

## Spectrum-Based *De Novo* Repeat Detection in Genomic Sequences

HUY HOANG DO,<sup>1</sup> KWOK PUI CHOI,<sup>2,3</sup> FRANCO P. PREPARATA,<sup>4</sup>  
WING KIN SUNG,<sup>1</sup> and LOUXIN ZHANG<sup>3</sup>

### ABSTRACT

A novel approach to the detection of genomic repeats is presented in this paper. The technique, dubbed SAGRI (Spectrum Assisted Genomic Repeat Identifier), is based on the spectrum (set of sequence  $k$ -mers, for some  $k$ ) of the genomic sequence. Specifically, the genome is scanned twice. The first scan (FindHit) detects candidate pairs of repeat-segments, by effectively reconstructing portions of the Euler path of the  $(k-1)$ -mer graph of the genome only in correspondence with likely repeat sites. This process produces candidate repeat pairs, for which the location of the leftmost term is unknown. Candidate pairs are then subjected to validation in a second scan, in which the genome is labelled for hits in the (much smaller) spectrum of the repeat candidates: high hit density is taken as evidence of the location of the first segment of a repeat, and the pair of segments is then certified by pairwise alignment. The design parameters of the technique are selected on the basis of a careful probabilistic analysis (based on random sequences). SAGRI is compared with three leading repeat-finding tools on both synthetic and natural DNA sequences, and found to be uniformly superior in versatility (ability to detect repeats of different lengths) and accuracy (the central goal of repeat finding), while being quite competitive in speed. An executable program can be downloaded at <http://sagri.comp.nus.edu.sg>.

**Key words:** genomic repeat, Hamming metric, Levenshtein metric, repeat finding, sequence spectrum.

### 1. INTRODUCTION

**R**EPEATS ARE UBIQUITOUS in large eukaryotic genomes (Charlesworth et al., 1994; Human Genome Sequencing Consortium, 2001; Arabidopsis Genome Initiative, 2000). Genome sequencing indicates that a large proportion of a vertebrate genome is composed of five major classes of repeat sequences: short and long interspersed elements, inactive retroposed copies of cellular genes (usually referred as processed pseudogenes), simple short sequence repeats such as  $A^n$  or  $(CGG)^n$ , segmental duplications, and tandem repeats. They complicate genome assembly, genome comparison, and the study of genome

---

<sup>1</sup>Department of Computer Science, National University of Singapore, Singapore.

<sup>2</sup>Department of Statistics and Applied Probability, National University of Singapore, Singapore.

<sup>3</sup>Department of Mathematics, National University of Singapore, Singapore.

<sup>4</sup>Department of Computer Science, Brown University, Providence, Rhode Island.

rearrangement. They also increase the difficulty in probe signal analysis for microarray (that is, the probe signal may be inaccurate if the probe sequence overlaps with repeat regions). Biologically, the distribution of repeats is highly correlated with chromatin structure, which influences the activity of genes. They are also believed to play significant roles in genome evolution and proneness to disease (Bowen and Jordan, 2002; Medstrand et al., 2005). For example, Pelizaeus-Merzbacher disease (PMD) commonly arises from genomic duplications of the dosage-sensitive proteolipid protein gene. Hence, repeat identification has attracted considerable attention and many algorithms have been designed to identify the repeats in a genome.

Traditionally, repeats are found by scanning the DNA sequence with some known repeat patterns. RepeatMasker (<http://repeatmasker.genome.washington.edu>) is the most prominent example in this class. It uses a dictionary of known repeat sequences, for example, RepBase database (Jurka, 2000), and performs an exact or approximate string matching of the given sequence against all the dictionary entries. The comparison between the database entries and the query sequence is performed by the program `cross_match`, an efficient implementation of the Smith-Waterman-Gotoh algorithm developed by Smit and Green. MaskerAid (Bedell et al., 2000) is an accelerated version of RepeatMasker, where `cross_match` is replaced with a wrapper for WU-BLAST. Availability of a repeat library is crucial to this approach. Note that a repeat library is not universal and has to be manually compiled for new genomes.

When the repeat library is not available, we need to discover repeats without prior knowledge. *De novo* repeat discovery consists of two steps. Step (1) is the identification of repeats. It locates all pairs of substrings of certain minimum length  $\ell$  in the DNA sequence such that the members of a pair are within some prescribed editing distance  $d$ . Step (2) is the classification of repeats. It classifies the repeats into different classes depending on the pattern.

For Step (2), a number of solutions are proposed including RepeatFinder (Volfovsky et al., 2001), RECON (Bao and Eddy, 2002), RepeatGluer (Pevzner et al., 2004), and PILER (Edgar and Myers, 2005). For Step (1), where we just want to identify exact repeats (that is  $d = 0$ ), a linear time solution is available based on the use of suffix tree (Kurtz and Schleiermacher, 1999). This solution is scalable to whole genome level.

With regards to approximate repeats (that is  $d > 0$ ), previous results are summarized in Kurtz et al. (2000). Here, we survey some recent results. The algorithm by Schmidt (1998) finds all maximal approximate repeats in  $O(n^2 \log n)$  time and  $O(n^2)$  space. Sagot (1998) designed an algorithm for finding approximate repeats of length  $p$  and with  $d$  errors using  $O(np^d |\mathcal{A}|^d)$  time where  $|\mathcal{A}|$  is the alphabet size. Kurtz et al. (2000) produced software, called REPuter, for finding all approximate repeats in a DNA sequence of length  $n$  with certain minimum length,  $\ell$ , and certain maximum number of errors,  $d$ . The running time of their algorithm is  $O(n + zd^3)$ , where  $E[z] = O(n^2/4^{\ell/(d+1)})$  (here random variable  $z$  denotes the number of seeds for performing extensions in the REPuter algorithm). Though REPuter (Kurtz et al., 2000) can handle errors, it can only find long repeats with a small number of errors. Adebisi et al. (2001) have suggested another solution which can find short repeats with a few errors. However, the running time is about  $O(n^{1.7} \log n)$ , which may not be scalable to genome-level searches.

All the above solutions can only handle a constant number of errors  $d$ . However, the number of errors will obviously increase as the length of the repeat increases. Recently, Edgar and Myers (2005) developed a software package, called PALS (Pairwise Alignment of Long Sequences), which allows us to find all repeats of certain minimum length  $\ell$  containing at most  $\mu\%$  errors. They utilize a filtering method to speed up the running time. However, when the error rate  $\mu$  increases, PALS will get less and less efficient. At about the same time, Zhang and Waterman (2005) proposed to use the De Bruijn graph to find repeats, based on the observation that most of the repeats appear as heavy Eulerian paths in the De Bruijn graph of the DNA sequence. Hence, repeats can be identified from the analysis of heavy Eulerian paths. The running time of their algorithm is approximately linear in the genome size. However, their method cannot guarantee to find all repeats of certain minimum length  $\ell$  containing at most  $\mu\%$  errors.

## 2. MOTIVATION AND OUTLINE OF OUR APPROACH (SAGRI)

A *repeat* is naturally construed as the product of a faithful (possibly multiple) copy of a sequence segment, followed by independent mutational drift of each individual segment. As is customary, a genomic

sequence is viewed as being read from left to right. Thus detection of a repeat involves the establishment of a relationship of “similarity” between the text being scanned and some text that occurs to its left (the scanned prefix). Provided that the editing noise is not extreme (in which case, the very notion of repeat becomes questionable), the two corresponding segments will share identical substrings of a prescribed length  $k$  (the principle of “filtration”), which are a subset of the set of all  $k$ -mers of said prefix. Such set is the *spectrum* of the prefix (denoted  $\mathcal{S}_m$  if  $m + k - 1$  is the length of the prefix).

Our approach SAGRI (Spectrum Assisted Genomic Repeat Identifier) is based on the use of the spectrum. Such choice may at first appear somewhat surprising. Indeed, the spectrum—as a set—is used in the model of sequencing-by-hybridization because microarrays provide no information as to the position or multiplicity of any  $k$ -mer. Such information is available here but is (surprisingly) discarded. The alternative is the use of an index, where for each  $k$ -mer we store the positions where it occurs. The intuition behind preferring the spectrum over the index is that only a small fraction of the index information is presumably relevant to repeat finding (a  $k$ -mer may occur in a large number of positions completely irrelevant to the repeat), and that the consensus of the repeat  $k$ -mers provides a much simpler, and quite effective solution. By analogy with previous work in Sequencing-by-Hybridization (Preparata et al., 2005), a  $k$ -mer, queried in the spectrum, will be frequently referred to as a *probe*.

In fact, the “discovery” function of our algorithm is the reconstruction of a (putative) Euler path of a homologous segment occurring to the left of the current segment. If the reconstructed path is judged acceptable, the discovered pair of segments is output as a candidate repeat-pair; each  $k$ -mer of the reconstructed path is appropriately labeled with unique identifiers linking the putative “left” segment to the “right” segment being scanned: this linkage is critical.

Indeed, the “validation” function of SAGRI vets the collection of candidate repeat-pairs using the spectrum of the “left” segments of repeat pairs (whose  $k$ -mers are now labelled as follows: each left-segment  $k$ -mer is labelled with an integer denoting the repeat-pair, the location of the corresponding right-segment  $k$ -mer and a bit denoting match/mismatch). This process is carried out by a scan which tags the entire genomic sequence against this restricted spectrum: pairwise repeats will yield a single label, whereas multiple repeats will yield sets of label. Identically labeled tags corresponding to matches will form consecutive runs; these runs identify “diagonal segments” in a hypothetical dot-diagrams pertaining to the two segments of the pair. The validation is therefore completed by aligning corresponding inter-run portions of the two segments and will produce repeat pairs consisting of *actual* genome substrings.

This preliminary discussion should substantiate the somewhat counter-intuitive claim about the adequacy of the spectrum. In fact, one can argue that the spectrum, with its minimal overhead and the robustness deriving from the consensus of “distributed filtration,” is a most appropriate device for genomic repeat identification.

The spectrum is used by SAGRI both to detect repeat events and to localize the region of the leftmost term of a repeat pair (in this case, the search is sharper because we deal with the much smaller spectrum of the candidate repeats). Most revealing is the robustness of spectra in the execution of the described functions (see Section 3.2 for a quantitative substantiation of this assertion).

The adoption of spectra determines the running time of SAGRI. In fact, each of the two scans of the genome runs in time essentially proportional to the genome length. The time of the validation phase is a function of the productive output (reported repeats), so that the technique is output-sensitive. Assuming moderate editing noise, it is easy to see that this additional running time is proportional to the total length of the repeats.

Section 3 describes the algorithms and Section 4 reports the selection of the algorithmic parameters, based on the detailed analysis contained in Appendix A (optional reading). Section 5 reports extensive experimentation on both synthetic and natural DNA sequences, substantiating the versatility, accuracy (sensitivity and specificity), and efficiency of the proposed technique, and Section 6 summarizes the paper and outlines possible extensions.

### 3. SPECTRUM-BASED REPEAT FINDING ALGORITHMS

A *repeat* is a sequence of typically disjoint substrings  $L_1, L_2, \dots, L_r$  of the genomic sequence, where  $L_i$  conventionally occurs to the left of  $L_{i+1}$ . A repeat is *simple* if  $r = 2$  and *multiple* otherwise.

Repeats can be classified in a nested hierarchy

$$R1 \subset R2 \subset R3 \subset R4$$

of increasing complexity (and difficulty), as follows:

- R1: exact simple repeats;
- R2: simple repeats differing only by substitutions (Hamming repeats);
- R3: simple repeats differing by substitutions, insertions, and deletions (Levenshtein repeats);
- R4: Multiple repeats, each consecutive pair being a Levenshtein repeat.

The selection of the algorithm parameters will be discussed in Section 4. The fundamental parameter is  $k$ , the length of the  $k$ -mers (probes). The symbol positions of the sequence are naturally numbered  $1, \dots, N$  from left to right. A  $k$ -mer is said to occur at position  $i$  (of the genomic sequence) if  $i$  is the leftmost position of the  $k$ -mer.

Here and hereafter, we shall consider simple repeats  $(L_1, L_2)$ , with segment  $L_1$  occurring to the left of segment  $L_2$  in the genome. Since  $L_1$  is the segment to be detected while the algorithm scans  $L_2$ , segment  $L_2$  will be denoted the *reference*.

The algorithm consists of two separate phases, (i) the *FindHit phase* and (ii) the *Validation phase*. In the FindHit phase, the algorithm runs in two alternating modes: the *advancing mode* and the *detection mode*. (Details of this procedure will be discussed in Section 3.1.) When the FindHit phase has been completed (the entire genome sequence has been processed), a collection of candidate repeats are output. This output forms the input for the Validation phase, which verifies if the candidate repeats are actual repeats. See Section 3.2 for details.

For presentational purposes, we shall discuss jointly cases R2 and R3, with emphasis on R2, since R3 is a relatively simple modification of R2 (R1, on the other hand, is a trivial specialization of R2). Case R4 will be discussed in the context of the Validation phase (Section 3.2).

### 3.1. The FindHit phase

In this phase, the algorithm scans the sequence from position 1 to position  $N - k + 1$  and operates in two distinct modes, the *advancing mode* and the *detection mode*, alternating between the two as and when appropriate. The sequence being scanned is called the “reference path,” while any other path constructed on the basis of the spectrum is simply referred to as a “path.” The algorithm initially starts in the advancing mode, and remains in it until the current spectrum is found to contain a probe matching the current  $k$ -mer (denoted, for short, as  $k$ -match). Upon this event, it switches to the detection mode. In the detection mode, it verifies whether the spectrum supports the construction of paths which agree with the reference path within some prescribed deviation, and terminates any path which exceeds the specified deviation. Details of this procedure are discussed in the context of the DetectRepSeg procedure below. Any path being terminated is potentially a candidate repeat, and is passed on to subsequent scrutiny by the validation procedure, only if its length exceeds a certain threshold since an artifact of randomness could be mistaken for a short repeat. The algorithm reverts to the advancing mode when no path subsists. The criteria for path termination and for acceptance of candidate repeats are based on discriminating between repeats and naturally occurring events in random sequences, which will be discussed in Appendix A.

The criterion for the transition from advancing to detection mode indicates that an actual repeat can be detected only if the pair  $(L_1, L_2)$  contains at least a  $k$ -match; after detection of the first  $k$ -match in the left-to-right scan, actual determination of the detected repeat will be effected in the Validation phase (Section 3.2).

We now give a high-level pseudo-code description of the algorithm. The crucial component of the algorithm is the call of procedure *DetectRepSeg* (Line 5 of Algorithm 1) to be discussed next.

**3.1.1. Procedure *DetectRepSeg*( $\mathcal{S}_p, p$ ).** Recall that the spectrum,  $\mathcal{S}_m$ , is the collection of all the  $k$ -mers of the genomic sequence from positions 1 to  $m + k - 1$ . Upon the detection of a  $k$ -match at position  $p$ , the algorithm implicitly switches from the advancing mode to the detection mode (Line 5 of Algorithm 1). It calls upon the function, *DetectRepSeg*( $\mathcal{S}_p, p$ ), to verify whether the spectrum  $\mathcal{S}_p$  supports the construction of paths which agree with the reference path within some prescribed deviation. We reconstruct the putative

**Algorithm 1** Spectrum-Assisted-Genomic-Repeat-Identifier (SAGRI), *FindHit()*


---

```

1: mode ← advancing
2: i ← 1
3: while  $i \leq N - k + 1$  do
4:   if  $k$ -match detected in  $\mathcal{S}_i$  then
5:     (repeat_len, repeat) ← DetectRepSeq( $\mathcal{S}_i$ , i)
6:     if repeat_len > threshold then
7:       Report potential repeat
8:       mode ← skip
9:     i ← i + 1
10:  update spectrum  $\mathcal{S}_i$ 

```

---

repeat path  $Curr[1..]$  by comparing the genomic sequence starting at position  $p$ , called now  $Ref[1..]$  with the spectrum  $\mathcal{S}_p$ .

For reasons to be discussed below, we have chosen to carry out the exploration of the competing putative paths by a suitable modification of *depth-first-search* (DFS).

The high-level pseudo-code is given below (Algorithm 2). The structure constructed by the algorithm is dubbed *budded path*. It is basically a rooted tree, consisting of a path, whose leaves (dubbed “buds”) are children of path nodes. The *level* is the distance from the root.

The event type (a node of a putative path) handled by the procedure is a quadruple  $(i, j, w; type)$ . Indices  $i$  and  $j$  indicate that  $Curr[1..i]$  represents the current putative repeat aligned with the reference segment  $Ref[1..j]$ . The handling of Hamming repeats ( $i = j$ ) is covered by lines 15–22. (For Levenshtein repeats,  $i$  and  $j$  may be different.) Item  $w$  is the last  $k$ -mer of the putative path and  $type \in \{match, mismatch, insertion, deletion\}$ . Each node of a budded path is scored: in the simplest case (Hamming repeats) the score is the number of accumulated mismatches.

Path  $Curr[1..i]$  is referred to as the *match-path*: its score is minimum among the scores of all buds of the budded path. To extend  $Curr[1..i]$  one position, we concatenate its  $(k - 1)$ -suffix with  $X \in \{A, C, G, T\}$  to form  $w = Curr[i - k + 2..i]X$ . If  $w \in \mathcal{S}_p$ , then we test the relation of  $w$  to the  $k$ -suffix of  $Ref[1..j]$  by exploring in succession the cases (*types*): deletion, insertion, mismatch, match (note that if a match exists, it will be placed at the top of the STACK). Each spectrum-supported  $w$ , whose score does not exceed a chosen threshold, generates a record of the form  $(i, j, w; type)$  to be pushed onto a STACK for possible future processing. Each record in the STACK identifies a tree bud. If all the potential buds generated at the current step have scores exceeding the threshold, then the algorithm backtracks to the record obtained by popping the stack.

We illustrate the procedure by an example in Figure 1. Suppose the genomic sequence is ACGAAGTGATTAACCCCTCGACGCGATCC (repeat segments are underscored and the mismatches are in bold-face), we use  $k = 3$  and the threshold is 3 (i.e., at most two mismatches are tolerated). At  $p = 18$ , probe CGA is supported by  $\mathcal{S}_{18}$  and the algorithm switches to the “detection mode.” There are two feasible buds, T and A, both mismatches (score 1). Suppose the algorithm pursues the extension of bud GAT. The spectrum supports ATT (and no other ATX), increasing the score to 2; further extension to TTA attains score 3 and the path is terminated. At this point, the algorithm reverts to bud GAA (score 1), from which two new buds are supported by the spectrum AAG: (match, score 1) and AAC (mismatch, score 2). Further pursuit of bud G (i.e., probe AAG) leads to the detection of the path CGAAGTGAT and further extension achieves a score exceeding the threshold, thus terminating the path. Depth-first policy requires the extension of AAC and ACG. However, both lead to early termination, so that (CGAAGTGAT, CGACGCCGAT) is reported as a candidate repeat.

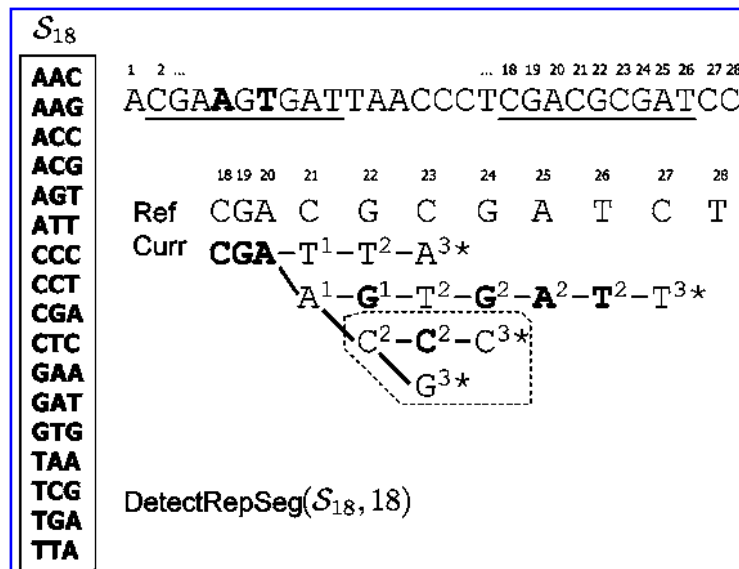
To detect Levenshtein repeats, the only modification needed is that in the detect-repeat-segment procedure as many as 9 buds may be generated at each stage. We allow up to 4 buds to take care of a possible deletion at the reference path, and 1 bud for a possible insertion at the reference path. We penalize insertion/deletion more than a mismatch. Specifically, we give an error score of three for an insertion/deletion bud, whereas only a score of one for a mismatch bud. Furthermore, we extend the buds in this order: match, mismatch, insertion/deletion (see pseudocode of *DetectRepSeg*).

**Algorithm 2** *DetectRepSeg*( $S_p, p$ )

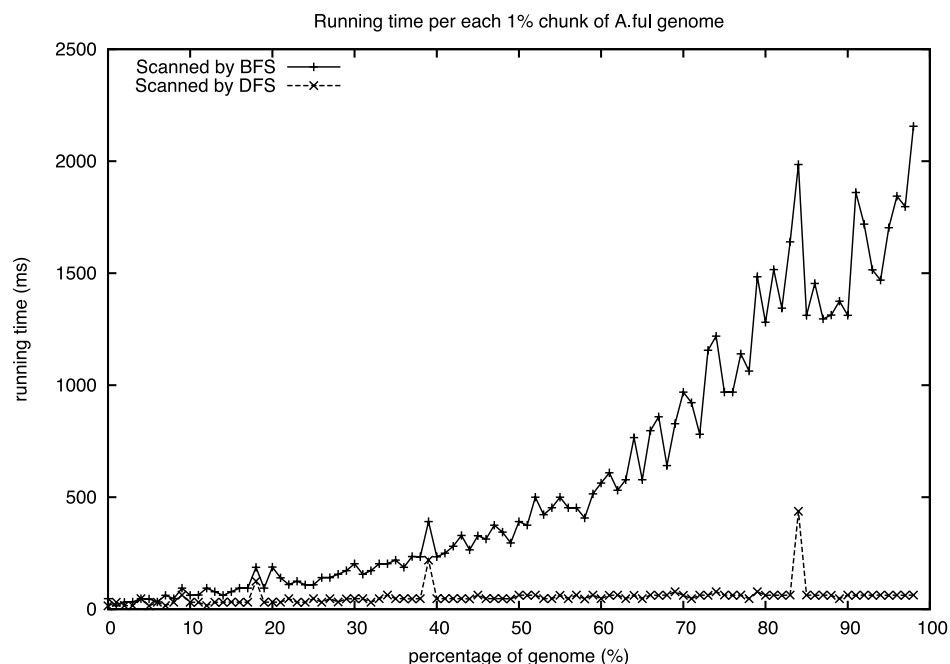
```

1:  $STACK \leftarrow (k, 1, Ref[1..k]; match)$ ;
2: while  $STACK$  is not empty do
3:    $(i, j, w; type) \leftarrow STACK$ 
4:    $Curr[i - k + 1..i] \leftarrow w$ 
5:   update score of  $Curr[i - k + 1..i]$ 
6:   if score  $\geq$  threshold then
7:     break;
8:   {-Deletion case-}
9:   for all  $X \in \{A, C, G, T\}$  do
10:     $u \leftarrow Curr[i - k + 2..i]X$ 
11:    if  $u \in S$  then
12:       $STACK \leftarrow (i, j + 1, u; deletion)$ 
13:   {-Insertion case-}
14:    $u \leftarrow Curr[i - k + 1..i]$ 
15:   if  $u \in S$  then
16:      $STACK \leftarrow (i + 1, j, u; insertion)$ 
17:   {-Mismatch cases-}
18:   for all  $X \in \{A, C, G, T\}$  do
19:     $u \leftarrow Curr[i - k + 2..i]X$ 
20:    if  $u \in S$  and  $X \neq Ref[i + 1]$  then
21:       $STACK \leftarrow (i + 1, j + 1, u; mismatch)$ 
22:   {-Match case-}
23:    $u \leftarrow Curr[i - k + 2..i]Ref[i + 1]$ 
24:   if  $u \in S$  then
25:      $STACK \leftarrow (i + 1, j + 1, u; match)$ 
26: return  $\max\{j\}$ 

```



**FIG. 1.** Detect repeat at position 18. The superscript indicates the score (number of mismatches), symbol “\*” indicates path termination due to mismatches, a bold-face character means a match. The portion within the broken-line enclosure will not be constructed under the bounded backtrack policy.



**FIG. 2.** Running time to complete the scan of 1% of the genome of *Archaeoglobus fulgidus* (2.1 Mb) as a function of the position (as a percentage of the sequence length) for BSF and DFS.

To improve the running time, we adopt the following significant heuristic for pruning (i.e., not extending) remote buds. Rather than complying with the standard depth-first policy, in the *FindHit* phase backtracking to unextended buds is restricted as follows:

- Buds whose depth is lower than that of the path being extended by some value  $h$  will not be extended ( $h$  is referred to as the *cut-off*).

For example, by choosing  $h = 4$ , the tree portion in the broken-line enclosure in Figure 1 will not be constructed.

The rationale for this heuristic (to be analyzed in Appendix A) is the intuition that a bud lagging substantially behind the match-path is not likely to be competitive. Such policy, however, may occasionally cause premature abortions of correct detections due to substantial editing noise. This shortcoming may be alleviated by the provision of adopting an affine, rather than constant, cut-off of the form  $h = h_0 + cd$ , where  $c$  is a design parameter and  $d$  is the length of the current match-path. We notice, however, that a match path is reported for validation if its length reaches a predetermined threshold  $H$  (typically,  $H = 50$ ). In fact, editing noise occurring while  $d \leq 50$  can cause false positives.

Qualitatively, with breadth-first-search (BFS), each of the spurious paths (those that accumulate a critical number of mismatches) is *explicitly* terminated upon reaching its end; with our modified depth-first search, however, spurious paths are *implicitly* terminated (abortively) by the just discussed heuristic. Since the time-consuming extension of spurious paths is avoided, we expect much higher efficiency by deploying our modified depth-first-search. The diagram of Figure 2, which illustrates the running time used to complete the scan of one percentage point of the genome length as a function of the position (also expressed as a percentage of the genome length), strikingly confirms our intuition.

### 3.2. Validation phase

The FindHit phase of the algorithm produces a set of candidate pairs of the form  $(L'_1, L'_2)$ , where  $L'_2$  is an actual substring of the genomic sequence, whereas  $L'_1$  is in reality an algorithmic construct, which may or may not coincide with a substring of the genome. In general, it does not, although it is expected to closely match a substring of the genome occurring to the left of  $L'_2$  interacting with it.

The Validation phase consists in fact of two distinct functions, concurrently executed: Localization and Alignment.

1. The purpose of Localization is to locate, along the genome, each individual segment which may have given rise to (most of) the spectrum of  $L'_1$  (the presumed region of occurrence of  $L'_1$ ).
2. The purpose of Alignment is to carry out a two-sequence alignment of the known region of  $L'_2$  with the presumed region of  $L'_1$  as found by Localization to yield a best-score aligned pair  $(L_1, L_2)$ . If the score of  $(L_1, L_2)$  exceeds a suitable threshold, then  $(L_1, L_2)$  is reported as a validated pair, else it is discarded.

Localization is the crucial part of the procedure, since standard tools (based on Dynamic Programming) exist or can be adapted to implement Alignment.<sup>1</sup>

To implement Localization algorithmically we propose the following technique, which is also based on the use of a spectrum.

$L'_2$ -type segments are numbered progressively according to their occurrence along the genome from left to right; each  $k$ -mer of an  $L'_1$ -type segment is associated, during the FindHit phase, with the number  $n(L'_2)$  of the corresponding  $L'_2$  segment.

We call  $\mathcal{L}_1$  the collection of the segments of type  $L'_1$  produced by the FindHit algorithm. From  $\mathcal{L}_1$  we create the multiset of  $k$ -mers  $\Gamma = \cup_{L'_1 \in \mathcal{L}_1} \{w | w \text{ is a } k\text{-mer of } L'_1\}$ . (This multiset is the spectrum of  $\mathcal{L}_1$  as a collection of sequences.) We store  $\Gamma$  in a hash-table  $\mathcal{H}$ , where each record consists of a  $k$ -mer (the key) and the number of an associated  $L'_2$  segment. Note that a given  $k$ -mer could be associated with several  $L'_2$ -type segments.

After this pre-processing of  $\mathcal{L}_1$ , we begin the left-to-right scan of the genome. We score a hit in position  $j$  if the  $k$ -mer in this position coincides with a  $k$ -mer stored in  $\mathcal{H}$ . The scan is carried out by maintaining a moving window (of fixed width  $K$ ) and keeping a count  $\nu(i)$  of the hits intercepted in the interval  $[i - K + 1, i]$  of the genome, and for each hit we retrieve the number of an  $L'_2$ -type segment. The count  $\nu(i)$  is used as circumstantial evidence of  $\mathcal{L}_1$ -hot-spots; if a hot-spot is detected, the segment numbers of the hot-spot  $k$ -mers are used to identify the  $L'_2$ -type segments that are candidates for “possible association”; these candidates are recorded and among them the closest to the right of the current  $L'_1$ -type region (hot-spot) is selected. The detected  $(L'_1, L'_2)$  pair is declared a “possible association” and is passed on to an Alignment Algorithm to certify their classification as an “approximate repeat.”

Although the process involved in this scan appears reasonable for the detection of single (i.e., two-segment) repeats, it may result inadequate for multiple repeats. In fact it rests on the condition of possible association of just consecutive repeats, while two consecutive terms of a multiple repeat may not exhibit sufficient similarity. To obviate this shortcoming, we undertake the following remedial action:

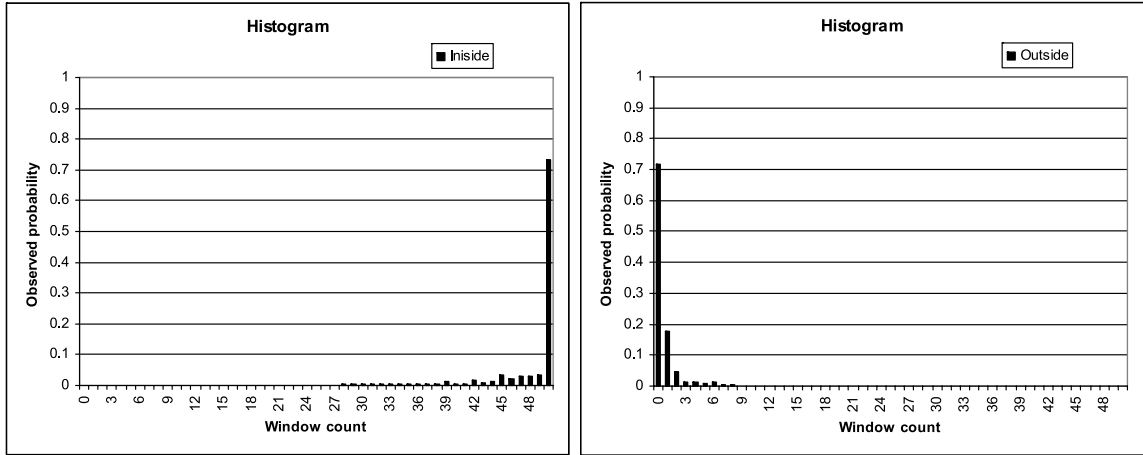
- For each candidate  $L'_2$ -type segment which has remained unprocessed, we retrieve an  $L'_1$ -type segment with which it could realize a possible association, and pass the pair to the Alignment Algorithm.

To illustrate the discriminating effectiveness of the parameter  $\nu(i)$  we ran the following experiment whose results are illustrated in Figure 2. In synthetic sequences of length 100,000 we implanted 100 repeat-pairs of length 100 with 10% Hamming noise. For each sequence we constructed the sequence of the hits for the localization of  $L'_1$ -segments. With a window of size  $K = 50$  we constructed the diagrams of the (normalized) frequencies of windows with  $\nu = j$  fully inside the (implanted)  $L_1$ -intervals (left diagram in Fig. 3) and fully outside both  $L_1$ - and  $L_2$ -intervals (right diagram in Fig. 3). These diagrams clearly substantiate the claim.

Thus, we use the function  $\nu(i)$  to infer the extremes of the interval of positions of  $L'_1$ -segments.<sup>2</sup> Once the putative region of  $L'_1$  has been located, the associated  $L'_2$  is determined as the most frequent value of  $n(L'_2)$  exhibited by the  $k$ -mers of  $L'_1$ , and the alignment action is triggered.

<sup>1</sup>Given the pair  $(L'_1, L'_2)$  one may determine first by Dynamic Programming a pair  $(L''_1, L''_2)$  realizing the best local alignment. This pair may be next extended at both ends in a BLAST-like fashion.

<sup>2</sup>If an  $L'_1$  segment coincides with an actual genomic substring, the parameter  $\nu$ , as a function of the position, would increase in a nominal linear fashion from 0 to  $K$  over  $K$  positions, remain at value  $K$  for  $|L'_1| - K$  positions, and decrease linearly to 0. In reality, since  $L'_1$  is spectrum-generated, we reasonably expect little deviations from such behavior.



**FIG. 3.** Histograms of the normalized frequencies of width-50 windows with  $i$  hits fully inside and outside  $L_1$ -type intervals.

Next, we give a high-level pseudocode of the algorithm (Algorithm 3):

---

**Algorithm 3** *Validate*( $\mathcal{L}_1$ )

---

```

1: while  $j \leq N - k$  do
2:    $w \leftarrow$  k-mer starting at  $j$ 
3:   if  $w \in \mathcal{H}$  then
4:     retrieve numbers  $n(L'_2)$ 
5:      $v(j) \leftarrow 1$ 
6:     if hot-spot is detected then
7:        $L_1^* \leftarrow$  hot-spot
8:       filter  $n(L'_2)$  {segment numbers for possible association}
9:        $n(L_2^*) = \min\{n(L'_2)\}$ 
10:      Align( $L_1^*, L_2^*$ )
11:      for all unprocessed  $n(L'_2)$  do
12:        Align( $L_1^*, L'_2$ ) {remedial action}

```

---

#### 4. SELECTION OF SAGRI PARAMETERS

Before tackling the evaluation of SAGRI against its leading competitors (see next section), in this section we illustrate the selection of the crucial operational parameters of the systems. Such selections are based on the detailed theoretical analysis (based on random sequences) carried out in Appendix A.

Given a DNA sequence (natural or synthetic) of length  $N$ , the most significant parameter is the length  $k$  of the probe used in the FindHit phase, which is selected as  $\lceil \log_4 N \rceil + 2$ . This choice appears adequate to guarantee a satisfactory discrimination of actual repeats from artifacts of randomness (Subsection A.1), and simultaneously protects the execution both from missing legitimate repeats (false negatives, Subsection A.2) and from accepting false positives (Subsection A.4).

In the extension mode of the FindHit scan a current match-path is terminated when the score due to accumulated mismatches exceeds a predetermined threshold. This threshold  $\tau(r)$ , referred to substitution errors, is a slowly growing function of the length  $r$  of the current match-path (Subsection A.3).

Finally, in our experiments, we select as  $H = 50$  the length of the shortest accepted repeats (although  $H$  can be judiciously reduced). The backtrack depth  $h$  of the modified DFS (a key feature of FindHit procedure) is selected as  $\lceil \log_4 L + 1 \rceil$ , where  $L$  is the (known or estimated) total length of the repeat-segments (Subsection A.5).

## 5. EXPERIMENTAL RESULTS

In this section we report on a comparative evaluation of the performance of SAGRI and three leading repeat-finding algorithms, namely, REPuter (Kurtz et al., 2000), PILER (PALS) (Edgar and Myers, 2005), and EulerAlign (EuAl) (Zhang and Waterman, 2005). The benchmarks selected for repeat identification belong to two distinct categories:

- (i) randomly generated sequences (referred to as “synthetic sequences”), discussed in detail in Subsection 5.1; and
- (ii) natural genomic data, discussed in Subsection 5.2.

For synthetic sequences, we introduce two measures to characterize the sensitivity of each algorithm. One measure expresses the number of repeats found, and the other the total number of bases of these repeats. Since in this case the repeats are known *a priori*, we can naturally benchmark the number of repeats found and the total number of repeat bases against those of the implanted repeats (see Subsection 5.1). Dealing with natural genomic data, however, since the repeat set is *a priori* unknown, it will be appropriate to compare different algorithms only in terms of number of repeat bases (see Subsection 5.2).

We have obtained from their originators executable versions of the competing algorithms. The four algorithms have been run on the same platform.<sup>3</sup>

### 5.1. Experiments with random synthetic sequences

The procedure given below, which generates random genomic sequences, mimics our modeling of duplication repeats in a genome as analyzed in Appendix A.

1. We generate a background synthetic genomic sequence of chosen length  $n$  with i.i.d symbols  $\{A, C, G, T\}$ .
2. Reference repeats are obtained by first generating  $m$  short segments, whose lengths are sampled from a Poisson distribution with mean value  $\ell$ , as in Step 1. For each segment, we sample without replacement  $\lfloor \ell/10 \rfloor$  of its positions and create its companion segment by substitutions in those positions. This creates  $m$  pairs of simple repeats with approximately 10% error rate. These repeats form the reference repeats for evaluating the performance of the various algorithms.
3. Repeats are implanted in the background genomic sequence by choosing  $2m$  positions at random without replacement from the background genomic sequence (generated in step 1) and inserting the  $m$  pairs of repeat segments generated in step 2. The resulting sequence forms our random synthetic genomic sequence.

We introduce two measures, *count ratio* ( $CR$ ) and *shared region ratio* ( $SRR$ ), to evaluate the sensitivity of an algorithm. We shall adopt the convention that a repeat is considered to be found by an algorithm if at least 50% of this repeat overlaps with a repeat reported by the algorithm:

- *Count ratio* of an algorithm  $\mathcal{G}$  is defined as the fraction of the reference repeats found by  $\mathcal{G}$ ;
- *Shared region ratio* of  $\mathcal{G}$  is the ratio of the total number of bases of the reference repeat pairs and those reported by  $\mathcal{G}$ .

We compute  $CR(\mathcal{G})$  and  $SRR(\mathcal{G})$  for each random genomic sequence, repeat this procedure 50 times, and calculate the averages of  $CR(\mathcal{G})$  and  $SRR(\mathcal{G})$ .

Table 1 reports the average  $CR$  and  $SRR$  for each algorithm for different choices of  $\ell$ ,  $m$  and  $n$ . Using the default parameters as specified in its manual (e.g.,  $k$ -mer length, number of iterations), EulerAlign crashed<sup>4</sup> when  $n$  is either 50,000 or 70,000. REPuter comes with two user options: (i) the exact length and number of substitutions allowed (i.e., errors), and (ii) the percentage error rate. REPuter achieved 100% sensitivity in  $CR$  and  $SRR$  when we chose option (i) but gave very poor results when we chose

---

<sup>3</sup>Pentium 4, 3-GHz, 1-GB RAM, WinXP. Experiments on long chromosomes have been run on Pentium 4, 2.8-GHz, 2.5-GB RAM, Linux.

<sup>4</sup>When we chose a small  $k$  in EulerAlign, the program ran out of memory and then crashed. When a larger  $k$  was chosen, the program ran for a very long time and was externally aborted.

TABLE 1. EXPERIMENTAL RESULTS FOR THE *CR* AND *SRR* MEASURES OF SENSITIVITY (SYNTHETIC SEQUENCES)

$n$	$\ell$	$m$	<i>EuAl</i>		<i>PALS</i>		<i>SAGRI</i>	
			<i>CR</i>	<i>SRR</i>	<i>CR</i>	<i>SRR</i>	<i>CR</i>	<i>SRR</i>
30,000	100	100	0	0.290	0.377	0.369	0.994	0.923
50,000	200	100	—	—	0.879	0.884	1	0.959
70,000	300	100	—	—	0.923	0.920	1	0.973

option (ii). Since option (i) does not reflect our chosen experimental setting, we do not include the results from REPuter in Table 1.

The conclusion derived from these simulations is that, based on the averages of *CR* and *SRR*, SAGRI appears to uniformly outperform both PALS and EulerAlign in terms of sensitivity.

## 5.2. Experiments with natural genomic data

We also compared the sensitivities of the four algorithms on the following genomic sequences:

- Mycoplasma (M.gen) of length 0.6 Mb and sequence id NC\_000908;
- Chlamydia trachomatis (C.tra) of length 1 Mb and sequence id NC\_000117;
- Archaeoglobus fulgidus (A.ful) of length 2.1 Mb and sequence id NC\_000917;
- Escherichia coli (E.coli) of length 4.6 Mb and sequence id NC\_000913;
- a 2 Mb segment of human chromosome 1 (Chr01-2M);
- a 2 Mb segment of human chromosome 2 (Chr02-2M);
- a 2 Mb segment of human chromosome 3 (Chr03-2M);
- a 4 Mb segment of human chromosome 3 (Chr03-4M);
- a 5 Mb segment of human chromosome 3 (Chr03-5M).

The first four organisms of the above list had been considered in the papers illustrating REPuter (Kurtz et al., 2000) and EulerAlign (Zhang and Waterman, 2005).

In order to focus on moderately long repeats, we set the parameters in each algorithm for repeats of length at least 50 with at most 10% error rate.

As mentioned earlier, since the set of reference repeats is unknown *a priori*, we take a cross-validation approach to compare algorithm  $\mathcal{G}$  relative to algorithm  $\mathcal{H}$ . Thus, it appears more reasonable to concentrate on the shared-region ratio (SRR), since the count-ratio (CR) involves the further problematic recognition of fragmented repeats. We introduce a relativized version of SRR of  $\mathcal{G}$  given  $\mathcal{H}$ , denoted by  $SRR(\mathcal{G}|\mathcal{H})$ , defined as the ratio of the shared repeat region found by  $\mathcal{G}$  and  $\mathcal{H}$  to the region found by  $\mathcal{H}$ .

We make some simple remarks below about  $SRR(\mathcal{G}|\mathcal{H})$  to facilitate an intuitive comparison of the algorithms.

1. If  $SRR(\mathcal{G}|\mathcal{H})$  is nearly 1 but  $SRR(\mathcal{H}|\mathcal{G})$  is not, then most of the repeats found by  $\mathcal{H}$  are also found by  $\mathcal{G}$ , but  $\mathcal{G}$  finds substantially more. Hence  $\mathcal{G}$  “outperforms”  $\mathcal{H}$ .
2. If both  $SRR(\mathcal{G}|\mathcal{H})$  and  $SRR(\mathcal{H}|\mathcal{G})$  are not nearly 1, then we can conclude that these algorithms are complementary, in the sense that each finds a substantial number of repeats not found by the other. It is then beneficial to run both algorithms for a more comprehensive mining of the repeats.
3. If both  $SRR(\mathcal{G}|\mathcal{H})$  and  $SRR(\mathcal{H}|\mathcal{G})$  are nearly 1, these algorithms perform similarly, and there is no relative advantage to run both algorithms.

Table 2 presents a comparative evaluation of the four algorithms in terms of the relativized SSR measure (no data is shown in those cases where an algorithm was found to crash). The table consists of three columns, one for each of the competing algorithms. For each of the chosen genomic sequences each column contains two entries:  $X/S$  and  $S/X$ , where  $S$  stands for SAGRI and  $X$  is  $E$  (for EuAl), or  $P$  (for PALS), or  $R$  (for REPuter). The  $X/S$  and  $S/X$  entries denote respectively  $SSR(X|S)$  and  $SSR(S|X)$ . The overriding conclusion is that with respect to sensitivity SAGRI strongly outperforms both PALS and REPuter (on the chosen variety of specimens, the  $S/X$  ratio never falls below 0.959). Although superior,

TABLE 2. COMPARISON OF SAGRI WITH THE THREE OTHER METHODS

	<i>EuAl</i>		<i>PALS</i>		<i>REPuter</i>	
	<i>E/S</i>	<i>S/E</i>	<i>P/S</i>	<i>S/P</i>	<i>R/S</i>	<i>S/R</i>
M.gen	0.764	0.962	0.766	0.998	0.755	1.000
C.tra	0.859	0.958	0.759	0.996	0.598	1.000
A.ful	0.307	0.806	0.518	0.995	0.413	0.999
E.coli	—	—	0.659	0.996	0.559	1.000
chr01-2M	0.693	0.914	0.520	0.974	0.230	0.974
chr02-2M	0.590	0.866	0.311	0.974	0.102	0.973
chr03-2M	0.750	0.884	0.484	0.968	0.143	0.974
chr03-4M	—	—	0.510	0.959	0.163	0.963
chr03-5M	—	—	0.513	0.964	0.175	0.970

SAGRI does not overwhelmingly outperform *EuAl*: indeed, for  $X = \text{EuAl}$  in a handful of instances the  $S/X$  ratio falls below 0.95, and there are cases for which the  $X/S$  and  $S/X$  ratio have comparable values, so that we conclude that SAGRI and *EuAl* may be considered complementary in terms of sensitivity, which is, after all, the most important performance measure in genomic repeat detection.

A more complete picture of the relative merits of the four algorithms is obtained when also the respective running times are taken into consideration. The relevant data is reported in Table 3. Here, for each of the nine genomic sequences and each of the four algorithms we report the total number of discovered repeat bases (rounded to Kbases) and the measured running times (in seconds) on our computing platform. An additional column reports the Hit/Validation split of SAGRI's running times.

The data in Table 3 is more effectively illustrated in Figure 4, which contrasts time and sensitivity. In a semilogarithmic plot (linear for sensitivity, logarithmic for time) the abscissas display the ratios of sensitivities, and the ordinates the ratios of running times, where in each case SAGRI's parameters are taken equal to 1. The diagram clearly shows that SAGRI dominates its competitors in absolute sensitivity. Although generally faster (and more sensitive) than *EuAl* and *PALS*, SAGRI is remarkably slower than *REPuter*, but significantly more sensitive.

Table 3 also reports data for two additional long-sequence experiments with SAGRI (Chr21-36M and Chr22-35M); these experiments cannot be completed by the other competing algorithms.

TABLE 3. TOTAL NUMBER OF THE REPEAT BASES (IN KBASES) DISCOVERED BY SAGRI AND OTHER METHODS AND THEIR RUNNING TIME ON EACH OF THE NINE GENOMIC SEQUENCES

	<i>EuAl</i>		<i>PALS</i>		<i>REPuter</i>		<i>SAGRI</i>		
	<i>Kb</i>	<i>Time</i>	<i>Kb</i>	<i>Time</i>	<i>Kb</i>	<i>Time</i>	<i>Kb</i>	<i>Time</i>	
M.gen	12	117	11	16	11	0.52	15	15.6	
C.tra	22	1326	18	19	14	1.13	24	4.3	
A.ful	22	199	30	86	24	2.56	59	11.5	
E.coli	—	—	129	338	109	5.9	196	32.7	
chr01-2M	229	565	161	116	71	2.25	303	15	9/6
chr02-2M	178	436	83	159	27	2.28	262	19	12/7
chr03-2M	484	609	285	157	83	2.19	571	29	11/18
chr03-4M	—	—	522	612	166	4.74	983	105	43/62
chr03-5M	—	—	609	1052	206	6.13	1144	106	38/68
chr21 (36M)							9209	6960	816/6144
chr22 (35M)							10,520	6840	213/6627

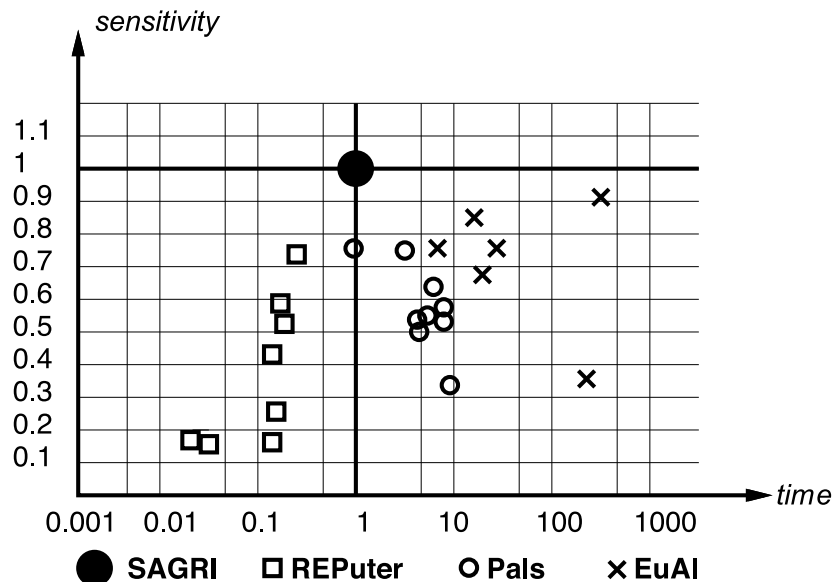


FIG. 4. Plot of sensitivity versus speed for the four analyzed algorithms.

## 6. CONCLUSION

In this paper, we have reported a novel approach to finding long approximate repeats in very long genomic sequences. This function is central to functional genomics, system biology, and evolutionary biology. The novelty of our approach is the recourse to the  $k$ -mer spectrum of the sequence, thus avoiding the burdensome tracking of locations. Indeed, for nontrivial repeats the spectrum appears to act as a powerful identifier of “hot spots,” i.e., genomic regions likely to contain repeats. The detected hot-spots identify sequence segments, whose spectrum in turn enables the placement of “companion segments.” Standard alignment completes the validation of repeat pairs.

We have developed a careful analysis of the algorithmic parameters to assure that the spectrum remain sufficiently sparse even for substantially long genomic sequences, and to control both false negatives and false positives. Careful algorithmic design has achieved acceptable running times.

We have also carried out an extensive experimentation, with both synthetic and natural sequences, designed to provide a comparative evaluation of SAGRI against the current state of the art. The results of this effort show that SAGRI is substantially more sensitive than its competitors (with almost complete coverage of their outputs in most cases). SAGRI speed is also quite acceptable; REPuter is generally much faster than SAGRI, but performs very poorly in terms of sensitivity (averaging less than 20% of SAGRI’s output on human DNA).

Although we feel that SAGRI is already a viable research tool in the genomic arsenal, and we offer it as an open source facility, we plan to further refine it, so that it can be applied to much longer sequences—such as complete human chromosomes and possibly the entire genome—with no sacrifice of sensitivity.

## APPENDICES

### A. Analysis of algorithm parameters

Each instance of the problem is characterized by a collection of parameters, which will in turn determine the selection of the algorithm parameters.

Specifically, for the purpose of analysis, we assume to deal with Bernoulli genomic sequence with i.i.d. symbols. Problem instance parameters are sequence length  $N$  and characteristics of repeat noise. The latter is modeled as a Bernoulli process of “noise events” with a probability  $q$ . In the general case, to be discussed later, a noise event can be either a substitution or an indel (in fixed random proportion). Notice

that the noise is meant to model the editing distance between the two repeat segments ( $L_1, L_2$ ) and not the mutation events along the genomic sequence. Moreover, without loss of generality, all editing noise events are assumed to occur in segment  $L_1$  (obviously, a substitution is indifferently attributed to  $L_1$  or  $L_2$ , and an insertion/deletion in  $L_2$  is equivalent to a deletion/insertion in  $L_1$ ).

The algorithm parameters are the length  $k$  of the spectrum terms and the function  $\tau(r)$  that defines the criterion for path termination.

We shall analyze the various features in separate subsections.

*A.1. Selection of probe length  $k$*

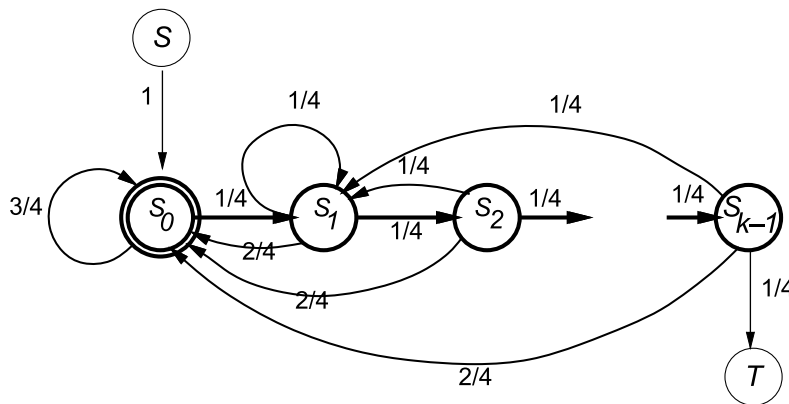
The parameter  $k$  is chosen as a compromise between effectiveness and efficiency. If we choose  $k$  “too small,” then as the length  $m$  of the scanned sequence grows the spectrum  $S_m$  may become so crowded (i.e., a large fraction of  $k$ -mers are represented) that any discrimination between repeat  $k$ -mers and random  $k$ -mers may be lost (this happens when  $k < \log_4 m + 1$ ). On the other, if we choose  $k$  “too large,” then the algorithm will miss identifying a repeat whose two segments do not contain a perfectly preserved  $k$ -mer.

We begin by considering the crowding of the spectrum. This analysis requires the evaluation of the probability that our random sequence *does not contain* a specific  $k$ -mer as a substring (and, by complement to 1, that it contains it). To this end, for a specific  $k$ -mer we can construct its recognition automaton  $\mathcal{A}$ , whose states correspond to the recognition of the distinct  $k$  proper prefixes of the  $k$ -mer (see Fig. 5 for a typical illustration of the state diagram of  $\mathcal{A}$ , where state  $s_0$  corresponds to the empty prefix). This automaton can be interpreted as follows: If we inject a “commodity” (probability) of value 1 from an auxiliary state  $S$  at time 0, the amount of commodity exiting to state  $T$  after exactly  $j$  steps is the total probability  $c_j$  of the sequences of length  $j$  ending with the selected  $k$ -mer. Thus, if we multiply each arc weight by the formal delay variable  $z$ , the transmission function from  $S$  to  $T$  is the generating function  $c(z)$  of the sequence  $\{c_j\}$ .

It is well known that the state diagram of  $\mathcal{A}$  depends upon the specific  $k$ -mer. In fact, state  $s_j$  ( $j = 1, \dots, k - 1$ ) has four outgoing arcs: a “recognition” arc directed from  $s_j$  to  $s_{j+1}$ , and three “return” arcs. A return arc from  $s_j$  is directed to  $s_i$  ( $i \leq j$ ) if and only if the length- $(i - 1)$  suffix of  $s_j$  coincides with the length- $(i - 1)$  prefix of the  $k$ -mer. It follows that each  $s_j$  ( $j \geq 1$ ) has a return to  $s_1$  (in correspondence with the symbol representing the length-1 prefix) and, possibly, one return to some  $s_i$  ( $2 \leq i$ ) depending on the stated prefix-suffix condition; in all cases, however, each  $s_i$  has at least one return to  $s_0$ . It is clear that, for values of  $k > 10$ , a very small fraction of states will have returns to downstream states  $s_i$  ( $i \geq 2$ ). Therefore, here we make the simplifying assumption that the third return arc is always directed to  $s_0$ , thereby mildly underestimating the probability of occurrence of a  $k$ -mer.

The transmission function from  $S$  to  $T$  of automaton  $\mathcal{A}$  of Figure 5 can be calculated using standard flow-graph reduction techniques (eliminating one node at a time), and we obtain:

$$c(z) = \frac{(z/4)^k (1 - z/4)}{1 - z + (z/4)^k (3z/4) - (z/4)(1 - z)(1 - (z/4)^{k-1})} = \frac{f(z)}{g(z)}$$



**FIG. 5.** State diagram of automaton  $\mathcal{A}$ . The path recognizing the selected  $k$ -mer is shown in bold face.

The asymptotic behavior of the sequence  $\{c_j\}$  is determined by the largest root of the polynomial  $\phi(x) = x^{k+1}g(1/x)$ , which is easily and conveniently approximated<sup>5</sup> as  $1 - 4^{-k}$ .

The conclusion of this analysis is that we may comfortably skirt the difficulty of distinguishing among distinct generic  $k$ -mers. Within this simplification, we wish to evaluate the probability that a sequence of length  $m$  **contains** a given  $k$ -mer, i.e.,  $1 - \sum_{i>m} c_i$ . First we claim that, denoting  $\alpha = 1 - 4^{-k}$ ,  $c_k \alpha^{j-k}$  is a suitable estimate of  $c_j$ ; indeed,  $c_1 = \dots = c_{k-1} = 0, c_k = 4^{-k}$  and, for  $j > k$ , we may use  $c_j = c_k \alpha^{j-k}$ . It follows that for large  $m$

$$1 - \sum_{i>m} c_i \approx 1 - \frac{4^{-k}}{\alpha^{k-1}(1-\alpha)} \alpha^m$$

Since  $4^{-k}/(\alpha^{k-1}(1-\alpha)) \approx 1/(1-(k-1)4^{-k}) \approx 1$ , we conclude that

$$1 - \left(1 - \frac{1}{4^k}\right)^m \tag{1}$$

is an acceptable estimate of the sought probability, indicating that the conditioning due to string overlap is indeed very mild, and that we may consider valid the analogy of throwing  $m$  balls into a collection of  $4^k$  equally likely bins (a bin that remains empty corresponds to a missing  $k$ -mer).

Since  $(1 - 4^{-k})^m \approx e^{-m/4^k}$ , we can say that

$$1 - e^{-m/4^k}$$

approximates the probability that the sequence *contains* a specific  $k$ -mer.

A  $k$ -mer arising from an arbitrary position in the sequence is referred to as a *fooling probe*. We wish to ensure that, as  $m$  approaches  $N$ , the probability of a fooling probe remains sufficiently small, so that we may confidently discriminate an actual repeat from an assembly of fooling probes. For a small value  $\epsilon_1$ , we may require  $1 - e^{-N/4^k} \leq \epsilon_1$ , which yields

$$k \geq \log_4 N + \log_4 \epsilon_1$$

so that  $k = \log_4 N + 2$  corresponds to  $\epsilon_1 \leq 1/16$ .

On the other hand, for  $m = aN$  and  $k = \log_4 N + \delta$ , the probability of a fooling probe decreases to  $e^{-a/4^\delta}$ .

### A.2. The occurrence of false negatives (missed repeats)

Once  $k$  has been selected, we wish to estimate the probability of a false negative (i.e., the miss of an actual repeat). Due to the design of our algorithm, such event occurs when the two segments of the repeat contain no perfectly preserved  $k$ -mer, i.e., when a binary Bernoulli sequence of length  $\ell$  (the length of the repeat segment) with parameter  $\text{Prob}(0) = p = 1 - q$  *does not contain* even one string of 0s and length  $k$ .

Let  $S$  be a sequence of independent Bernoulli random variables in which, for  $1 \leq i \leq n$ ,  $S[i]$  takes two values 0 and 1 with probabilities  $p$  and  $q = 1 - p$  respectively. Here, we shall confine ourselves to  $p > 1/2$ . For  $1 \leq j \leq n$ , we let  $b_j$  be the probability that  $0^k$  is first found in  $S[1, j]$ , i.e.,  $S[j - k + 1, j] = 0^k$  and there does not exist any  $1 \leq i < j$  such that  $S[i - k + 1, i] = 0^k$ . Let  $B_\ell = \sum_{1 \leq j \leq \ell} b_j$ , which denotes

<sup>5</sup>Indeed  $\phi(1) = 3/4^{k+1}$  (which is very close to 0); moreover,

$$\phi'(x) = (k+1)x^k - kx^{k-1} - \frac{1}{4}(x^{k-1} - 4^{-k+1} + (x-1)(k-1)x^{k-2})$$

so that  $\phi'(1) = 3/4 + 4^{-k}$ . It follows that  $\phi(1 - 1/(4^k + 4/3)) \approx 0$ . If we approximate  $1 - 1/(4^k + 4/3)$  with  $1 - 1/4^k$ , we make an error of about  $4^{-2k}$ .

Notice also that the same analysis gives  $1 - 4^{-k}$  as the approximation of the root of  $1 - z + (z/4)^k(3z/4)$ , the polynomial one obtains when all return arcs are directed to  $s_0$ . This remark substantiates the adequacy of our approximation.

the probability that  $0^k$  can be found somewhere in  $S[1, \ell]$ . Thus, for  $\ell \geq k$ ,  $\bar{B}_\ell = 1 - B_\ell$  represents, in our context, the probability of a false negative for a repeat of length  $\ell$ .

For  $\ell \geq k + 1$ , the event that  $0^k$  occurs for the first time in  $S[1, \ell]$  is the same event as:  $S[\ell - k + 1, \ell] = 0^k$ ,  $S[\ell - k] = 1$  and  $0^k$  does not occur in  $S[1, \ell - k - 1]$ . That is,

$$b_\ell = p^k q \bar{B}_{\ell-k-1}, \quad \ell > k.$$

Since  $b_\ell = \bar{B}_{\ell-1} - \bar{B}_\ell$ , we have

$$\bar{B}_\ell - \bar{B}_{\ell-1} + p^k q \bar{B}_{\ell-k-1} = 0, \quad \ell > k.$$

This recurrence relation, with initial values  $\bar{B}_0 = \dots = \bar{B}_{k-1} = 1$ ,  $\bar{B}_k = 1 - p^k$ , enables us to compute the analytical value of  $\bar{B}_\ell$ . Moreover this recurrence relation implies that

$$\bar{B}_\ell = \beta \alpha^\ell [1 + o(1)]$$

as  $\ell \rightarrow \infty$  for some positive constants  $\alpha$  and  $\beta$ . Furthermore,  $\alpha$  is the largest positive root of the polynomial  $\phi(x) := x^{k+1} - x^k + p^k q$ .<sup>6</sup>

### A.3. Criterion for path termination

The criterion for path termination during the execution of the scan algorithm is embodied by the function  $\tau(r)$ . The specification of  $\tau(r)$  is based on the need to discriminate between likely repeats and artifacts of the “random” background of the sequence.

Here we take as null hypothesis  $H_0$  the condition that “we are dealing with an actual repeat,” i.e., that the discrepancies with the reference path are due to editing noise. Correspondingly, for a given integer  $s$ , we evaluate the probability that a binary Bernoulli string of length  $r$  and parameter  $q$ , beginning and terminating with a 1, contains at least  $s$  1’s, that is:

$$P_q(s) = q^2 \sum_{j=s-2}^{r-2} \binom{r-2}{j} q^j (1-q)^{r-2-j}$$

and, for a chosen confidence level  $\epsilon$ , define the threshold

$$\tau(r) = \max\{2 \leq s \leq r - 2 \text{ s.t. } P_q(s) > \epsilon\}$$

$\epsilon$  is therefore the  $p$ -value of the test (the probability of making an error if  $H_0$  is rejected, i.e., the path is terminated). Function  $\tau(r)$  is characterized by the integers  $r_1, r_2, r_3, \dots$ , where  $r_j \geq 2$  is such that for  $r_j \leq r < r_{j+1}$ ,  $\tau(r) = j$  (here, conventionally,  $\tau(1) = 1$ ). As an example, below we report the table of values of  $r_j$  for  $j = 1, \dots, 7$ ,  $\epsilon = 0.005$  and  $q = 0.1$ :

$j$	1	2	3	4	5	6	7
$r_j$	1	2	9	19	29	39	49

---

<sup>6</sup>Root  $\alpha$  is easily estimated observing that  $b_1 = \dots = b_{k-1} = 0, b_k = p^k, b_{k+1} = \dots = b_{2k} = qp^k, b_j \approx qp^k \alpha^{j-2k}$  for  $j > 2k$ . This yields the equation  $p^k + kqp^k + qp^k \alpha / (1 - \alpha) \approx 1$ , and hence

$$\alpha \approx \frac{1 - p^k(1 + kq)}{1 - p^k(1 + (k - 1)q)}.$$

For example, when  $k = 12$  and  $p = 0.9$ , we obtain  $\alpha = 0.931\dots$ , the correct solution being  $\alpha = 0.943\dots$ .

#### A.4. Control of false positives

Finally, we wish to investigate *false positives*, i.e., events that may appear as repeats due to the interaction of our algorithm with the randomness of the sequence background, but should be discriminated against *bona fide* repeats (for synthetic sequences, deliberately inserted repeats). The distinguishing feature will be that the random artifacts are statistically of small length, so that a threshold on the length will be used to discriminate them.

For the analysis we refer to formula (1), which approximates the probability of a specific  $k$ -mer occurring in a sequence of length  $m$ . Although, in principle, a false positive may result from the contribution of several disjoint sites in the genomic sequence, we argue here that we may restrict ourselves to considering a single site. Our coarse argument goes as follows. By (1) the probability of a specific  $\ell$ -mer is  $1 - e^{-m/4^\ell} \approx m/4^\ell$ , where  $\ell = k + s$ . If the observed  $\ell$ -mer is the result of the contribution of *two* disjoint sites, one contributes an  $\ell_1$ -mer, the other an  $\ell_2$ -mer, with  $\ell_1 = k + s_1$ ,  $\ell_2 = k - 1 + s_2$ , and  $s_1 + s_2 = s$ . The probability of such event is approximately  $\binom{m}{2}(1/4^{\ell_1})(1/4^{\ell_2}) \approx (m^2/2)(1/4^{\ell+k-1})$ . Since  $k = \lceil \log_4 N \rceil + 2$  the ratio of the two probabilities (double site/single site) is thus  $2m/4^k \leq (2m/N)4^{-2} \leq 2 \cdot 4^{-2} = 1/8$ , justifying our restriction to false positives arising from a single site.

For simplicity, we shall consider Hamming noise and represent a false-positive event as a binary strings  $z$  of agreements/disagreements with respect to the reference path (a disagreement being represented as a 1). We wish to obtain a coarse estimate of the probability that a random genomic sequence of length  $m$  contains a length- $\ell$  false positive.

Report of a false positive implies path termination. In turn, path termination implies that the length- $\ell$  binary string  $z$  ends with a suffix  $u$  of the form  $1(0 \vee 1)^{|u|-2}1$  of length  $|u|$ , containing  $\tau(|u|)$  1's ( $|u| \geq 3$ ) (our criterion for path termination) and begins with a prefix of the form  $0^k$ , determining path initiation (i.e., a match with the reference path). Since each 0 has probability  $1/4$  (agreement), and each 1 has probability  $3/4$  (disagreement), the probability of  $z$  is coarsely estimated as

$$\frac{m}{4^k} \frac{3^{\text{wt}(z)}}{4^{\ell-k}} N(\ell - k)$$

where  $\text{wt}(z) = \tau(|u|)$  is the number of 1s of  $z$  and  $N(\ell - k)$  is the number of ways in which the  $(\ell - k)$ -suffix  $u$  of  $z$  can be selected.

The combinatorial evaluation of  $N(\ell - k)$  is very involved, and its difficulty probably exceeds its usefulness. In fact, the distribution of the 1s occurring in  $1(0 \vee 1)^{|u|-2}1$  is such that in each proper prefix  $v$  of it the number of 1s must be smaller than the termination bound  $\tau(|v|)$ . We avoid this tedious analysis, and consider the upper bound  $\binom{|u|-2}{\tau(|u|)-2}$ , corresponding to an unrestricted selection of the 1s. Letting  $r = \ell - k$ , the sought probability is very conservatively bounded above by

$$\frac{3m}{4^k} \cdot \frac{3^{\tau(r)}}{4^r} \cdot \binom{r-2}{\tau(r)-2}.$$

We have already seen that for  $k \geq \log_4 N + 2$  the first term of the above product is bounded above by  $1/16$ . Choosing, for example,  $r = 15$ , then  $\tau(r) = 3$ , the above product has value  $\approx 2 \times 10^{-8}$ .

These observations provides guidelines for the acceptance of candidate repeats on the basis of their length.

#### A.5. Backtrack depth of modified DFS

The depth-first-search advancing mechanism of the FindHit phase is modified by restricting the backtrack depth to a suitable value  $h$  (so that all buds at higher depth are automatically eliminated). A detailed analysis of the most general case (Levenshtein distance) is exceedingly complex and presumably not worth the effort. Therefore, we restrict ourselves to substitution errors and adopt a simplified model for analysis.

We ask the following question:

- What is the probability that the correct path originates  $h$  positions upstreams of the (current) position where the current (spurious) putative path is not yet terminated?

We wish to obtain a conservative (pessimistic) upper bound to the sought probability. Since the current path is spurious, it is supported position-by-position by fooling probes. Since a mismatch fooling probe is three times as probable as a match fooling probe (it can be chosen in three ways rather than one) and the spurious path is not being terminated, the most conservative case occurs when  $\tau(h) - 1$  mismatches have occurred in the last  $r$  positions. Assume that  $j \leq \tau(h) - 1$  is the actual number of mismatches. Since the current path is spurious, both matches and mismatches are supported by fooling probes: in the most conservative hypothesis, the support for the spurious path arises from a single site at some position to the left of the current one ( $m$ ). Thus, the probability of this event is approximately

$$P(h, j) \approx 3^j \frac{m}{4^{k+h}}$$

and

$$P(h) = \sum_{j=0}^{\tau(h)-1} P(h, j) = \sum_{j=0}^{\tau(h)-1} < \frac{m}{4^{k+h}} \frac{3^{\tau(h)}}{2}.$$

Recalling that, by design,  $m/4^k \leq 1/16$  (Subsection A.1), for  $h = 9$  we have  $\tau(h) = 3$  (Subsection A.3), and we obtain

$$P(9) < 3.3 \cdot 10^{-6}.$$

### ACKNOWLEDGMENT

We thank Y. Zhang for providing the EulerAlign program. K.P.C. was supported by the National University of Singapore (NUS) ARF (research grant R-155-000-051-112). F.P.P. was supported in part by the Kwan Im Thong Visiting professorship at NUS. L.X.Z. was supported by NUS ARF (grant 146-000-068-112).

### DISCLOSURE STATEMENT

No competing financial interests exist.

### REFERENCES

- Adebiyi, E.F., Jiang, T., and Kaufmann, M. 2001. An efficient algorithm for finding short approximate non-tandem repeats. *Bioinformatics* 17, S5–S12.
- Arabidopsis Genome Initiative. 2000. Analysis of the genome sequence of the flowering plant *Arabidopsis thaliana*. *Nature* 408, 796–815.
- Bao, Z. and Eddy, S. R. 2002. Automated *de novo* identification of repeat sequence families in sequenced genomes. *Genome Res.* 12, 1269–1276.
- Bedell, J., Korf, I., and Gish, W. 2000. Maskeraid: a performance enhancement to repeatmasker. *Bioinformatics* 16, 1040–1041.
- Bowen, N.J., and Jordan, I.K. 2002. Transposable elements and the evolution of eukaryotic complexity. *Curr. Issues Mol. Biol.* 4, 65–76.
- Charlesworth, B., Sniegowski, P., and Stephan, W. 1994. The evolutionary dynamics of repetitive DNA in eukaryotes. *Nature* 371, 215–220.
- Edgar, R.C., and Myers, E.W. 2005. PILER: identification and classification of genomic repeats. *Bioinformatics* 21, i152–i158.
- Human Genome Sequencing Consortium. 2001. Initial sequencing and analysis of the human genome. *Nature* 409, 860–921.
- Jurka, J. 2000. Repbase update: a database and an electronic journal of repetitive elements. *Trends Genet.* 16, 418–420.
- Kurtz, S., Ohlebusch, E., Schleiermacher, C., et al. 2000. Computation and visualization of degenerate repeats in complete genomes. *Proc. Int. Conf. ISMB*, 228–238.

- Kurtz, S., and Schleiermacher, C. 1999. Reputer: fast computation of maximal repeats in complete genomes. *Bioinformatics* 15, 426–427.
- Medstrand, P., van de Lagemaat, L., Dunn, C., et al. 2005. Impact of transposable elements on the evolution of mammalian gene regulation. *Cytogenet. Genome Res.* 110, 342–352.
- Pevzner, P.A., Tang, H., and Tesler, G. 2004. *De novo* repeat classification and fragment assembly. *Genome Res.* 14, 1786–1796.
- Preparata, F.P., Upfal, E., and Heath, S. 2005. Sequence reconstruction from nucleic acid micro-array data, 177–193. In: Nunnally, B., ed., *Analytic Techniques from DNA Sequencing*. M. Dekker, New York.
- Sagot, M.-F. 1998. Computation and visualization of degenerate repeats in complete genomes. *Proc. 3rd LATIN* 111–127.
- Schmidt, J.P. 1998. All highest scoring paths in weighted grid graphs and their application to finding all approximate repeats in strings. *SIAM J. Comput.* 27, 972–992.
- Volfovsky, N., Haas, B.J., and Salzberg, S.L. 2001. A clustering method for repeat analysis in DNA sequences. *Genome Biol.* 2, RESEARCH0027.
- Zhang, Y., and Waterman, M. 2005. An Eulerian path approach to local multiple alignment for DNA sequences. *Proc. Natl. Acad. Sci. USA* 102, 1285–1290.

Address reprint requests to:  
Dr. Franco P. Preparata  
Department of Computer Science  
Brown University  
Providence, RI 02912

E-mail: franco@cs.brown.edu

