

Reconstructing the Sequences of Broken Transposable Elements

Abanish Singh*, Umeshkumar Keswani*, David Levine*, Cedric Feschotte[†] and Nikola Stojanovic*[‡]

*Department of Computer Science and Engineering

[†]Department of Biology

University of Texas at Arlington, Arlington, Texas 76019, USA

[‡]Contact author: nick@cse.uta.edu

Abstract—Interspersed repeats, mostly resulting from the activity and accumulation of transposable elements, occupy a significant fraction of many eukaryotic genomes. More than half of human genomic sequence consists of known repeats, however a very large part has not yet been associated with neither repetitive structures nor functional units. We have postulated that most of the seemingly unique content of mammalian genomes is also a result of transposon activity, wrote software to look for weak signals which would help us reconstruct the ancient elements with substantially mutated copies, and integrated it into a system for *de novo* identification and classification of interspersed repeats. In this manuscript we describe the basic approach we used, and report on the methods for building the consensus sequences of these transposons.

Index Terms—Algorithms, DNA sequence analysis, graph, repeats, transposons.

I. INTRODUCTION

Most eukaryotic DNA is comprised of repetitive sequence. The repeats can be of tandem nature, or derived from mobile (transposable) genetic elements which move around the genomes, often duplicated in the process, over and over again. On the other hand, almost half of the human genomic sequence (and even more in rodents and many other species) is considered unique [8], but only a small fraction, about 5% of the total, is thought to be functional, whether coding or not. This leaves an open question about the origin and role of the presumably unique non-functional sequence, including gene introns, especially in the light of our earlier study which has established a remarkable micro-repetitive structure of these parts of the human genome [20]. It is very likely that this sequence also derives from transposons which have diverged so much that they cannot be recognized as such by current methods any more.

So far, many families of repeats have been manually annotated and deposited in databases such as Repbase [10], [11], then used by programs such as RepeatMasker (<http://www.repeatmasker.org>) to identify their traces in any given DNA segment. Primarily due to its reliance on already characterized consensus sequences for each family, this method is not optimal for newly sequenced genomes, as mobile element families in different species can be substantially different. Another computational approach to identifying transposons consists of *de novo* strategies, designed to identify previously unknown repeats in a genome based on their

copy counts, without resorting to homology with an outside catalog. Two basic approaches have been used for *de novo* identification of repeats: query vs query similarity searches and word counting/seed extension. The first relies on self-comparison, for example an entire genome aligned to itself, and it usually recognizes sequences as members of the same family when they share at least 85% nucleotide similarity over 75% of their length. This approach, employed by tools such as Miropeats [15], RepeatFinder [24] and RECON [4], is computationally intensive, requiring vast memory capacity and substantial processing time. Its results are also affected by the relative lack of sensitivity of the programs used for the initial self-comparison, mostly Blast [2] and Blast-like software, and subsequent problems related to the clustering methods. This often leads to an imprecise definition of the repeat ends. An alternative and increasingly popular approach involves word counting and seed extension. Methods based on these strategies bypass the need for whole genome alignments by building a set of repeat families starting with short strings (seeds) repeated in the genome, which are then progressively extended into a consensus through comparisons of their copies in the query sequence. RepeatScout [16] and ReAS [14] have been developed along those lines, however they rely on a substantial conservation of the elements, reflected in the high similarity between the copies. A recent study [19] has compared a large number of *de novo* repeat finding tools, and found ReAS and RepeatScout to be the most promising.

Despite of their increasing popularity and usefulness for annotating newly sequenced genomes, there are several problems associated with any attempt to computationally identify and characterize repeats. Repeat libraries, and in particular the division of transposon-derived sequences into families and other hierarchical structures, have been manually built, and the logic of these groupings was sometimes fuzzy, not naturally corresponding to different cutoff thresholds used by the software. That led to wide discrepancies between the numbers and groupings of repetitive elements determined automatically with these deposited in the curated databases. Second, the boundaries of the manually annotated elements were determined by considering various biological properties, and so far the software for *de novo* repeat identification was all but ignoring them, concentrating solely on the similarity of substrings in strings of letters over the DNA alphabet.

Consequently, while automatically determined consensus sequences (when these programs have been run on already studied genomes) usually did have a significant overlap with the annotated repeats, they almost never were a perfect match, either capturing just a part, or broadly extending over one or both sides of true transposon locations. Indeed, our recent study of the consensuses built by RepeatScout has shown that only 5% were (almost) perfect end-to-end matches, and these matched elements tended to be short.

In this paper we report on the efforts of our groups to integrate previously developed computational tools for the classification of repeated elements according to biological criteria [17] with *de novo* repeat element finding based on sequence information alone [21], [22]. The central part of that work was the creation of consensus sequences, which we address in the next section.

II. METHODS

We have reported on the core method for the identification of degenerated repeated sequences in a given genome elsewhere [21], [22], however our algorithm has substantially evolved since the last publication. We shall omit many technical details of the work which has already been published, and instead refer the reader to the original source, before proceeding with the description of the current effort on consensus building and repeat masking.

A. Identification of repetitive sequences

Our approach is based on the postulate that many short motifs which are dramatically over-represented in mammalian genomes [20] derive from ancient repeats which, over time, became so degraded by mutations that they cannot be recognized as copies of the original elements any more (thus creating an impression of a large amount of unique non-functional sequence). We start with the identification of such motifs, and try to associate them into groups which seem to co-occur with suspicious frequency.

Different transposons have copies conserved at different rates [7], and we assume that these which are more similar (presumably corresponding to more recent replication activity) will share long motifs in a relatively stable order, while these which have been substantially degraded will feature random subsets of short motifs, and in a more *ad hoc* manner (due to occasional randomly formed oligonucleotide sequences or multiple insertions at loci close to each other).

We attempt to isolate the repeated sequences in a series of iterative runs. In each iteration we identify a set of sequences, starting with these featuring most conserved copies and then looking for progressively more degraded elements. At the end of each run we mask the sequences we have identified so far, in order to exclude them from further consideration, reduce the size of seed motifs we attempt to cluster, and increase the number of times we need to see the seed motif repeated before we include it in the list used for further consideration. We end this process at about 30% degradation of the copies, when

the motifs become so short (7–8 characters) that their over-representation due to being a part of a transposable element becomes dwarfed by the number of chance occurrences (i.e. the variance of the numbers of chance occurrences), and when practically any transposon would feature several copies of the motif. The exact calculation of these numbers and the estimate of the probabilities of associations of such short seeds is a daunting task, so we have established them by running a large number of simulations and recording the effectiveness with which we could recognize longer degraded copies.

Each iterative run consists of several steps, as described in [21]:

- 1) We first locate short exact motifs significantly over-represented in the genomic segment under consideration. We start by counting the number of occurrences of oligonucleotides of fixed size (varies between iterations, from initial the length 14) in linear time, using a modification of the Karp–Rabin pattern matching algorithm [12]. We store the counts of each motif in a hash table, which we then scan in order to locate the most frequent entries. Using the Poisson distribution, we look for motifs with probability of a chance occurrence less than 10^{-5} . We keep these motifs as candidate seeds only if their count exceeds the expected number of occurrences plus a heuristic base count, which we have optimized through simulations to $\min\{1000, 10 \times M\}$, where M is the total length of the sequence in megabases. This was necessary to assure the minimal number of copies, since we need a sufficient number of motifs that we can work with, and for larger ones the basic probabilistic count may even be too small for a component of a transposon. We also took care that the selected seeds are truly informative, and eliminated these consisting of simple sequence (mostly remnants of poly-A tails) and tandemly repeated structure. For simple sequence identification, we measured the overall uncertainty as Shannon’s entropy, and discarded candidate seeds with the uncertainty lower than 1.5 bits (out of the maximum of 2 for 4 equally likely outcomes). Candidate motifs were further scanned for significant mutual overlaps, for which purpose we have modified the Smith–Waterman [23] algorithm to produce maximal gap-free local alignments and eliminated motifs where local alignment with one of higher p -value spanned more than 80% of the length of the shorter motif.
- 2) We next build a graph modeling the seed motifs as vertices and their associations as edges. In order to do that, we need to map the selected seed motifs back to their genomic locations, which we can again perform in linear time using the classic Aho–Corasick [1] algorithm and matching them all at once. Knowing the exact position of each occurrence of every seed, now represented as a vertex in our graph, we construct the edges connecting them by looking at windows of pre-defined size (determined heuristically as a fraction of the

size of an average transposon, by default set to 1000). If the motifs co-occur within the window, we add one weight unit to the edge connecting the corresponding vertices in the graph.

After assigning the weights we retain only these edges for which the probability of a random association of the corresponding vertices (assuming the uniform distribution of motifs in the original genomic segment) is less than 0.05, by default.

- 3) After the seed motif graph has been built, we proceed to locate cliques representing groups of motifs which repeatedly co-occur within the windows. The clique problem is NP-complete, so this step is computationally expensive, but it was nevertheless manageable even for entire chromosomes — this is because the graph size is dictated by the number of motifs we consider, a relatively stable value, and not by the size of sequence we analyze. Our computational method is based on locating intersections among adjacency lists of the vertices, through indexing and then using the C++ standard template library.
- 4) The next step involves the mapping of the cliques back to the genome, and we did this using the algorithm for locating the constrained heaviest segments. This algorithm, whose early version has been described by Jon Bentley in [5], and later used and modified by several authors, works on arrays of numerical scores, locating areas which “peak” over their environment in terms of their cumulative score, in time linear with the size of the score array. Briefly, a constrained heaviest segment is an interval $\mathcal{I}_{i..j}$ whose cumulative score S_{ij} is greater than or equal to the cumulative score S_{kl} of any of its subintervals $\mathcal{I}_{k..l}$, where $i \leq k \leq l \leq j$, and for which there is no interval $\mathcal{I}_{m..n}$, where $m \leq i \leq j \leq n$ and either $m < i$ or $n > j$, with $S_{mn} \geq S_{ij}$. By keeping track of the local minima and maxima of the cumulative score within the array, counting from its beginning, and updating the information about previous lower minima and higher maxima as the algorithm progresses through the array, one can report all constrained heaviest segments by the time the last array entry is processed (amortized linear time).

Locating these segments is necessary because the windows used to determine the initial motif associations are in general smaller than an average repeat, and without merging (resulting from segment determination) this would result in the fragmentation of element consensus. We determine the segments by first assigning a slight negative score (−1) to all base positions in the original sequence. As we know the genomic positions of each constituting motif, for each clique we can assign a positive score to all bases it covers. The optimal value for this score highly depends on the degradation level of the element represented by the clique, varying from +1 for perfectly conserved copies, to about +15 for very degraded ones (30% or more). We gradually increase this

value in each successive iterative run of our program. After assigning the scores and identifying the segments we consider them as the locations of the copies of the repeat element represented by the clique. If stopped at this point, this method bypasses the need to repeat mask the genome with the established consensus as a separate last step, which is the most time-consuming component in the performance of *de novo* repeat finders (which generally use RepeatMasker software for this purpose). However, although at this stage we know where the repeats are, and roughly which one corresponds to which clique, we still do not know what they are or what are the consensus sequences of the original transposons.

B. Determining the consensus sequences

The positions where the original cliques mapped to the genome can be used for the determination of the consensus sequence of the corresponding transposon. For well conserved copies resulting from recent insertional events this task is almost trivial, however when copies are degraded more than about 15% it becomes a challenge.

We first look at the heaviest segments resulting from the mapping of the cliques, and for each pair calculate their distance, as measured by the number of shared motifs. This is a computationally expensive, but nevertheless often necessary step (which we have made optional, as justified in the Discussion section below). Since that makes our problem one of the distance between sets, we have first tried traditional measures such as Jaccard distance [9] and several other alternatives, but they have not performed well in our context. We have thus modeled this problem as a drawing of a random variable from the hypergeometric family of distributions.

Motifs within the two segments we are comparing belong to the same superset, we can label it \mathcal{S} , of cardinality N : this is our original set of seed motifs which we used to build the graph. We can label the segments we are comparing as \mathcal{S}_1 and \mathcal{S}_2 , of cardinality $D_1 = |\mathcal{S}_1|$ and $D_2 = |\mathcal{S}_2|$. Thus, $D_1 \leq N$ and $D_2 \leq N$. We can also assume, without loss of generality, that $D_1 \geq D_2$. We model the intersection $|\mathcal{S}_1 \cap \mathcal{S}_2|$ as an experiment in which we draw D_2 motifs from \mathcal{S} , and consider the probability:

$$P_k(N, D_1, D_2) = \frac{\binom{D_1}{k} \binom{N-D_1}{D_2-k}}{\binom{N}{D_2}}$$

that the intersection of \mathcal{S}_1 and \mathcal{S}_2 contains exactly k motifs, i.e. that during the “assembling” of \mathcal{S}_2 exactly k motifs came from \mathcal{S}_1 . Consequently, if \mathcal{S}_1 and \mathcal{S}_2 share K motifs, the probability that these two segments would share K or more of them is calculated as $P_K = \sum_{k=K}^{D_2} P_k(N, D_2, D_2)$, and we can adopt it as a measure of distance between \mathcal{S}_1 and \mathcal{S}_2 .

We use this distance as a basis for the single linkage clustering of the segments. We set the cut on the resulting tree somewhat heuristically (again optimized through simulations) so that in every cluster we have at least 3 segments. In general, it is better to err in placing too few segments in a single cluster

than placing too many, since similar consensus sequences can always be further merged, while consensus built from only distantly related elements are bound to be poor.

After the clusters of segments have been built, we proceed to align the DNA sequences corresponding to the segments placed in a single cluster. Rather than writing the alignment software ourselves, we have interfaced to CLUSTALW [13]. We then looked at the columns of the resulting alignment in order to assign the consensus character to each position (or, better say, ancestral character, since we are determining the most likely sequence of the transposon whose copies we have identified).

While many columns do yield a natural consensus character, there is also a large fraction of these featuring substantial ambiguity. Conservatively, one would want to assign the letter ‘N’ (as a “do not know” symbol) to these positions, but for our purposes that would only lead to complications. Since we would like to use the consensus sequences we build to further mine the genome for similarities not initially discovered by our software, and also to attempt repeat classification, ‘N’ at any given position would not be of help. In particular, the RepeatMasker program would automatically consider it a mismatch, in addition to true character mismatches, potentially leading to quite a few missed copies. In our tests, sequences with more than 10-15% N’s did not yield almost any matches under RepeatMasker’s default settings, despite the overall good agreement of other, real, characters.

We have thus decided to follow the maximum likelihood principle for determining the consensus characters, and in cases of ambiguity instead of assigning an ‘N’ use evolutionary criteria, i.e. the existing knowledge about nucleotide substitution patterns. It is well known that the predominant substitution in vertebrates is the neighbor-dependent and irreversible CpG methylation deamination process ($\text{CpG} \rightarrow \text{CpA/TpG}$) [3]. Furthermore, studies on pseudogene sequences [25] have shown that at least in the human genome relative substitution rates for four nucleotides can be arranged in the following relative order: $\text{Substitution(G)} > \text{Substitution(C)} > \text{Substitution(A)} > \text{Substitution(T)}$. We use this information to resolve any ambiguity in cases where a majority character is not clear. We treat a gap in the multiple alignment as a character, but choose it for a consensus (insertion/deletion) only if it is present in a clear majority of aligned sequences.

C. Classification of consensus

Once the consensus sequence of each cluster of elements has been determined, we proceed to determine its biological properties, and attempt its classification. For that purpose we used the previously developed RepClass software [17].

RepClass is a toolset which automatically classifies transposable elements. It is a high throughput workflow model, leveraging our own custom scripts and other third party programs such as WU-blast (Warren Gish, 1996–2004, <http://blast.wustl.edu>), and palindrome and inverted detection from the EMBOSS suite [18], in order to identify and classify

transposons in new genomes. RepClass employs a multi-step approach which gathers classification information using several independent methods, and combines the collected information in order to come up with a tentative transposon classification. In particular, RepClass integrates the results of three independent classification methods: homology, target site duplication (TSD) search and structural search.

When using homology the software tries to classify sequence consensus by comparing them to already annotated transposon families in RepbaseUpdate [11]. During the target site duplication search RepClass looks for TSDs which are formed at the host site during the transposition. In the structural search it tries to classify the elements by identifying their structural characteristics such as terminal repeats flanking the transposon copies.

Classification of a single transposable element requires a single run of each of the classification methods listed above, whereas classification of a complete genome requires several hundreds or thousands of iterations. That task takes anywhere from several days to several months, depending on the size of the genome and the quality of its assembly, when done sequentially. Since the classification methods used in RepClass are independent and since the classification of one sequence is not dependent on the classification of any other, this process yields to a parallel implementation, which scales well on clusters and grid. We have thus run such implementation of RepClass on the DPCC (Distributed and Parallel Computing Center) cluster at the University of Texas at Arlington. DPCC currently consists of 81 dual processors, 2.667 GHZ and 2.44 GHZ Xeon compute nodes with 2GB of local memory each. The software used a varying number of processors on the cluster for different genomes, from 20, 40 and 60 (different in several runs) processors for *Drosophila* genome to 100 for human.

The topmost division RepClass attempts to achieve is the classification of the transposons into classes, depending on the presence of the traces of the reverse transcriptase coding region. If they can be recognized the element can be classified as Class I retroposon, replicating through an RNA intermediate, otherwise it is considered Class II. After that, RepClass attempts the identification of the right subclass: for Class I it looks at non-LTR retrotransposons (LINEs or SINEs), LTR retrotransposons, DIRs and Penelope-s. For Class II it looks at Maverick-s, DNA transposons and Helitron-s. If the subclass has been successfully identified, it goes one step further in an attempt to allocate the superfamily. We look at the large number of possible superfamily classifications, and a manuscript detailing the biological aspects of this work is currently in preparation.

III. RESULTS

We have run our software toolkit on both simulated sequences (providing a well controlled environment in which we knew exactly how many repeats were inserted, and where they were), and real genomic data (human chromosome 21). We have used these sequences to (1) calibrate the numerous parameters to

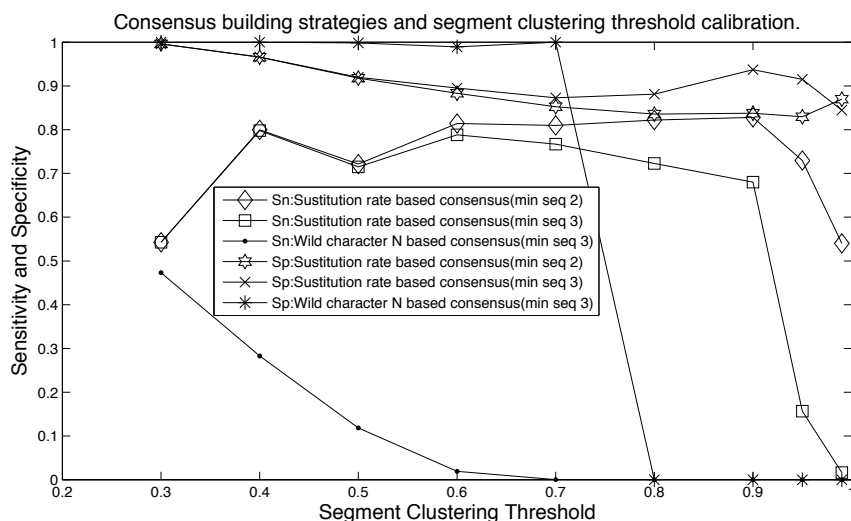


Fig. 1. Simulation results for consensus building in the simulated environment, for varying clustering thresholds and character assignment strategies. “Sn” stands for Sensitivity and “Sp” for Specificity.

our software, (2) compare the performance of our software with other publicly available tools, RepeatScout [16] and PILER [6] in particular, and (3) to estimate how much our findings agree with the annotations derived from RepBase Update, through RepeatMasker. In addition, we were also interested in how many of the consensuses we build can be classified, and how would these classifications relate to these from manually established genome annotations.

The first tests were performed in a simulated environment which we have created to resemble real genomic data. It consisted of 1.3 million bases of “testbed” sequence, one third of which was composed of an entirely random assembly of four DNA letters, another third produced by a second-order Markov model trained on one million bases from human chromosome 2, and the last third obtained from bacterial sequences from the *E. coli* genome (considering that prokaryotic sequences are generally non-repetitive). In order to complete the simulation dataset, we inserted 150 copies each of two previously characterized repeats, TIGGER and LIMCA_5, in the testbed sequence. We finalized the setting by introducing random mutations into this conglomeration, and degraded the sequence for about 30%. The mutations involved nucleotide substitutions, and no insertions and deletions. Since our model builds the tentative repeated elements from short pieces, and does not rely on positional information (at least until the elements are identified and the consensus is being constructed), the lack of indel mutations was not significantly affecting our results.

In order to calibrate the mostly orthogonal parameters to various components of our software suite we have run several hundred simulation runs. We have described the calibration of core repeat finding in [22], so we here concentrate only on our efforts to optimize the consensus building and the subsequent classification. We ran the measurements and computed the

sensitivity and specificity of locating the elements we have inserted (measured as the percentage of base positions) for three sets of specifications: (1) substitution rate based approach to consensus building with a minimum of 2 participating sequences (determined by the cutoff in the clustering tree), (2) substitution rate based approach to consensus building with a minimum of 3 participating sequences, and (3) ambiguity character (‘N’) based approach to consensus building with a minimum of 3 participating sequences. The outcome of this simulation is shown in Figure 1. In the simulations we have varied the clustering threshold (defined in the Methods above), from 0.3 to 0.99. The calibration has indicated that our software achieves the best tradeoff between the sensitivity and specificity at 0.4 threshold with a minimum of 3 sequences in the alignment, maximum likelihood derivation.

We have also estimated the effectiveness of our approach on the simulated sequence, and compared it with the results produced by PILER and RepeatScout. The results of a representative experiment, in which the testbed sequence and insertions have been constructed as described above, but the conglomeration was progressively degraded at 10%, 20% and 30%, are shown in Table I, which we originally presented in [21].

In the continuation of this work, reported here for the first time, we have decided to look at the results of RepeatScout only, since it was both our observation and the result of other related studies [19] that it generally outperforms other tools, including PILER. Moreover, RepeatScout is now emerging as a *de facto* standard in *de novo* repeat annotation, and thus as a benchmark with which any new solution should be compared.

Looking at the real genomic data, we have compared the performance of our algorithm with that of RepeatScout, using human chromosome 21. Since this comparison was mainly targeted towards identifying known and previously charac-

TABLE I
THE PERFORMANCE OF OUR SOFTWARE ON THE SIMULATED DATA SET, IN ONE REPRESENTATIVE EXPERIMENT. "Sn" STANDS FOR SENSITIVITY AND "Sp" FOR SPECIFICITY. TABLE REPRODUCED FROM [21].

Degradation	PILER Sn	PILER Sp	RepeatScout Sn	RepeatScout Sp	Our Algorithm Sn	Our Algorithm Sp
0%	1.0	0.999	1.0	0.999	0.998	0.982
10%	0.0	0.0	0.999	0.999	0.973	0.979
20%	0.0	0.0	0.999	0.999	0.748	0.738
30%	0.0	0.0	0.0	0.0	0.708	0.593

TABLE II
PERFORMANCE COMPARISON OF OUR ALGORITHM AND REPEATSCOUT IN IDENTIFYING KNOWN AND PREVIOUSLY CHARACTERIZED REPEATS.

	Our Algorithm Sn	Our Algorithm Sp	RepeatScout Sn	RepeatScout Sp
Experiment 1	0.5230201	0.8714505	0.4466029	0.9938296
Experiment 2	0.5752110	0.8680427	0.6082956	0.9840795

TABLE III
CLASSIFICATION COMPARISON OF OUR ALGORITHM AND REPEATSCOUT IN IDENTIFYING KNOWN AND PREVIOUSLY CHARACTERIZED REPEATS. "FAMILIES" REFER TO THE DETERMINED NUMBER OF CONSENSUS SEQUENCES.

	Total Families	Classified Families
Our Algorithm	1088	570
RepeatScout	422	235

terized repeats, it was easy to calculate the sensitivity (Sn) and specificity (Sp) by repeat masking the chromosome 21 using the RepBase human library and the libraries produced by both programs, then comparing the masked characters. The results are shown in Table II. Experiments 1 and 2 have been performed under different parameter settings for both programs. In experiment 1, we have chosen the l -mer size 7 for RepeatScout, making this size the same as the size of the smallest motif used by our software. In experiment 2, we have attempted to maximize the sensitivity and specificity for each program, separately, by choosing the optimal l -mer size (i.e. 14) for RepeatScout, and by choosing a different threshold for selecting gap over a character in the consensus building from multiple alignments in our software. The outcome of the classification by the RepClass for experiment 2 data is shown in Table III.

The results shown in tables II and III indicate that our method achieves results comparable with these of RepeatScout when identifying known, previously characterized, repeat families. However, the lower specificity of our program also indicates that we may have found additional repeat elements which are, presumably, too broken to be readily recognized, and which have thus not been identified yet. In order to see how well our program would identify broken, old, and previously unknown repeats (and confirm that we are indeed capturing these, and not just noise), we repeat masked chromosome 21 using the RepBase human library, and ran both our software and RepeatScout on the remaining unmasked sequence. The results of this experiment are shown in Table IV. It is worth noting that out of 10 repeat elements identified by

RepeatScout, most were very small (on the order of 100 base pairs or less). In contrast, the elements reported by our method were large, varying between 1000 and 5000 base pairs, on average.

TABLE IV
COMPARISON OF OUR ALGORITHM AND REPEATSCOUT IN IDENTIFYING OLD, BROKEN, PREVIOUSLY UNIDENTIFIED REPEATS. "FAMILIES" REFER TO THE DETERMINED NUMBER OF CONSENSUS SEQUENCES.

	Total Families	Classified Families
Our Algorithm	183	28
RepeatScout	10	0

IV. DISCUSSION

While the performance of our tool on well conserved, and thus very similar, copies of recent transposons is comparable with that of other *de novo* finders, it clearly over-performs even the best current tools in the identification of highly degenerated repeated elements. However, while it can locate low-copy number repeats in case of well conserved elements, it is only capable of finding intensively replicated highly degraded transposons. This places a constraint on its power, however one which we have to accept. When the noise overwhelms the signal there is very little one can do in order to reconstruct it.

Unlike the other currently available tools, our program can find the locations of repeats in a genome even before their consensus sequences have been reconstructed, and thus it does not depend on the RepeatMasker to report them. Often, this is all that the user wants: mask the clearly repetitive structures in order to concentrate on more interesting sequences. Furthermore, because of the expenses of running the time-consuming RepeatMasker matches, our software can accomplish the basic masking within minutes, with results comparable to what other tools, RepeatScout in particular, accomplishes in hours. This is not to say that our software runs in an instant — for all but short segments (where looking for repeats would not make much sense, anyway) the full masking is indeed computationally expensive, only that it performs the core tasks of repeat identification an order of magnitude faster.

There are cases when full annotation of large genomes is needed, such as following a new round of sequencing and assembly. For this purpose our program can be run in extensive mode, determining the consensuses and invoking the Repeat-Masker with this catalog (this strategy can detect additional copies which have been omitted when segments were laid out), as well as performing the classification. Depending on the genome size and the power of the computational infrastructure one has, this may take days of computation, however full *de novo* annotation of complete genomes is not a common task executed daily, and in this context one can afford the wait.

There is still work that can be done to improve our system. We are not content with setting up of the cuts in the segment clustering tree so that it results in a heuristic minimal amount of sequences in each cluster — these cuts should be made such that the resulting clusters precisely correspond to the division of elements into families a biologist would make when manually constructing a library. This is a very difficult task to achieve, as reflected in the poor correspondence of any automatically derived consensus (i.e. by any current tool) to the existing annotations of classes of transposable elements. Nevertheless, the sequences we discover are clearly repetitive, and a good share of them do classify. Even as most of our consensuses would need further work in merging, splitting, trimming or extending, this shows that the signal we capture is indeed real.

ACKNOWLEDGMENT

This work has been partially supported by NIH grant 1R03LM009033-01A1 to NS.

REFERENCES

- [1] A.V. Aho and M.J. Corasick. Efficient string matching: An aid to bibliographic search. *Comm. ACM*, 18:333–340, 1975.
- [2] S.F. Altschul, W. Gish, W. Miller, E.W. Myers, and D.J. Lipman. Basic local alignment search tool. *J. Mol. Biol.*, 215:403–410, 1990.
- [3] P.F. Arndt. The history of nucleotide substitution pattern in human. *Biophysics*, 48:Suppl. 1, 2003.
- [4] Z. Bao and S.R. Eddy. Automated *de novo* identification of repeat sequence families in sequenced genomes. *Genome Res*, 12:1269–1276, 2002.
- [5] Jon Bentley. Programming pearls: algorithm design techniques. *Comm. ACM*, 27:865–873, 1984.
- [6] Robert C. Edgar and Eugene W. Myers. PILER: identification and classification of genomic repeats. *Bioinformatics*, 21:i152–i158, 2005.
- [7] Cedric Feschotte. Transposable elements and the evolution of regulatory networks. *Nature Rev. Genet.*, Electronic version ahead of print, 2008.
- [8] International Human Genome Sequencing Consortium. Initial sequencing and analysis of the human genome. *Nature*, 409:860–921, 2001.
- [9] Paul Jaccard. Distribution de la flore alpine dans le bassin des drouces et dans quelques regions voisines. *Bull. Soc. Vaud. Sci. Nat.*, 37:241–272, 1901.
- [10] J. Jurka. Repbase Update: a database and an electronic journal of repetitive elements. *Trends Genet.*, 9:418–420, 2000.
- [11] J. Jurka, V.V. Kapitonov, A. Pavlicek, P. Klonowski, O. Kohany, and J. Walichiewicz. Repbase Update, a database of eukaryotic repetitive elements. *Cytogenet. Genome Res.*, 110:462–467, 2005.
- [12] R. Karp and M. Rabin. Efficient randomized pattern matching algorithms. *IBM J. Res. Devel.*, 31:249–260, 1987.
- [13] M.A. Larkin, G. Blackshields, N.P. Brown, R. Chenna, P.A. McGettigan, H. McWilliam, F. Valentin, I.M. Wallace, A. Wilm, R. Lopez, J.D. Thompson, T.J. Gibson, and D.G. Higgins. Clustal W and Clustal X version 2.0. *Bioinformatics*, 23:2947–2948, 2007.
- [14] R. Li, J. Ye, S. Li, J. Wang, Y. Han, C. Ye, H. Yang, J. Yu, and G.K. Wong. ReAS: Recovery of ancestral sequences for transposable elements from the unassembled reads of a whole genome shotgun. *PLoS Comput. Biol.*, 1:e43, 2005.
- [15] J.D. Parsons. Miropeats: graphical DNA sequence comparisons. *Comput. Appl. Biosci.*, 11:615–619, 1995.
- [16] A. Price, N.C. Jones, and P.A. Pevzner. *De novo* identification of repeat families in large genomes. *Bioinformatics*, 21:i351–i358, 2005.
- [17] N. Ranganathan, C. Feschotte, and D. Levine. Cluster and grid based classification of transposable elements in eukaryotic genomes. In *Proceedings of CCGrid06, 6th IEEE Int. Symp. on Cluster Computing and the Grid*, page 45, 2006.
- [18] P. Rice, I. Longden, and A. Bleasby. EMBOSS: The European Molecular Biology Open Software Suite. *Trends Genet.*, 16(6):276–277, 2000.
- [19] S. Saha, S. Bridges, Z.V. Magbanua, and D.G. Peterson. Empirical comparison of *ab initio* repeatfinding programs. *Nucleic Acids Res.*, Electronic version ahead of print, 2008.
- [20] A. Singh, C. Feschotte, and N. Stojanovic. A study of the repetitive structure and distribution of short motifs in human genomic sequences. *Int. J. Bioinformatics Research and Applications*, 3:523–535, 2007.
- [21] Abanish Singh, Cedric Feschotte, and Nikola Stojanovic. Micro-repetitive structure of genomic sequences and the identification of ancient repeat elements. In *Proceedings of BIBM 2007, The IEEE International Conference on Bioinformatics and Biomedicine*, pages 165–171, 2007.
- [22] Abanish Singh and Nikola Stojanovic. An algorithm for finding substantially broken repeated sequences in newly sequenced genomes. In *Proceedings of ICMB 2007, International Conference on Mathematical Biology*, volume 971, pages 79–88, Melville, New York, 2008. American Institute of Physics.
- [23] T.F. Smith and M.S. Waterman. Identification of common molecular subsequences. *J. Mol. Biol.*, 147:195–197, 1981.
- [24] N. Volfovsky, B.J. Haas, and S.L. Salzberg. A clustering method for repeat analysis in DNA sequences. *Genome Biol.*, 2:RESEARCH0027, 2001.
- [25] E.F. Zhang and M. Gerstein. Patterns of nucleotide substitution, insertion and deletion in the human genome inferred from pseudogenes. *Nucleic Acids Res.*, 31:5338–5348, 2003.