Published in final edited form as:

Int J Bioinform Res Appl. 2010 ; 6(2): 147–162.

An algorithm for the reconstruction of consensus sequences of ancient segmental duplications and transposon copies in eukaryotic genomes

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

[†]Department of Computer Science and Engineering, University of Texas at Arlington, Arlington, TX 76019, USA

[‡]Department of Biology, University of Texas at Arlington, Arlington, TX 76019, USA

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 our approach, and report on our methods for building the consensus sequences of these transposons.

Keywords

Algorithms; graphs; DNA sequence analysis; DNA sequence repeats; transposons

1 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 by current methods.

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 [25] 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 often leading 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 using this approach, however they rely on 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 effective.

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 their groupings was sometimes fuzzy, not naturally corresponding to cutoff thresholds used by the software. That led to widespread discrepancies between the numbers and groupings of repetitive elements determined automatically with these deposited in 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 4-letter 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 good match, either capturing just a part of the transposon, or broadly extending over one or both sides of true transposon locations. Indeed, our recent evaluation of the consensus built by RepeatScout has shown that only about 5% were (almost) perfect end-to-end matches, and these matched elements tended to be short.

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

2 Methods

Our initial efforts on the design of the core method for finding degenerated repeated sequences in human and other genomes have been reported in a series of publications [21,22]. We here integrate that approach with the latest extensions, consensus building and repeat masking in

particular, and with previously unrelated work on transposon classification [17], to present the first comprehensive report on this research.

2.1 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, enriched with conserved motifs, which many motif-finding projects then tend to flag as putatively functional). We start with the identification of over-represented short motifs, then try to associate them into groups which seem to co-occur with suspicious frequency.

Different transposons feature 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, whereas these which have been substantially degraded will be characterized by 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 round 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 chance occurrences (i.e. the variance of the numbers of chance occurrences, dictating that any possible contribution from a transposon can conceivably be just a random variation), 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.

The remainder of this section, and Section 2.2 below, describe the steps performed in each iterative run.

We first locate short exact motifs, which we denote by v'_i , significantly over-represented in the genomic segment S under consideration. We start by counting the number of occurrences of oligonucleotides of fixed size (varies between iterations, from initial length 14 downward) in linear time, using a modification of the classic Karp–Rabin pattern matching algorithm [12]. We represent each motif as a number in a base-4 system, and store their counts in a hash table, which we subsequently scan in order to locate the most frequent indices. Using the Poisson model, we look for repetitions whose probability of a chance occurrence is less than 10^{-5} (which had to be set so low in order to prevent the inclusion of too many motifs, as indicated by our findings in [20]). However, we keep the corresponding 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 $\min\{1000, 10 \times L\}$, where L 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 to work with, and for larger ones the basic probabilistic count may even be too small to be considered a component of a transposon.

We also took care that the selected seeds are truly informative, and eliminated these v_i' consisting of simple sequence (mostly remnants of poly-A tails) and these exhibiting a tandemly repeated structure. For simple sequence identification, we measured the overall uncertainty as Shannon's entropy (with probabilities calculated from the base counts within the motif), and discarded candidate seeds with the uncertainty lower than 1.5 bits (out of the maximum of 2 bits 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. Motifs which pass these filters comprise our seed set v_i .

We next build a graph $\mathcal{G}$ modeling the seed motifs v_i as vertices and their associations in $\mathcal{S}$ 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 v_i , now represented as a vertex in our graph, we construct the edges connecting them. For that purpose, we define *pairings* and *distinct pairings* of motifs:

Definition[Motif pairings] If we denote each distinct occurrence of a motif v_i by v_i^k , we define the set of pairings

$$p = \{(v_i^k, v_j^l) | k \in [1, n_i], l \in [1, n_j], i, j \in [1, M], i \neq j, d(v_i^k, v_j^l) \leq w\}$$

where n_i and n_j are the counts of occurrences of v_i and v_j in $\mathcal{S}$, respectively, $d(v_i^k, v_j^l)$ is the distance between motifs v_i^k and v_j^l in the genome, M is the total number of motifs, and w is a pre-defined window size.

Definition[Distinct motif pairings] We form the distinct motif pairings set $\mathcal{DP}$ from $\mathcal{P}$ by replacing all occurrences of (v_i^k, v_j^l) and $(v_i^k, v_j^{l'})$ with a single pair (v_i^k, v_j^l) ($v_j^{l'}$ unifying all $v_j^{l'm}$ such that $(v_i^k, v_j^{l'm}) \in \mathcal{P}$), and all pairings $(v_i^{k_1}, v_j^l)$ and $(v_i^{k_2}, v_j^l)$ with a single pair (v_i^k, v_j^l) . In other words, in $\mathcal{DP}$ each v_i^k connects to one and only one v_j^l , and each v_j^l connects to one and only one v_i^k .

We use the above definitions to construct the edges connecting vertices v_i in $\mathcal{G}$ in the following way. Sliding a window of a pre-defined size w from the beginning of $\mathcal{S}$, we build (and subsequently update) an adjacency list. We have left the window size as a parameter to our software, but our calibration runs have indicated that the best overall results can be achieved for sizes around 1000. This introduces some skepticism about how much our software can be adjusted to identify significantly diverged short elements, such as human ALUs, as the number of associated conserved short motifs within their span might be too small to form a large enough set of significant associations.

As we move the window through $\mathcal{S}$, from one motif v_i to the next v_j , we count the number of occurrences c_{ij} of each $(v_i, v_j) \in \mathcal{DP}$, $i \neq j$, and we assign c_{ij} as the weights of the edges e_{ij} connecting v_i 's and v_j 's. When $i = j$, i.e. when v_i and v_j are two distinct occurrences of the same motif, we do not introduce any self-edges, thus ignoring their co-occurrence.

Upon assigning the weights we retain e_{ij} only if the probability of a random association of v_i and v_j , assuming the uniform distribution of motifs in $\mathcal{S}$, is less than 0.05 (this parameter can be adjusted with respect to the desired tradeoff between sensitivity and specificity). We do that as follows. As before, let n_i and n_j be the numbers of occurrences of motifs v_i and v_j , and let c_{ij} be their observed number of co-occurrences within the intervals of width w . Then, if we denote the total length of $\mathcal{S}$ by L , the expected number of occurrences of v_j in a single interval is $\tau = n_j w/L$. Using the Poisson distribution, we estimate the probability of any interval containing v_i also accommodating one or more v_j 's as $1 - e^{-\tau}$, and the expected number of such intervals as $n_i(1 - e^{-\tau})$. Under the Poisson model the probability of a random variable X representing the chance co-occurrence of v_i and v_j , c_{ij} or more times within the distance of w or less is given by

$$P\{X \geq c_{ij}\} = 1 - \sum_{k=0}^{c_{ij}-1} \frac{[n_i(1 - e^{-\tau})]^k e^{n_i(1 - e^{-\tau})}}{k!}$$

We thus remove from $\mathcal{G}$ all edges for which $P\{X \geq c_{ij}\} \geq 0.05$.

After the seed motif graph $\mathcal{G}$ has been built, we proceed to locate cliques representing groups of motifs v_{ik} 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 approach is based on locating the intersections among adjacency lists of the vertices, through indexing and then using the C++ standard template library (functions `sort()` and `set_intersection()`). Although this method only approximates clique finding, and it can miss quite a few maximal cliques in $\mathcal{G}$, in practice it results in a good number of items of reasonable size, suitable for further work.

The next step involves the mapping of the cliques back to the genome, and we did this using the algorithm for locating constrained heaviest segments. This algorithm, whose early version has been described by Jon Bentley in [5], and later used and modified by several authors, including our own version [24], 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 consensuses. 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 locations 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 elements.

2.2 Determining the consensus sequences

The positions where the original cliques mapped to the genome can be used for the determination of the corresponding consensus sequences. For well conserved copies resulting from recent insertional or other duplication 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 another computationally expensive, but nevertheless often necessary step (which we have made optional, as justified in Section 4 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 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 S_1 and S_2 , of cardinality $D_1 = |S_1|$ and $D_2 = |S_2|$. 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 $|S_1 \cap S_2|$ as an experiment in which we draw D_2 motifs from 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 S_1 and S_2 contains exactly k motifs, i.e. that during the “assembling” of S_2 exactly k motifs came from S_1 . Consequently, if S_1 and S_2 share K motifs, the probability that these two segments would share K or more of them is calculated as

$$P_K = \sum_{k=K}^{D_2} P_k(N, D_1, D_2), \text{ and we can adopt it as a measure of distance between } S_1 \text{ and } S_2.$$

We use this distance as a basis for 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, whereas 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. For this we have interfaced to CLUSTALW [13], and 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 our

assumption was that in most cases 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 these consensus sequences to further mine the genome for similarities not initially discovered by our software, and also to attempt repeat classification, 'N' at any 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. Indeed, 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 [26] have shown that at least in the human genome relative substitution rates for four nucleotides can be arranged in the following relative order: Substitution(G) > Substitution(C) > Substitution(A) > 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 (signifying an insertion in some copies) only if it is present in a clear majority of aligned sequences.

2.3 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 have used our 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 (<http://blast.wustl.edu>), and palindrome and inverted repeat 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 classification strategies: homology, target site duplication (TSD) search and structural search.

When using homology the software attempts to classify the sequence consensus by comparing them to the 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 RepClass 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* 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 have looked at the large number of possible superfamily classifications, and a manuscript detailing the biological aspects of this work is currently in preparation.

3 Results

We have run our software toolkit both on simulated sequences (providing a 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 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 investigating how many of the consensus we build can be classified, and how would these classifications relate to these from manually established genome annotations.

The first tests were performed in the simulated environment which we have manipulated to resemble 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 have inserted 150 copies each of two previously characterized repeats, TIGGER and L1MCA 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, but no insertions or deletions. Since our model builds the tentative repeated elements from short pieces, and does not rely on positional information (at least until the elements have been identified and the consensus is 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 simulations, computing the sensitivity and specificity of locating the elements we have inserted (measured as the percentage of base positions). The parameters of core repeat finding, the initial window size w used to establish the edges e_{ij} of graph $\mathcal{G}$, and the threshold c_{ij} for filtering the edges have been determined by performing simulations with interval width ranging from 500 bases (corresponding to the approximate length of common families of short repeats) to 2000 bases (established somewhat *ad hoc*, as beyond that value we risked finding too many associations), and the threshold values ranging from 0.1 to 0.001, respectively. In a similar way, we have manually examined the segments resulting from the mapping of the cliques back to $\mathcal{S}$, and set the weights assigned to the covered

bases to produce the optimal widths in each iterative run of our software. In order to optimize the consensus building and the subsequent classification, we performed the measurements 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 these simulations is shown in Figure 1. We have varied the clustering threshold, defined in the Section 2 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 1. In our artificial data runs we have also varied the number of separate inserted elements and their copy counts, and the outcomes were consistent.

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

Consequently, in the real genomic data run, we have compared the performance of our algorithm with that of RepeatScout, on human chromosome 21. Since this comparison was mainly targeted towards identifying known and previously characterized repeats, it was easy to calculate the sensitivity and specificity by repeat masking 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 2. 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 3.

The data shown in tables 2 and 3 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 4. 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.

4 Discussion

While the performance of our tool on well conserved, and thus very similar, copies of recent transposon and other duplications 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, although 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. For all but short segments (where looking for repeats would not make much sense, anyway) the full masking done by our program is also computationally expensive, but it performs the core tasks of repeat identification an order of magnitude faster than the other existing tools.

There are indeed 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 consensus and invoking the RepeatMasker with this catalog (this strategy can also 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 which 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 ideally 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 goal 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 consensus would need further work in merging, splitting, trimming or extending, this testifies that the signal we capture is indeed real.

Acknowledgments

This work has been partially supported by NIH grant 5R03LM009033-02 to NS.

Biographical notes

Abanish Singh received his Ph.D. degree in Computer Science and Engineering in 2008 from the University of Texas at Arlington. He also holds an MS degree from the Motilal Nehru National Institute of Technology, Allahabad, India, and between 1993 and 2004 he served on the faculty of the Department of Computer Science and Engineering at the Sant Longowal Institute of Engineering and Technology, Longowal, India. His research interests are in bioinformatics and computational biology, specifically in the genomic sequence analysis and motif discovery.

Umeshkumar Keswani received his Master's degree in Computer Science. from the University of Texas in Arlington in 2008. His thesis research was in high performance computing applied to biology and high energy physics. In 2007, he served as a computer engineer at Brookhaven National Laboratories.

David Levine is on the faculty of Computer Science and Engineering at the University of Texas at Arlington. His research interests include high performance computing, scientific computing, and systems engineering. He is a graduate of the University of Texas at Austin.

Cedric Feschotte received his Ph.D. degree in Biology in 2001 from the University of Paris VI, France. He was a postdoctoral research associate at the Departments of Plant Biology and Genetics at the University of Georgia from 2001 to 2004, where he worked with Dr. Susan Wessler on transposable elements in plant genomes. Since 2004 he is an Assistant Professor in the Department of Biology at the University of Texas at Arlington. His current research focus is on the evolutionary history and genomic impact of mobile genetic elements, with an emphasis on the human genome.

Nikola Stojanovic received his Ph.D. degree in Computer Science and Engineering in 1997 from the Pennsylvania State University, University Park, PA. After five years of working on the Human Genome Project at the Whitehead Institute/MIT Center for Genome Research in Cambridge, MA, he joined the faculty of the University of Texas at Arlington in 2003, as an Assistant Professor. His research interests are in algorithms for genomic sequence analysis, phylogenetic studies and sequence alignments.

References and Notes

1. Aho AV, Corasick MJ. Efficient string matching: An aid to bibliographic search. *Comm. ACM* 1975;18:333–340.
2. Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ. Basic local alignment search tool. *J. Mol. Biol* 1990;215:403–410. [PubMed: 2231712]
3. Arndt PF. The history of nucleotide substitution pattern in human. *Biophysics* 2003;48(Suppl. 1)
4. Bao Z, Eddy SR. Automated de novo identification of repeat sequence families in sequenced genomes. *Genome Res* 2002;12:1269–1276. [PubMed: 12176934]
5. Bentley, Jon. Programming pearls: algorithm design techniques. *Comm. ACM* 1984;27:865–873.
6. Edgar, Robert C.; Myers, Eugene W. PILER: identification and classification of genomic repeats. *Bioinformatics* 2005;21:i152–i158. [PubMed: 15961452]
7. Feschotte, Cedric. Transposable elements and the evolution of regulatory networks. *Nature Rev. Genet.* 2008 Electronic version ahead of print.
8. International Human Genome Sequencing Consortium. Initial sequencing and analysis of the human genome. *Nature* 2001;409:860–921. [PubMed: 11237011]
9. Jaccard, Paul. Distribution de la flore alpine dans le bassin des drouces et dans quelques regions voisines. *Bull. Soc. Vaud. Sci. Nat* 1901;37:241–272.
10. Jurka J. Repbase Update: a database and an electronic journal of repetitive elements. *Trends Genet* 2000;9:418–420. [PubMed: 10973072]
11. Jurka J, Kapitonov VV, Pavlicek A, Klonowski P, Kohany O, Walichiewicz J. Repbase Update, a database of eukaryotic repetitive elements. *Cytogenet. Genome Res* 2005;110:462–467. [PubMed: 16093699]
12. Karp R, Rabin M. Efficient randomized pattern matching algorithms. *IBM J. Res. Devel* 1987;31:249–260.
13. Larkin MA, Blackshields G, Brown NP, Chenna R, McGettigan PA, McWilliam H, Valentin F, Wallace IM, Wilm A, Lopez R, Thompson JD, Gibson TJ, Higgins DG. Clustal W and Clustal X version 2.0. *Bioinformatics* 2007;23:2947–2948. [PubMed: 17846036]

14. Li R, Ye J, Li S, Wang J, Han Y, Ye C, Yang H, Yu J, Wong GK. ReAS: Recovery of ancestral sequences for transposable elements from the unassembled reads of a whole genome shotgun. *PLoS Comput. Biol* 2005;1:e43. [PubMed: 16184192]
15. Parsons JD. Miropeats: graphical DNA sequence comparisons. *Comput. Appl. Biosci* 1995;11:615–619. [PubMed: 8808577]
16. Price A, Jones NC, Pevzner PA. *De novo* identification of repeat families in large genomes. *Bioinformatics* 2005;21:i351–i358. [PubMed: 15961478]
17. Ranganathan, N.; Feschotte, C.; Levine, D. Cluster and grid based classification of transposable elements in eukaryotic genomes. *Proceedings of CCGrid06, 6th IEEE Int. Symp. on Cluster Computing and the Grid*; 2006. p. 45
18. Rice P, Longden I, Bleasby A. EMBOSS: The European Molecular Biology Open Software Suite. *Trends Genet* 2000;16(6):276–277. [PubMed: 10827456]
19. Saha S, Bridges S, Magbanua ZV, Peterson DG. Empirical comparison of ab initio repeatfinding programs. *Nucleic Acids Res* 2008;36:2284–2294. [PubMed: 18287116]
20. Singh A, Feschotte C, Stojanovic N. A study of the repetitive structure and distribution of short motifs in human genomic sequences. *Int. J. Bioinformatics Research and Applications* 2007;3:523–535.
21. Singh, Abanish; Feschotte, Cedric; Stojanovic, Nikola. Micro-repetitive structure of genomic sequences and the identification of ancient repeat elements. *Proceedings of BIBM 2007, The IEEE International Conference on Bioinformatics and Biomedicine*; 2007. p. 165-171.
22. Singh, Abanish; Stojanovic, Nikola. An algorithm for finding substantially broken repeated sequences in newly sequenced genomes. *Proceedings of ICMB 2007, International Conference on Mathematical Biology*; Melville, New York: American Institute of Physics; 2008. p. 79-88.
23. Smith TF, Waterman MS. Identification of common molecular subsequences. *J. Mol. Biol* 1981;147:195–197. [PubMed: 7265238]
24. Stojanovic, Nikola. An improved algorithm for the location of heaviest segments in genomic sequences. *Proceedings of the 2009 International Joint Conferences on System Biology, Bioinformatics and Intelligent Computing, IJCBS09*; 2009. To appear
25. Volfovsky N, Haas BJ, Salzberg SL. A clustering method for repeat analysis in DNA sequences. *Genome Biol* 2001;2(RESEARCH0027)
26. Zhang EF, Gerstein M. Patterns of nucleotide substitution, insertion and deletion in the human genome inferred from pseudogenes. *Nucleic Acids Res* 2003;31:5338–5348. [PubMed: 12954770]

Figure 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.

The performance of our software on a simulated data set, in one representative experiment. “Sn” stands for sensitivity and “Sp” for specificity.

Table 1

Degradation	PILER Sn	PILER Sp	RepeatScout Sn	RepeatScout Sp	Our Sn	Our 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 2

Performance comparison of our algorithm with RepeatScout in identifying known and previously characterized repeats in human chromosome 21.

	Our Algorithm Sensitivity	Our Algorithm Specificity	RepeatScout Sensitivity	RepeatScout Specificity
Experiment 1	0.5230201	0.8714505	0.4466029	0.9938296
Experiment 2	0.5752110	0.8680427	0.6082956	0.9840795

Table 3

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

Table 4

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