence that consists of only three types of literals: those that correspond to nucleotides A and G , and those that do not (represented by literal X). For example, $S = ACAAGCGA$ will be converted to $S' = AXAAAGXGA$. The benefit of this replacement is the following: consider a subsequence that is being split by the algorithm. If a high occurrence of two types of nucleotides occurs in one of the subsequences, its entropy will be larger than that of the other subsequences, since all nucleotides of different type (than the two frequent ones in the first sequence) are represented by one literal. Also notice that the above replacement improves the runtime of the algorithm, since the alphabet size is reduced, achieving faster entropy calculations.

3.2 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 alphabet item in that segment. Combining these statistics efficiently produces the complete set of H -regions in the sequence.

More formally, our algorithm is given a sequence S , a density factor d , a minimum window size min_win and a maximum window size max_win . The first step is to define a set of sliding windows \mathcal{W} . Let $\mathcal{W} = \{w^1, w^2, \dots, w^n\}$, where w^i corresponds to sliding window i and $n = |\mathcal{W}| = max_win - min_win + 1$. Each sliding window w^i is a triple $\{C^i, w_{start}^i, w_{end}^i\}$, where C^i is a set of statistics for w^i , w_{start}^i is an index to the starting position of w^i on S and w_{end}^i is an index to the ending position of w^i on S . C^i is a set of t counters $\{C_1, C_2, \dots, C_t\}$ one for each item in Σ . The value of each counter is the number of occurrences of the corresponding item in the window. Moreover, the piece of S 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.

The next step is to identify the set of H -regions using \mathcal{W} . Particularly, all the windows defined in \mathcal{W} will be sliding simultaneously. Conceptually, the above setting can be seen as having $M = max_win - min_win + 1$ levels of windows, one for each size. At any time instance, we check the statistics stored under each window in \mathcal{W} . If a set of items in a window w^i is found to satisfy the density constraint, then w^i is reported as an H -region. In parallel, we keep a list L of the H -regions discovered so far. Each record in L corresponds to an H -region label and points to a list of all the H -regions discovered so far with this label. Upon discovery of a new

H -region we insert it into L based on its label.

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 items of type c 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 items (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 H -region for each item.

The sliding window approach makes sense when the alphabet size is small, which holds for the application this paper is focused on, i.e. the DNA alphabet size is only four.

4 Discovering Frequent Arrangements of H -Regions in a DNA Sequence

Next, we apply an efficient algorithm [17] for mining frequent arrangements of H -regions on a single input sequence of event intervals.

4.1 Background

Let $\mathcal{E} = \{E_1, E_2, \dots, E_m\}$ be an ordered set of event intervals, called *event interval sequence* or *e-sequence*. As seen previously, each E_i is a triple $(e_i, t_{start}^i, t_{end}^i)$, where e_i is an event label, t_{start}^i is the event start point and t_{end}^i is the end point. The events are ordered by the start point. If an occurrence of e_i is instantaneous, then $t_{start}^i = t_{end}^i$. An e-sequence of size k is called a *k-e-sequence*. If the first event interval in an e-sequence of size m starts at point t_{start}^1 and the last event interval in the e-sequence ends at point t_{end}^m , then the *width* of the e-sequence is equal to $t_{end}^m - t_{start}^1 + 1$.

An arrangement \mathcal{A} of n events is defined as $\mathcal{A} = \{\mathcal{E}, R\}$, where \mathcal{E} is the set of event intervals that occur in \mathcal{A} , with $|\mathcal{E}| = n$, and R defines the pairwise relations between the event labels in \mathcal{E} . The size of an arrangement $\mathcal{A} = \{\mathcal{E}, R\}$ is equal to $|\mathcal{E}|$. An arrangement of size k is called a *k-arrangement*. Given an e-sequence s , s contains an arrangement $\mathcal{A} = \{\mathcal{E}, R\}$, if all the events in \mathcal{A} also appear in s with the same relations between them, as defined in R . Extending [17], we consider seven types of relations between two event intervals shown in Figure 2. Using these relations, general arrangements can be defined.

4.2 The Algorithm in Detail

Given an e-sequence S , 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 H -regions) of S . The window keeps sliding to the right (one event interval per slide) until it reaches the end of S , i.e. its right end includes the last event interval of S , for the first time. Based on this for-

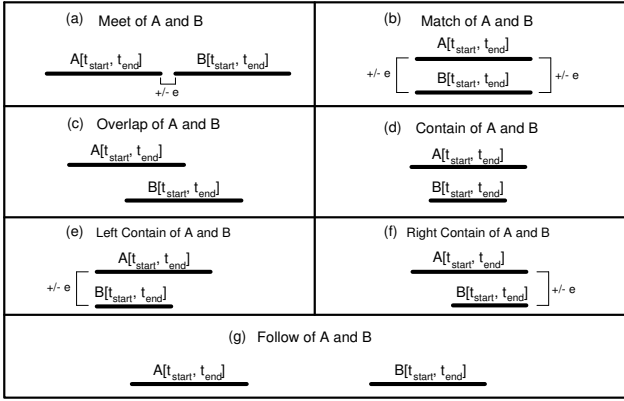


Figure 2. Basic relations between two event-intervals.

mulation, a total of $\mathcal{W} = |S| + win - 1$ 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: $freq(\mathcal{A}, win) = \frac{|\{w|\mathcal{A} \text{ occurs in } w\}|}{|\mathcal{W}|}$.

The algorithm uses the *arrangement enumeration tree* structure, introduced in [17], which is traversed in a DFS manner. The discovered arrangements are stored in a list L , along with their frequencies. In each window w , the set of arrangements contained in w is identified, and the list of active arrangements L is updated. If a new arrangement is found, it is inserted into L with support value 1. If an arrangement already exists in L , its frequency is increased by one. The complete set of frequent arrangements is determined by scanning the whole sequence and by increasing the support of each arrangement by one, for every window in which it occurs. Eventually, the complete set of frequent arrangements of H -regions in S is produced, by extracting those arrangements in L with support that satisfies the minimum support threshold.

5 Experimental Evaluation

Our datasets were obtained from NCBI (<http://www.ncbi.nlm.nih.gov>), that includes sequence records and map data generated at NCBI or used in NCBI resources. The files in this directory provide assembled sequences for the chromosomes of the reference assembly. For our experiments, 39 chromosomes (including the X chromosome) of the organism *Canis familiaris* (dog) have been used. Before applying our algorithms, the input DNA sequences have been pre-processed to remove the runs of N s. Eventually each sequence followed alphabet $\Sigma = \{A, C, G, T\}$. The experimental evaluation is divided into two phases. In the first phase, our algorithms have been applied to 39 chromosomes and have been compared in terms of runtime and accuracy. In the second phase, we applied the mining algorithm described in Section 4 to extract frequent arrangements of H -regions on 3 chromosomes (1, 2 and 38).

Chromosome	Arrangement	Support in %	Arrangement	Support in %
Chromosome 1		65%		69%
		56%		41%
Chromosome 2		63%		66%
		42%		53%
Chromosome 38		72%		56%
				52%

Figure 4. A sample of the Extracted Set of Frequent Arrangements for Chromosomes 1, 2 and 38 of the *Canis Familiaris*.

5.1 Extracting H-regions

The following factors have been considered: (1) size of the input sequence, (2) density of the H -regions, (3) size of the minimum and maximum windows.

Regarding runtime, the basic observation was that the sliding window approach outperformed the recursive segmentation approach both in small and large window ranges and density values. In Figure 3, we show the performance of each algorithm with regard to the density factor, which has been varied from 40% to 80%, on Chromosome 1 (approximately 127 million bases).

As far as accuracy is concerned, the sliding window approach finds the complete set of H -regions because it does exhaustive search. The recursive segmentation approach had poorer performance but did not drop below 85% in accuracy. This was expected, because the recursive segmentation, might choose split points inside some H -regions, and especially at the early segmentations when the segments are large. The accuracy of the recursive segmentation varies between 85% and 90%, performing slightly better in small sequences (Chromosome 38) and slightly worse in larger sequences (Chromosomes 1 and X). We also tried the recursive segmentation without the sequence conversion described in Section 3.1 and the performance dropped by an average of 15%.

5.2 Extracting Frequent Arrangements

Finally, the mining algorithm, described in Section 4, has been applied to the extracted H -regions to detect frequent arrangements between them. Specifically, the algorithm has been applied on the extracted H -regions of Chromosomes 1, 2 and 38 of the *Canis Familiaris*. The size of the sliding window was 1000. We managed to extract an interesting number of frequent patterns. In all three cases a great number of overlaps and contains has been detected between H -regions of C and G . These regions are in fact the $G + C$ -rich ones, and the CpG islands. Figure 4 gives a sample of the frequent arrangements that have been extracted for Chromosomes 1, 2 and 38 of the *Canis Familiaris*.

By and large, an interesting number of arrangements was

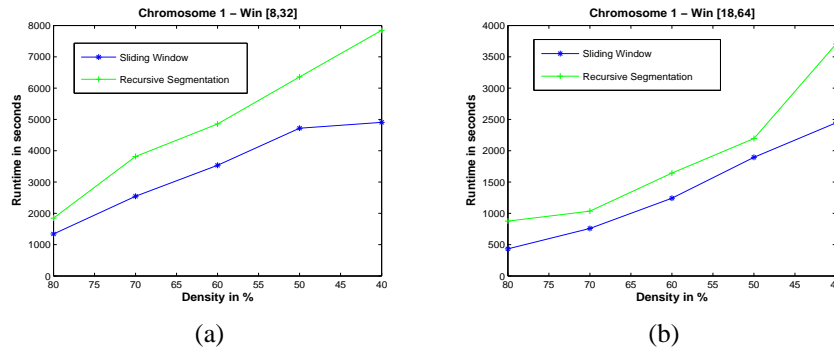


Figure 3. Results on Runtime Comparison: (a) Runtime Performance of the Two Algorithms for Chromosome 1 of the Canis Familiaris, with window range [8,32].; (b) Runtime Performance of the Two Algorithms for Chromosome 1 of the Canis Familiaris, with window range [18,64].

detected for these Chromosomes, with the minimum support threshold varying from 60% to 40%. Chromosome 38 gave the greatest number of arrangements despite the fact it was the smallest (in length) chromosome examined. The size of the extracted arrangements was limited between two and three. Note that in the above experiments the size of H -regions was in the range [18, 64]. When the algorithm was applied to smaller H -regions (of size [8, 32]): (1) the number of arrangements increased, (2) there was a significant increase in the number of relations of type *follow*, which is expected, since the smaller the size of event intervals, the greater the chance of a *follow* to occur.

6 Future Work

Some directions for future research include: (1) improvement of our algorithms to be able to work efficiently for larger alphabet sizes, (2) application of our algorithms to proteins, and (3) application of the mining algorithm to multiple DNA and protein sequences aiming the detection of arrangements of H -regions that occur frequently in these sequences.

References

- [1] J.F. Allen and G. Ferguson. Actions and events in interval temporal logic. Technical Report 521, The University of Rochester, July 1994.
- [2] I. E. Auger and C. E. Lawrence. Algorithms for the optimal identification of segment neighborhoods. *Bulletin of Mathematical Biology*, 51:39–54, 1989.
- [3] T. R. Bement and M. S. Waterman. Locating maximum variance segments in sequential data. *Mathematical Geology*, 9:55–61, 1977.
- [4] P. Bernaola-Galvan, R. Roman-Roldan, and J. L. Oliver. Compositional segmentation and long-range fractal correlations in dna sequences. *Physical Review E*, 53:5181–5189, 1996.
- [5] J. V. Braun, R. K. Braun, and H. G. Mueller. Multiple change-point fitting via quasi-likelihood, with application to dna sequence segmentation. *Biometrika*, 87:301–314, 2000.
- [6] J. V. Braun and H. G. Mueller. Statistical methods for dna segmentation. *Statistical Science*, 13:142–162, 1998.
- [7] E. Carlstein, H. G. Mueller, and D. Siegmund. Change-point problems. *Lecture Notes and Monograph Series*, 23(2), 1994.
- [8] G. A. Churchill. Stochastic models for heterogeneous dna sequences. *Bulletin of Mathematical Biology*, 51(1):79–94, 1989.
- [9] G. A. Churchill. Hidden markov chains and the analysis of genome structure. *Computes and Chemistry*, 16(2):107–115, 1992.
- [10] J. W. Fickett, D. C. Torney, and D. R. Wolf. Base compositional structure of genomes. *Genomics*, 13:1056–1064, 1992.
- [11] C. Freksa. Temporal reasoning based on semi-intervals. *Artificial Intelligence*, 54(1):199–227, 1992.
- [12] Y-X. Fu and R. N. Curnow. Maximum likelihood estimation of multiple change points. *Biometrika*, 77:563–573, 1990.
- [13] I. Grosse, P. V. Galvan, P. Carpena, R. R. Roldan, J. Oliver, and H. E. Stanley. Analysis of symbolic sequences using the jensen-shannon divergence. *Physical Review E*, 65:041905, 2002.
- [14] F. Larsen, G. Gundersen, R. Lopez, and H. Prydz. Cpg islands as gene markers in the human genome. *Genomics*, 13:1095–1107, 1992.
- [15] W. Li, P. Bernaola-Galvan, H. Fatameh, and I. Grosse. Applications of recursive segmentation to the analysis of dna sequences. *Computes and Chemistry*, 26(2):491–510, 2002.
- [16] H. Mannila and H. Toivonen. Discovering generalized episodes using minimal occurrences. In *Proc. of ACM SIGKDD*, pages 146–151, 1996.
- [17] P. Papapetrou, G. Kollios, S. Sclaroff, and D. Gunopulos. Discovering frequent arrangements of temporal intervals. In *Proc. of IEEE ICDM*, pages 354–361, 2005.
- [18] J. C. Venter. The sequence of the human genome. *Science*, 291:1304–1351, 2001.
- [19] C-T. Zhang, F. Gao, and R. Zhang. Segmentation algorithm for dna sequences. *Physical Review E*, 72:041917, 2005.