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

Willard Davis, Ananth Kalyanaraman and Diane Cook

***Abstract*— The unprecedented rate at which genomic data is accumulated underscores the need to develop highly efficient analytical capabilities. Traditionally, most of the effort post-sequencing has been focused on the identification and annotation of genes along with their 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. Discovering new patterns is an important problem in both whole- and comparative genomic application domains. The proposed approach can provide an information-theoretic framework for conducting pattern and knowledge discovery on genomic data.**

I. 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 occupy 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 mis-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], [13], [15], [19], [21], [25]. While these methods may differ from one another in their algorithms and complexities, they all share the following theme in their underlying approaches:

i) detection based on sequence similarity, length and location parameters;
ii) targeted detection of specific types of repeats, and
iii) assumption that the set of structural attributes that characterize each targeted repeat class is known a priori to the user so that they can be provided as part of the input.

For example, a tandem repeat finder expects similar sequences to lie adjacent along the genome; an LTR retrotransposon finder looks for similar sequences representing the 5' and 3' LTRs within a distance 10Kbp-15Kbp of each other, while checking for other attributes as added criteria. 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 significance of the approach proposed here is that it complements existing approaches, and has a scope that is 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. For example, a non-coding gene paralog in a genome may be always found flanked by some repeat sequences unlike its functional counterpart; or, there could be genes that have conserved regulatory motifs in their proximity with a specific periodicity or frequency. These are examples of meta-level "patterns" that do not fall under the purview of traditional repeat finders. Nevertheless, they are "repeating patterns" at a coarser level that are statistically

Willard Davis, Ananth Kalyanaraman and Diane Cook are with the School of Electrical Engineering and Computer Science, Washington State University, Pullman, WA (email: {wdavis, ananth, cook}@eecs.wsu.edu).

This work was partially supported by National Science Foundation grant IIS-0505819.

supported in the input genomic data. Detection of such patterns can provide 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 and frequency. Our definition of "patterns" supports the incorporation of annotated information during the pattern discovery process. This is achieved by not restricting our method to just the DNA alphabet. This functionality is critical because plenty of annotation information is increasingly becoming available in public databases and the capability would allow our method to leverage state-of-the-art findings. The pattern discovery algorithm uses Minimum Description Length [22] 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, we explore interestingness as one that encapsulates the pervasiveness of the pattern throughout the data.

II. Related Work

Repeats occupy a majority of most known eukaryotic genomes; e.g., their constitution ranges from over 50% in the human [16] and maize genome [20], to over 90% in the wheat genome [8]. There are numerous known types of repeats — e.g., tandem repeats, retrotransposons (LTR and non-LTR variants), LINEs, SINEs, ALU, MITEs, pseudo-satellites, just to name a few. We will refer to these repeats as *conventional repeats* in this paper. Their classification is generally based on a set of known structural (and to an extent functional) attributes such as length, sequence similarity, and presence of some specialized motifs, genes and signals. As a result, there are now several specialized methods and excellent software tools available, each targeting individual subsets of repeat classes. For example, LTR_STRUC [19] and LTR_par [13] are programs that target full-length LTR retrotransposons. Programs such as RECON [2], REPuter [15] and PILER [9] allow a certain degree of generality by providing options for choosing a target repeat type from a set of characterized types.

A relatively open-intent classification scheme was developed into a program called RepeatFinder by Volfovsky *et al.* in 2001 [25]. RepeatFinder is a clustering method that first identifies similar subsequences in an input DNA sequence and performs a transitive-closure clustering for partitioning them into "clusters" based on overlap and distance. This strategy allows the grouping of similar subsequences in a genome and thereby provides a mechanism to classify new types of repeats. This method has been successfully applied on Arabidopsis, rice and microbial genomes to identify thousands of repeat classes. However, this method provides a snapshot of repetitiveness only at the nucleotide level. Our objectives are (a) to detect repeating patterns at a coarser levels with support for incorporation of annotation information, and (b) to employ an information theoretic means to determine the repetitiveness and interestingness of the genomic DNA.

III. 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 I), 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 said to be its *signature*. Two or more patterns can share the same signature — e.g., even though p_1, p_2 and p_3 may all look different at the nucleotide sequence level, they can be described in 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.

The above definition allows for portions within a pattern occurrence to be different at the nucleotide-level from their counterparts in other instances (e.g., despite gene_1 and gene_2 being different at the nucleotide level they can be part of the same pattern as shown in Figure 1). In general, our definition does *not* necessitate two subsequences to be similar at the nucleotide-level in order to be labeled the same pattern. For example, since p_1 and p_2 bear the same pattern signature, our model replaces their occurrences by an arbitrarily assigned unique label (corresponding to the pattern signature). Therefore, genomic regions p_1, p_2 and p_3 would be considered "similar" after being compressed with this label, even though they may be different at the nucleotide-level. The idea is to broaden the scope of our search methodology beyond just conventional repeats. The generality in this goal cannot be achieved by the current suite of available tools and methods because all of them operate at a nucleotide-level and check for only a well-defined set of structural constraints. Nevertheless, such generality becomes a necessary functionality to be implemented if we want to enhance the reach of genomic discovery for capturing non-trivial repeating patterns prevalent in genomic data.

IV. Methods

Rather than focusing on a specific class 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

TABLE I

SET OF STRUCTURAL ATTRIBUTES (DENOTED BY Θ).

Structural Attributes	Description
Length (in bp)	minimum length cutoff for exact matches ($MinExactMatch$), and minimum/maximum cutoffs for 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

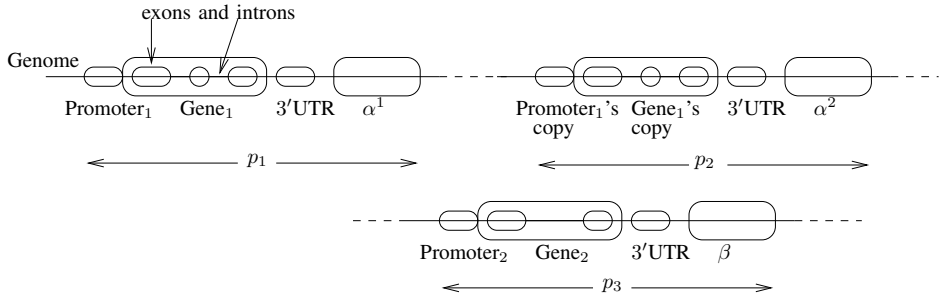

Fig. 1. Illustration of patterns: the figure shows a pattern with its different occurrences $\{p_1, p_2, p_3\}$. The pattern consists of other smaller patterns such as α and β, and bears the signature: "promoter → gene→ 3′UTR→ known repeat pattern".

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 patterns in DNA sequences.

The definition of our target patterns (described in Section III) implies that the number of patterns may be at most 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 redundancy in 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 bounded 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 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 (optionally) interspersed with annotation symbols, 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 can 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.

A. 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 I. 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 [13]. 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 this 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 and user-supplied annotation alphabet; thereafter, every pattern identified and used for sequence compression contributes to a unique symbol in the alphabet for the following iterations.

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 IV-A*)
$CandidatePatterns \leftarrow$ Identify candidate patterns in S_i using Θ_i as constraints
(*Phase 2; Section IV-B*)
$P_i \leftarrow$ Evaluate each pattern in the $CandidatePatterns$ list based on its compression value and perform a greedy selection
(*Phase 3; Section IV-C*)
$S_{i+1} \leftarrow$ Compress S_i using P_i
$\Theta_{i+1} \leftarrow$ 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 set of parameters as specified in Table I. P_i is a running list of patterns identified at iteration i. S_i is the sequence input to iteration i.

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 [18], 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 [13] — 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 this algorithmic variant.

Each generated exact matching seed is extended using traditional dynamic programming methods [23] 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 transitive-closure step, referred to as "pattern merging", is implemented using a union-find data structure [24], which 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.

B. Phase 2: Candidate Pattern Evaluation and Selection

In order to decide 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 [22] 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.

C. Phase 3: Sequence Compression and Parameter Transformation

Once a set of candidate patterns is 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 (i.e., 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 I) 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 [12]. 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.

D. Parallelization

To support large-scale data analysis on commodity clusters we parallelized the RepDig algorithm as follows. Let p be the number of processors. Our approach follows the master-worker paradigm, with a designated master node and $p-1$ worker nodes. At the start of each each RepDig iteration, the master node computes the seeds based on exact matches. This process is fairly quick ($<$10% of overall time) and so does not need to be distributed. The main computation effort is the evaluation of seeds using dynamic programming. Each alignment task is however independent and the master processor dynamically distributes the set of alignment tasks to the worker processors, which then compute the alignment and return the results to the master. The master processor then retains only those alignment tasks that succeeded, and then redistributes this reduced list for pattern instance merging. In this step, patterns that overlap over a genomic region are merged to form larger pattern instances. The results are gathered back at the master processor, which then passes the set of MDL filtered patterns to the next iteration. We have implemented the RePDiG software program in C and MPI (for inter-processor communication).

V. Results and Discussion

Experiments were conducted on the *Drosophila melanogaster* genome. The NCBI Release 5.1 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. Alongside, the genome annotation provided by FlyBase [11] was also downloaded through the NCBI Map Viewer web portal (`http://www.ncbi.nlm.nih.gov/projects/mapview/`). All our experiments were conducted on a Linux cluster of 24 nodes, each with 8 2.33 GHz Xeon processors and 8 GB RAM.

A. Data Preparation

For our experiments we prepared three sets of inputs from the Drosophila genome and annotation data.

- Input I) The genomic DNA sequence was used directly as downloaded from NCBI, one chromosome at a time.
- Input II) Genes and repeats annotation information was integrated into the nucleotide sequence of each chromosome. This was done by transforming the nucleotide sequence as follows: the nucleotide stretch corresponding to each annotated gene or repeat element is substituted by a unique symbol identifying that element. To account for the case where an element occurs at multiple locations, each occurrence is labeled with an occurrence identifier.
- Input III) Instead of labeling each gene by a different symbol, a unified symbol is used for all genes. Each gene occurrence, regardless of which gene it corresponds to, is assigned a unique occurrence identifier. The same treatment is given to repeats as well, with all the repeats getting a unified identifier different from the one applied to genes. This revised labeling scheme is then integrated into the nucleotide sequence of each chromosome, similar to Input II.

All input symbols were assigned an integer tuple (pid,oid), corresponding to a pattern identifier and an occurrence identifier. To conform to the transformation scheme, standard nucleotide symbols ($\{$A,C,G,T$\}$) were also assigned unique integer labels ($\{$1,2,3,4$\}$). Note that the alphabet size created by our transformation scheme is no longer a constant. In fact, it is limited by $O(n)$, where n is the length of the original genome.

As an example consider an input sequence:

$$s = AC \ldots ATG \ldots TAA \ldots ATT \ldots ATT \ldots$$

such that the block $ATG \ldots TAA$ corresponds to some gene i's first occurrence and the block $ATT \ldots ATT$ corresponds to some repeat j's fifth occurrence. Then, s is transformed into three strings s_1, s_2 and s_3, corresponding to Inputs I, II, and III respectively, as follows:

$s_1 =$ (1,1) (2,1) ... (1,10) (4,54) (3,34) ...
(4,100) (1,102) (1,103) ... (1,230) (4,430)
(4,431) ... (1,450) (4,700) (4,701) ...

$s_2 =$ (1,1) (2,1) ... (i,1) ... (j,5) ...

$s_3 =$ (1,1) (2,1) ... (g,1) ... (r,5) ...

where i,j,g, and r are values greater than the DNA alphabet size (4). Values i and j correspond to gene and repeat identifiers as per Input II, whereas g and r are the two unified identifiers for genes and repeats respectively as per Input III.

B. Experimental Results

Input I: Using the set of already annotated repeats, we first experimented with RepDig in order to study the effect of

TABLE II

PARAMETERS Θ_1 AND Θ_2 USED IN OUR EXPERIMENTAL STUDIES.

Attribute	Values & Cutoffs		
		Θ_1	Θ_2
Length (in *bp*)	$MinExactMatch$ [$MinAlignLen$, $MaxAlignLen$]	30 [100,5K]	15 [80,5K]
Similarity (in %identity)	$MinSimilarity$	75	60
Proximity (in bp)	$[d_{min}, d_{max}]$	[100,100K]	[100,100K]
Orientation	F for forward B for both	F	F
MDL	#patterns per iteration	10	10

parameters. We experimented with two sets of parameters, Θ_1 and Θ_2 on Input I. Θ_2 is a more relaxed setting than Θ_1. The corresponding parameter values are shown in Table II. For each parameter set, we performed an experiment on the Input I of chromosome 2R. The program was run for several iterations until no further new patterns were found. The results were compared against the standard annotated set of repeats. Genes are not part of this test because RepDig is designed to find only repeating patterns in the input. A cutoff of 100 bp was used as the minimum length required of a repeat to be qualified as a valid pattern in either schemes (RepDig vs. standard annotation).

Our analysis showed that RepDig was rather conservative in its pattern finding ability capturing only 224 of 2,684 known repeat elements under Θ_1. Under the more relaxed Θ_2 setting this number increased to 349 patterns. Upon further investigation we found that this is primarily because: (a) most repeating patterns were enumerated by RepDig during one of the iterations, but were not selected during the MDL-based filtering phase; (b) some of the annotated repeats were repeats across chromosomes, but in our studies we ran RepDig separately on each chromosome; and (c) a substantial number of annotated repeat families were largely divergent in their nucleotide content that they failed our parameter threshold. The average pattern length reduced from 936 bp to 796 bp from Θ_1 to Θ_2.

In addition to the known repeat elements, RepDig also found 181 new patterns under Θ_1 and 237 new patterns under Θ_2. These patterns satisfied the similarity and length cutoff criteria set by the parameter settings. Therefore, we suspect these were regions of genomic similarity left un-annotated by the standard suite of repeat identification tools.

Input II: In our second experiment, we ran RepDig on Input II for chromosomes 2L, 2R, 3L and 3R. The goal was to incorporate already annotated information into the pattern finding process. We used the Θ_2 parameter settings for all runs. For each chromosome we found over 100 RepDig patterns. About half the number of patterns included a small subset of already annotated genes and repeats as part of them. For example, in the 131 patterns detected by RepDig for chromosome 2L, a total of 24 genes and 106 repeat elements were included as part of some pattern's hierarchy. This is out of a total 2,756 genes and 10,495 repeats known on chromosome 2L. Table III summarizes our findings for all the chromosomes studied. As expected, the number of included repeats outnumber the number of included genes.

TABLE III

SUMMARY OF THE EXPERIMENTS ON INPUT II.

	RepDig Pattern Statistics			Annotation Statistics	
		Already annotated			
Chr.	#Patterns	#Genes	#Repeats	#Genes	#Repeats
2L	131	24	106	2,756	10,495
2R	131	11	45	3,025	10,688
3L	177	13	41	2,809	12,384
3R	107	12	33	3,549	12,890

Input III: We ran a third set of experiments on Input III. The purpose was to test the capability of RepDig to capture patterns at a more abstract level, without differentiating among genes (and among repeats). To achieve this, all genes are assigned a common identifying label, and all repeat elements are assigned another common label, as described in Section V-A. The results are summarized in Table IV for chromosomes 2L and 2R.

TABLE IV

SUMMARY OF THE EXPERIMENTS ON INPUT III.

	RepDig Pattern Statistics			Annotation Statistics	
		Already annotated			
Chr.	#Patterns	#Genes	#Repeats	#Genes	#Repeats
2L	259	53	180	2,756	10,495
2R	264	12	78	3,025	10,688

From comparing Table IV with Table III, it can be observed that the number of RepDig patterns found from Input III is more than double the number of patterns from Input II. This increase can be expected because patterns which would otherwise look different if we were to consider different genes/repeats as different, would be detected from Input III because of the unified labeling scheme. More interesting is the fact that the number of genes and repeats covered by the

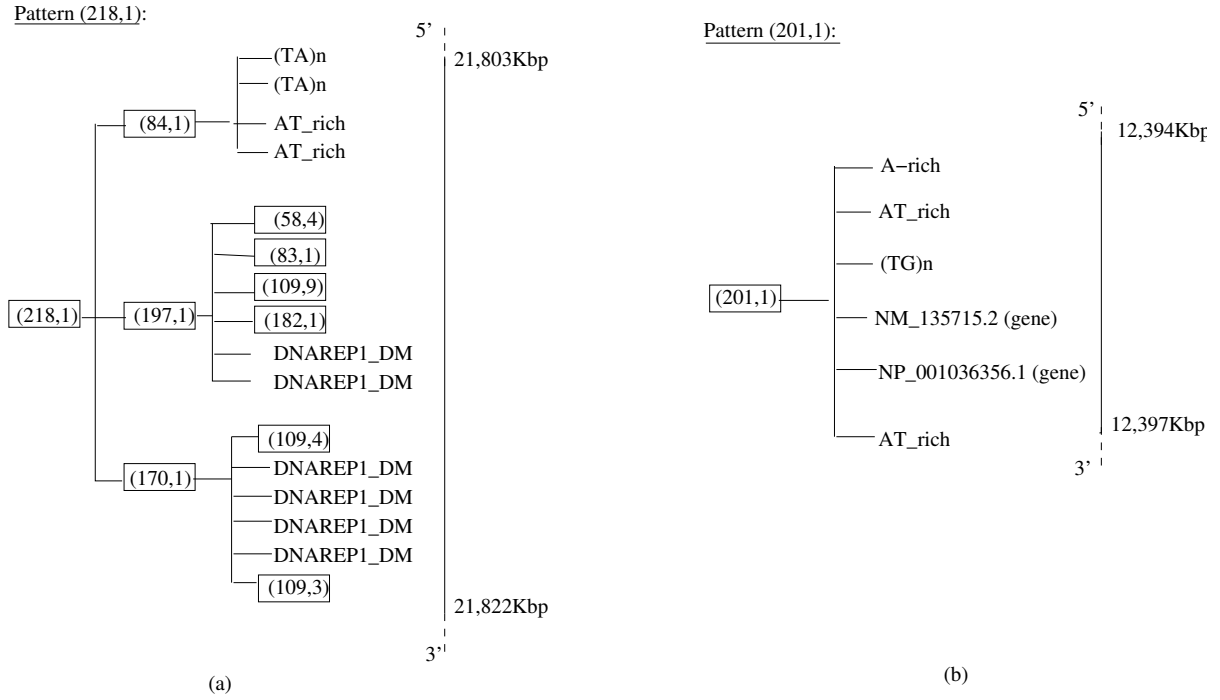

Fig. 3. Examples of test results on Input III: Hierarchy of two different patterns ((218,1) in (a) and (201,1) in (b)) reported by RepDig when run on Input III on chromosome 2L. Each boxed node denotes a RepDig pattern. Leaf nodes that are not boxed denotes a standard annotation element. The genomic co-ordinates that the patterns map to are also shown (not to scale).

set of RepDig patterns also significantly increased in Input III.

We examined a few RepDig patterns and found some interesting cases. Two such examples are shown in Figure 3. Pattern (218,1), shown in part (a), contains a 3-level hierarchy made up of smaller RepDig patterns and known annotated repeat elements. The smaller patterns were repeating blocks of nucleotides which were not captured in the standard annotation. Pattern (201,1), shown in part (b), is a smaller pattern (3Kbp) but contained a combination of known repeats and genes.

C. Performance

TABLE V
TABLE SHOWING THE RUNNING TIME FOR THE REPDIG PROGRAM ON INPUT II OF DIFFERENT DROSOPHILA CHROMOSOMES USING A 24-NODE/192-PROCESSOR LINUX CLUSTER.

Chr.	Length (in bp)	Runtime (minutes)	# Processors
2L	23,011,544	60	130
2R	21,146,708	500	30
3L	24,543,557	240	30
3R	27,905,503	570	30

Table II shows the run-time results for running the RePDiG program on chromosomes 2L, 2R, 3L and 3R. All runs were conducted on the 24-node Linux cluster described earlier. As can be observed, the amount of time that is spent by the program to analyze 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. While the preprocessing time to construct suffix and LCP arrays is linear in the input size, a bulk of the time is spent on evaluating the seeds using dynamic programming alignment techniques. For example, in our experiments, this accounted for more than 90% of the run-time — a step that is accelerated by RepDig through a dynamic scheme for distributing the alignment workload across multiple processors.

D. Strengths and Significance

For this paper, we tested our method against annotation information that included only already known genes and repeat elements. The RepDig algorithm and software, however, is generic enough that annotation information about any genomic elements such as promoters, miRNAs, nucleosomes, etc., can be directly input for analysis[1]. Incorporation of a more comprehensive annotation data will significantly enhance the value of the RepDig tool, as more interesting experiments can be designed to detect patterns at different levels of abstraction. For example, in one experiment we can include promoters, repeats, coding genes, and non-coding gene paralogs, and look for patterns that surround non-coding gene paralogs and differentiate them from the patterns that surround coding genes. Such differential studies can potentially lead to the understanding of the effect of genomic neighborhood on the function of genes (e.g., imprinted murine genes).

[1]As of this writing, such information were not available from Map Viewer portal.

Another significance of the RepDig tool is that it can be used as a tool to describe a genome in a compressed format. The set of patterns generated represents a compact genomic "*signature*" that has a footprint much smaller than the entire genome. Furthermore, since synteny can be typically expected among genomes of related species, the corresponding RepDig signatures from related species can also be expected to vastly intersect. Comparative studies along this direction can lead to building a pattern-based signature database, which can serve as a valuable and unique means to differentiate and relate species using the genomic content as the basis.

The RepDig approach has the advantage of being able to benefit from annotation information. Such information however may sometimes have inconsistencies. This can be overcome by consulting annotation standards such as the Gene Ontology convention (`http://www.geneontology.org/`). On the other hand, a tool like RepDig can help in flagging such inconsistencies. If two DNA blocks have been identified as instances of the same pattern and they do not share the same annotation information, then it is likely that one of the instances is missing annotation information.

VI. Conclusions

Repeat identification is a thoroughly researched topic for over a decade now. As increasing number of complete genome sequences are becoming available by the day, 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 RepDig, 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 demonstrate the utility of the tool and its potential in large-scale studies after taking into account multiple genomes and broader annotation information.

Acknowledgments

A poster highlighting our preliminary developmental effort was presented at the 2007 LSS Computational Systems Bioinformatics conference.

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.* **312**:227–242, 1986.

[11] W.M. Gelbart, M. Crosby, B. Matthews, W.P. Rindone, J. Chillemi *et al*. FlyBase: A Drosophila database. The FlyBase consortium. *Nucleic Acids Research* 1997; **25**:63-66.

[12] 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.

[13] 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.

[14] 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* **2089**: 1611-3349, 2001.

[15] S. Kurtz and C. Schleiermacher. REPuter: fast computation of maximal repeats in complete genomes. *Bioinformatics* **15(5)**: 426-427, 1999.

[16] International Human Genome Sequencing Consortium. Initial sequencing and analysis of the human genome. *Nature***409**:860-921, 2001.

[17] J. Karkkainen and P. Sanders. Simple linear work suffix array construction. *Lecture Notes in Computer Science* 2003; **2719**:943-955.

[18] U. Manber and G. Myers. Suffix arrays: A new method for on-line string searches. *SIAM Journal of Computing* 1993; **22**:935-948.

[19] E.M. McCarthy and J.F. McDonald. LTR_STRUC: a novel search and identification program for LTR retrotransposons. *Bioinformatics* 2003; **19(3)**:362-367.

[20] 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.

[21] P.A. Pevzner and H. Tang and G. Tesler. De novo repeat classification and fragment assembly. *Genome Research* 2004; **14(9)**:1786-1796.

[22] J. Rissanen. Stochastic Complexity in Statistical Inquiry. World Scientific Publishing Company, 1989.

[23] T.F. Smith and M.S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology* 1981; **147**:195-197.

[24] R.E. Tarjan. Efficiency of a good but not linear set union algorithm. *Journal of the ACM* 1975; **22(2)**:1215-225.

[25] 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.