An Information Theoretic Approach for the Discovery of Irregular and Repetitive Patterns in Genomic Data

Willard Davis, Ananth Kalyanaraman* and Diane Cook
School of Electrical Engineering and Computer Science
Washington State University,
Pullman, WA 99164, U.S.A
{*wdavis, ananth, cook*}*@eecs.wsu.edu*

The unprecedented rate at which genomic data is accumulated underscores the need to develop highly efficient and powerful analytical capabilities. Traditionally, most of the effort post-sequencing has been focused on the identification and annotation of genes and their associated sequences such as promoters and regulatory elements. However, a major part of the vastness outside the gene-space is still left unexplored because of a lack of appropriate computational tools. Here, we propose a new approach for exploring and describing a genome without biasing the search process towards already known structural entities. Our primary objective is to discover novel conserved patterns that would typically fall off the scope of the current suite of repeat finding tools because of irregularities in their structure. The output is a hierarchy of patterns with arbitrary structural characteristics. A hierarchical representation captures the genomic sequence content at an abstract level and offers novel ways to examine the information contained in them. Our approach is an information theoretic search process which uses pattern matching techniques for processing the sequence data. Preliminary evaluation on the Drosophila genome has resulted in the finding of a number of irregular patterns, including a histone gene cluster. Discovering new patterns is an important problem in both whole- and comparative genomic application domains. It is our intent to use this research as a launch pad towards developing a comprehensive information-theoretic framework for conducting pattern and knowledge discovery on genomic data.

1. Introduction

A sustained interest and a continued investment in genome sequencing projects have led to an overwhelming growth of genomic sequence data. Hundreds of genomes have been sequenced within a decade's time (*http://www.ncbi.nlm.nih.gov/Genomes/*). This increasing availability of genomic data presents a unique opportunity for scientists to understand their fundamental composition and discern the patterns that govern the functioning of organisms directly from their genomes — a luxury in information that did not exist only a decade ago.

The tasks that typically follow genome sequencing are the identification, location, and structural/functional annotation of most (if not all) of its genes. This huge interest in genes, despite the fact that they are known to occupy only an insignificant portion of higher order genomes (e.g., under 3% in humans), is understandable because of their pivotal functional implications. Nevertheless, there are other genomic entities besides genes that are either known to play important biological roles or have functional identities that are as yet undiscovered. These portions of the genome, often labeled as "junk DNA", occupy the majority of genomes and have been gaining research focus of late. For instance, most genomes have abundant copies of identical or highly similar subsequences called "repeats" scattered all over them — e.g., the human genome contains at least 50% of its sequence in repetitive regions, while the wheat genome is expected to contain more than 90% in repeats. While some of the repeats are better understood for their roles in diseases[5], genomic evolution[10] and genomic rearrangements[3, 4], a majority are either uncharacterized and/or do not have a clear functional role identified yet[8]. Nonetheless, devising mechanisms to discover repetitive genomic portions and classify them into their respective types are essential steps towards determining their biological identity.

Repeat identification is a well-studied problem. Substantial research over the last decade has led to the development of several excellent repeat identification methods and software tools[2, 9, 12, 14, 18, 20, 24]. While these methods differ from one another in their underlying algorithms and approaches, most of them share the following

*Corresponding author.

set of characteristics in their general approach towards repeat identification: (i) detection based on sequence similarity, (ii) targeting specific types of repeats, and (iii) assuming that the set of structural attributes that characterize each of their target repeat classes is known a priori to the user so that they can be provided as part of the input. These attributes typically include length, sequence similarity and distance. This is a fair and effective approach to take when both the target repeat class and its structural signature are known and well defined.

The problem we propose here complements the existing approaches, with a scope not limited to conventional repeats. We are interested in capturing generic recurring "patterns" that are novel and potentially irregular that may fall out of the scope of the existing suite of repeat identifying tools. Detection of such patterns is geared towards enabling a novel means to "describe" a genome at an abstract level. Providing this capability would significantly enhance the scope of discovery beyond already characterized repeats to the vast expanse of previously unexplored types of recurrent patterns.

Our approach is based on information theory. Instead of targeting a specific repeat class, our approach detects recurring "patterns" satisfying multiple combinations of basic structural attributes such as sequence similarity, length, genomic proximity, frequency and periodicity. Our definition of "patterns" also supports the incorporation of annotated information during the process of pattern discovery. This is achieved by not restricting our method to just the DNA alphabet. The algorithm uses minimum description length[21] during its search and filter process which provides the information theoretic basis for our approach. The output of our algorithm is a hierarchy of detected patterns which are classified by the attributes they have in common and are ranked by their "interestingness" levels. Determining interestingness of repeating patterns is a ubiquitous challenge for data mining algorithms. For this application, interestingness encapsulates the confidence we have in a predicted pattern and the pervasiveness of the pattern throughout the data.

2. Problem Description

We call a genomic region an "interesting" *pattern* if it satisfies the following criteria: (i) there are a "significant" number of occurrences of the pattern in the data, (ii) for every occurrence, there exists at least another occurrence which satisfies an arbitrary set of structural constraints (given by Table 1), and (iii) each pattern occurrence can be decomposed into the same sequence of "blocks" where each block is either a smaller pattern or a contained stretch of nucleotides. Figure 1 illustrates this definition using an example. The decomposition of a pattern into blocks represents a description of the pattern and is referred to as its *signature*. Two or more patterns can share the same signature — e.g., even though p_1 and p_2 are two different patterns, they can be described the same way as a region in which a promoter is followed by a gene, a $3'$ UTR and a simple repeat (assuming α and β are both simple repeats). The above feature allows for incorporation of annotated information into the pattern discovery process rather than restricting similarity searches only at the nucleotide level.

3. Methods

The goal of this work is to discover new and interesting patterns in genomic data. Rather than focusing on a specific set of patterns, our algorithm performs an iterative search over all possible patterns, finding the most interesting, as measured by principles from information theory. The essence of this strategy is derived from a prior approach used by Cook and Holder in mining generic graph data [6], though here we enhance it by using pattern matching techniques to identify repeating sequences in DNA sequences.

The definition of our target patterns (described in Section 2) implies that the number of patterns may be at worst quadratic in the genome size (i.e., $O(n^2)$). This is because each pattern occurrence is basically a "substring" within a genome. To remove redundant results in a given output, we define a *maximal pattern occurrence* as one that is not entirely contained within an occurrence of a larger pattern. This implies that there are only $O(n)$ number of maximal pattern occurrences to be detected in the worst-case. However, if we start gathering mutually maximal pattern occurrences into sets then the number of such sets is bound only by the number of all possible subsets of the maximal patterns, which is exponential in the input size. We do not seek to enumerate all possible combinations of patterns and their occurrences; instead, we aim at reporting one

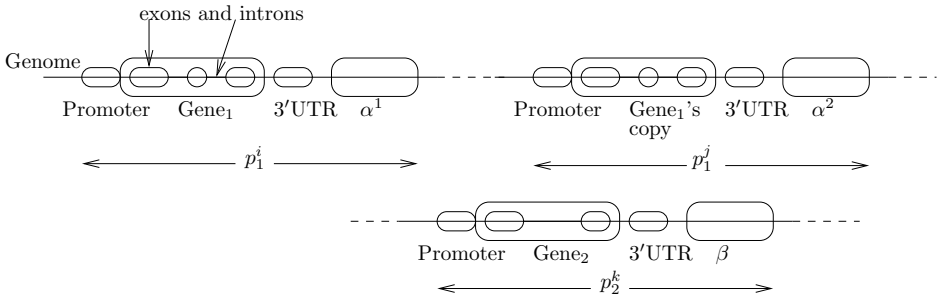

Fig. 1. Illustration of patterns: the figure shows two different patterns p_1 and p_2 with their respective occurrences $\{p_1^i, p_1^j\}$ and $\{p_2^k\}$. Occurrences of the patterns α and β are also shown. The patterns p_1 and p_2 bear the same signature: "promoter → gene→ 3′UTR→ conventional repeat pattern".

Table 1. Set of structural attributes (denoted by Θ).

Structural Attributes	Description
Length (in *bp*)	minimum and maximum length cutoffs for exact matches ($[MinExactMatch$, $MaxExactMatch]$), and inexact matches ($[MinAlignLen$, $MaxAlignLen]$)
Similarity (in % identity)	similarity cutoff defined in terms of alignment scoring ($MinSimilarity$)
Proximity (in *bp*)	minimum and maximum number of bases between starting points of any two similar occurrences of a pattern along a genome ($[d_{min}, d_{max}]$)
Orientation	patterns occurring in same or reverse strands

combination that minimizes the Minimum Description Length (MDL), which provides the information theoretic basis of our approach.

Our approach to pattern identification is an iterative procedure that achieves a bottom-up identification of patterns. (Which is to say that in case of nested patterns, the smaller patterns are found first.) We start with the original input, which is a genomic DNA sequence, and at each iteration apply the following three phases: (i) candidate pattern identification, (ii) candidate pattern evaluation, and (iii) sequence compression. At the end of each iteration, the sequence at that iteration is "compressed" to a smaller sequence using the patterns identified during that phase, which is then carried over as the input sequence for the next iteration. This is continued until no more new patterns could be found. We call our approach *RePDiG*, which stands for Repetitive Pattern Discovery in Genomes. The algorithm for RepDiG is presented in Figure 2 and its individual phases at each iteration are explained below.

Algorithm 1. *RePDiG*

RePDiG (Input: Sequence** G**, Structural Attributes** Θ**)

$S_0 \leftarrow G$, $\Theta_0 \leftarrow \Theta$, $i \leftarrow 0$, $P_0 \leftarrow \emptyset$.
REPEAT
(Phase 1; Section 3.1)
CandidatePatterns ← Identify candidate patterns in S_i *using* Θ_i *as constraints*
(Phase 2; Section 3.2)
P_i *← Evaluate each pattern in the CandidatePatterns list based on its compression value and perform a greedy selection*
(Phase 3; Section 3.3)
S_{i+1} *← Compress* S_i *using* P_i
Θ_{i+1} *← Recalculate the values of the structural attributes based on* S_{i+1}
UNTIL $P_i = \emptyset$
OUTPUT P_i

Fig. 2. RePDiG. G denotes a genomic DNA sequence, and Θ denotes the base values of the attributes specified in Table 1. P_i is a running list of patterns identified at iteration i.

3.1. Phase 1: Candidate Pattern Identification

The goal of this phase is to identify a set of candidate patterns in the input sequence that satisfy the required similarity, length and distance constraints as specified in Table 1. This is achieved by deploying a strategy of first identifying pairs of exact (maximal, to be precise) matching substrings as "seeds" and extending the seeds outwards through sequence alignment[12]. The rationale is that a substantially long ($MinExactMatch$) exact match is a necessary but not sufficient indicator for a satisfactory alignment ($MinSimilarity$) — thus, generating pairs of loci with long exact matching pairs provides a good filter to predict potential aligning regions.

One challenge in our approach is the need to handle an expanding alphabet set with each iteration. Only the sequence input in the first iteration is over the DNA alphabet; thereafter, every pattern identified and used for sequence compression contributes to a unique symbol in the alphabet for the following iteration. But the number of such patterns is bounded by $O(n)$ at any given iteration. This ensures that the size of the alphabet is also bound by $O(n)$ at any iteration. Our algorithm for seed generation uses the suffix array (SA) data structure[17], which is a lexicographically sorted array of all suffixes of a given sequence over $O(n)$ alphabet size, along with its longest common prefix (LCP) array. The LCP array is a $(n-1)$-long array where each $LCP[i]$ stores the length of the longest common prefix between suffixes $SA[i]$ and $SA[i+1]$. The algorithm is a minor variant of a previously developed approach[12] — our version takes into account the proximity parameter as well. The run-time cost of generating each seed is $O(1)$. The space complexity is $O(n)$. Due to lack of space, we omit the details of the algorithm.

Each generated exact matching seed is extended using traditional dynamic programming methods[22] until the computed similarity drops below the $MinSimilarity$ threshold or the length of the aligning regions exceeds $MaxAlignLen$. All such successful extensions are recorded in a list sorted by starting positions. This list is traversed to create a candidate pattern list through the following merging scheme: If a pair of similar regions have at least one of their occurrences significantly overlapping in its genomic positions with another occurrence from a different pair of similar regions, then all the four occurrences are combined to correspond to just one representative pattern. This is similar to the transitive-closure clustering scheme in Volfovsky *et al.*[24], with the main difference in our implementation which uses a union-find data structure[23] that enables us to perform each such merge in near constant time, independent of the size of the lists being merged. At the end of this mechanism we have a set of candidate patterns prevalent in the input sequence data, each of which has a list of its occurrences.

3.2. Phase 2: Candidate Pattern Evaluation and Selection

In order to decision which patterns identified in the first phase to report, we evaluate each candidate pattern according to how greatly it reduces the description length of the data. RePDiG's search is guided by the Minimum Description Length (MDL) principle[21] from information theory. This principle defines the best theory to describe some data as that theory which minimizes the number of bits required to describe the data. As specified in Equation 1, the MDL value of a pattern P can be defined as the description length of the original input sequence $DL(S)$ divided by the description length of the compressed sequence using pattern, or $DL(S|P)$. The best pattern is the one that maximizes this compression ratio.

$$Compression(P) = \frac{DL(S)}{DL(P) + DL(S|P)} \quad (1)$$

One way to use the MDL idea in our approach is to compress the sequence using the best discovered pattern before advancing to the next iteration. This implies we have as many iterations as there are number of such identified patterns. However at any given iteration, there are likely to be other patterns which do not overlap with the selected top pattern, and which when compressed along with the top pattern would yield a much higher aggregate compression value. To take advantage of this we developed an alternative method which selects a set of non-overlapping candidate patterns in a greedy manner so as to get the best aggregate compression value at any given iteration. This is achieved by computing the compression value for each candidate pattern, then ranking them in a non-increasing order, and perform a greedy selection of non-overlapping

candidate patterns. Given the prevalence of repeats and other patterns in genome data, this approach is expected to approximate the original single-pattern selection MDL approach while ensuring the practicality of our search process.

3.3. Phase 3: Sequence Compression and Parameter Transformation

Once a set of each candidate patterns are selected, each such pattern is given a unique character label. As there are only a $O(n)$ number of patterns that could be selected in the worst-case, an integer (i.e., $O(n)$) alphabet is sufficient over all iterations. The original sequence is then compressed into a new sequence such that each occurrence (ie., a substring) of a selected pattern in the original sequence is replaced by the unique character label associated with the pattern. In addition, we also store additional information describing each pattern used during compression in a separate record at each iteration. The transformed input sequence is then input to the next iteration.

Given that the next iteration is going to operate on the compressed sequence with a new alphabet, the current set of values used for the structural attributes (shown in Table 1) becomes no longer appropriate for the next iteration. Thus, we perform a simple transformation of each parameter value by scaling it down relative to the new length of the compressed sequence. For example, the new value of $MinExactMatch$ for the iteration $i+1$ is given by $MinExactMatch_{i+1} \leftarrow MinExactMatch_i \times \frac{|S_{i+1}|}{|S_i|}$, where $|S_i|$ and $|S_{i+1}|$ denote the lengths of input sequences at the start of iterations i and $i+1$ respectively. Note that there is no need to change the similarity threshold ($MinSimilarity$) after each iteration.

By repeating the process of finding a repeating sequence in DNA data and compressing the input string with this pattern, the RePDiG algorithm produces a hierarchical clustering of patterns found in the input data[11]. The resulting organization of discovered patterns is actually a lattice, where each cluster can be defined in terms of one or more parent patterns. The RePDiG algorithm iterates until no more compression can be obtained or the number of iterations exceeds a user-defined number.

4. Results and Discussion

We implemented the RePDiG software program in C. For validation, we used the *Drosophila melanogaster* genome. The 2005 FlyBase Release 4.1 version of the 120 Mbp genome was downloaded from the NCBI's GenBank repository. The genome consists of 6 chromosomal sequences: CHR_2R, CHR_2L, CHR_3R, CHR_3L, CHR_4 and CHR_X. For our experiments, we used a machine with a 2.33 GHz Xeon processor and 8 GB RAM.

Table 2. Table showing the running time for the first iteration of the RePDiG software on the Drosophila genome using a 2.33 GHz Xeon processor machine.

Chromosome	Length (in bp)	Runtime (minutes)
CHR_2L	23,011,544	96
CHR_2R	21,146,708	160
CHR_3L	24,543,557	41
CHR_3R	27,905,503	194
CHR_4	1,351,857	45
CHR_X	22,422, 827	1470

Table 2 shows the run-time results for executing RePDiG on each chromosome for the first iteration. As can be observed, the amount of time spent by the software analyzing each chromosome is not proportional to the length of the chromosome. This is because of the disproportionate number of seeds and patterns found in the chromosomal sequences. For example, the number of seeds generated on CHR_X was ≈200,000, as opposed to roughly ≈20,000 seeds on CHR_2L. While the preprocessing time to construct suffix and LCP arrays require only linear time proportional to the input size, a bulk of the time is spent on evaluating the seeds using dynamic programming alignment techniques. E.g., in our experiments, this accounted for than 90% of the run-time — a step that can be accelerated by distributing the alignment workload across multiple processors.

We spent our effort on analyzing just one chromosome (CHR_2L). We ran the program for several iterations, which led to the identification several complex pattern hierarchies such as a histone gene cluster.

5. Conclusions

Repeat identification is a thoroughly researched topic for over a decade now. As an increasing number of complete genome sequences there is a need to develop more sophisticated tools that can capture not

just conventional repeats, but also more complex and irregular patterns. In this paper we report the design and development of a novel pattern discovery method that uses an information theoretic basis to hierarchically describe a genome up to an arbitrary coarse level. Our approach has been developed using efficient pattern matching techniques. The results that we obtained using the preliminary version of our program indicates a promising research direction.

References

1. S.F. Altschul, W. Gish, W. Miller, E.W. Myers and D.J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology* 1990; **215**:403-410.
2. Z. Bao and S.R. Eddy. Automated de novo identification of repeat sequence families in sequenced genomes. *Genome Research* 2002; **12(8)**:1269-1276.
3. J.L. Bennetzen. The contributions of retroelements to plant genome organization, function and evolution. *Trends in Microbiology* 1996; **4(9)**:347–353.
4. N.J. Bowen and I.K. Jordan. Transposable elements and the evolution of eukaryotic complexity. *Curr Issues Mol Biol* 2002; **4(3)**:65-76.
5. J. Buard and A.J. Jeffreys. Big, bad minisatellites. *Nature Genetics* 1997; **15**:327-328.
6. D.J. Cook and L.B. Holder. Graph-based data mining. *IEEE Intelligent Systems* 2000; **15(2)**:32-41.
7. J.M. Coffin, S.H. Hughes, and H.E. Varmus. Retroviruses. *Plantview* 1997.
8. K.M. Devos and J. Ma and A.C. Pontaroli and L.H. Pratt and J.L. Bennetzen. Analysis and mapping of randomly chosen bacterial artificial chromosome clones from hexaploid bread wheat. *Proc Natl Acad Sci USA* 2005; **102(52)**:19243-19248.
9. R.C. Edgar and E.W. Myers. PILER: Identification and classification of genomic repeats. *Bioinformatics* 2003; **1(1)**:1-7.
10. R.B. Flavell. Repetitive DNA and chromosome evolution in plants. *Philosophical Transactions of the Royal Society of London. B.* 1986; **312**:227–242.
11. I. Jonyer and D.J. Cook and L.B. Holder. Discovery and evaluation of graph-based conceptual clustering. *Journal of Machine Learning Research* 2001; **2**:19-43.
12. A. Kalyanaraman and S. Aluru. Efficient algorithms and software for detection of full-length LTR retrotransposons. *Journal of Bioinformatics and Computational Biology* 2006; **4(2)**:197-216.
13. T. Kasai and G. Lee and H. Arimura and S. Arikawa and K. Park. Linear-time longest-common-prefix computation in suffix arrays and its applications. *Lecture Notes in Computer Science* 2001; **2089**: 1611-3349.
14. S. Kurtz and C. Schleiermacher. REPuter: fast computation of maximal repeats in complete genomes. *Bioinformatics* 1999; **15(5)**: 426-427.
15. International Human Genome Sequencing Consortium. Initial sequencing and analysis of the human genome. *Nature* 2001; **409**:860-921.
16. J. Karkkainen and P. Sanders. Simple linear work suffix array construction. *Lecture Notes in Computer Science* 2003; **2719**:943-955.
17. U. Manber and G. Myers. Suffix arrays: A new method for on-line string searches. *SIAM Journal of Computing* 1993; **22**:935-948.
18. E.M. McCarthy and J.F. McDonald. LTR_STRUC: a novel search and identification program for LTR retrotransposons. *Bioinformatics* 2003; **19(3)**:362-367.
19. B.C. Meyers, S.V. Tingey, and M. Morgante. Abundance, distribution, and transcriptional activity of repetitive elements in the maize genome. *Science* 1998; **274**:765–768.
20. P.A. Pevzner and H. Tang and G. Tesler. De novo repeat classification and fragment assembly. *Genome Research* 2004; **14(9)**:1786-1796.
21. J. Rissanen. Stochastic Complexity in Statistical Inquiry. World Scientific Publishing Company, 1989.
22. T.F. Smith and M.S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology* 1981; **147**:195-197.
23. R.E. Tarjan. Efficiency of a good but not linear set union algorithm. *Journal of the ACM* 1975; **22(2)**:1215-225.
24. N. Volfovsky and B.J. Hass and S.L. Salzberg. A clustering method for repeat analysis in DNA sequences. *Genome Biology* 2001; **2(8)**:RESEARCH0027.