BIOINFORMATICS

Vol. 17 Suppl. 1 2001
Pages S5–S12

An efficient algorithm for finding short approximate non-tandem repeats

Ezekiel F. Adebiyi[1], Tao Jiang[2] and Michael Kaufmann[1]

[1]Wilhelm-Schickard-Institut für Informatik, Universität Tübingen, Sand 13, Tübingen, 72076, Germany and [2]Department of Computer Science and Engineering, College of Engineering, University of California, Riverside, USA

Received on February 6, 2001; revised and accepted on April 2, 2001

ABSTRACT

We study the problem of approximate non-tandem repeat[†] extraction. Given a long subject string S of length N over a finite alphabet Σ and a threshold D, we would like to find all short substrings of S of length P that repeat with at most D differences, *i.e.,* insertions, deletions, and mismatches. We give a careful theoretical characterization of the set of *seeds* (*i.e.,* some maximal exact repeats) required by the algorithm, and prove a sublinear bound on their expected numbers. Using this result, we present a sub-quadratic algorithm for finding all short (*i.e.,* of length $O(\log N)$) approximate repeats. The running time of our algorithm is $O(DN^{3pow(\epsilon)-1}\log N)$, where $\epsilon = D/P$ and $pow(\epsilon)$ is an increasing, concave function that is 0 when $\epsilon = 0$ and about 0.9 for DNA and protein sequences.

Contact: adebiyi@informatik.uni-tuebingen.de

INTRODUCTION

Sequence *motifs* (*i.e.,* special patterns in bio-molecular sequences that are conserved over evolution) play an important role in the identification of novel functional units (Hodgman, 1989). For example, the gene sequences of a gene family may share short, conserved motifs that correspond to meaningful *domains* in their protein structures and may have regulatory elements containing the same motif. A collection of such motifs as regular expressions is given in PROSITE database (Bairoch, 1992). A useful approach for identifying meaningful sequence motifs is to find non-tandem approximately repeating patterns that occur more frequently than expected by chance. Thus, finding unknown exact and approximate repeat is an important problem in computational biology.

The problem of finding exact and approximate repeats has been studied extensively in the mathematical and algorithmic literature (see, *e.g.,* (Gusfield, 1997; Kurtz et al., 2000)). The approaches considered include combinatorial methods, statistical modeling, and the use of suffix trees. The problem of exact repeat extraction seems well settled, since it has an optimal $O(N)$ time algorithm (Gusfield, 1997). However, the problem of approximate repeat extraction still remains a big challenge, since the nature of the problem implies an exponential combinatorial complexity in terms of the target length of approximate repeats considered.

A popular strategy for finding approximate repeats is to first search for exact repeats of small but appropriate lengths and then use (some of) the exact repeats as *seed strings* (or simply *seeds*) to form maximal approximate repeats by expansion (Kurtz et al., 2000). The latter step may take into account the target approximate repeat length. Two types of exact repeats can be used as seeds: maximal repeats and the supermaximal repeats. A *maximal repeat* β in a string S is a substring of S such that substrings $a\beta c$ and $b\beta d$ occur in S for some $a \neq b$ and $c \neq d$, $a, b, c, d \in \Sigma$. Note that a maximal repeat might be a proper substring of another maximal repeat. Therefore, a *supermaximal repeat* is a maximal repeat that does not occur as a substring of any other maximal repeat.

In this paper, we

1. prove a sublinear upper bound of $O(N^{2pow(\epsilon)-1-d})$ for some $d < 1$ on the expected number of supermaximal repeats in a string of length N, where $\epsilon = D/P$, D is the maximum number of differences allowed in the repeats, P is the length of the repeats, and $pow(\epsilon)$ is an increasing, concave function that is 0 when $\epsilon = 0$ and less than 1 when ϵ is small enough.
2. use the first result to design an $O(N + DN^{3pow(\epsilon)-1}\log N)$ expected-time algorithm for the finding of all short (*i.e.,* of length O($\log N$)) approximate repeats. The algorithm is based on a delicate construction of seeds from supermaximal and maximal exact repeats. In practice, the running time of the algorithm is about $O(N^{1.7}\log N)$ for DNA and protein sequences.

[†] Throughout the paper we only consider non-tandem repeats.

© Oxford University Press 2001

Our results are inspired by Myers' work on approximate keyword search. In (Myers, 1994), Myers presented a sub-linear time algorithm for the problem of searching for approximate keywords: Given (relatively) short query string W of length P, a long subject string S of length N, and a threshold D, find all substrings of S that align with W with at most D differences. Myers' algorithm is based on the notion of the *condensed D-neighborhood* of a string and uses an index structure containing information regarding where approximate matches of keywords of sizes up to $\log_{|\Sigma|} N$ are to be found in string S. These are among the key ingredients in our construction.

Note that the above results are based on the assumption that the strings are randomly generated by Bernoulli trials. It is known that many DNA sequences, are in fact, close to being random according to statistical tests(Blumer et al., 1989). We have also implemented our approach and tested it on real DNA genomic sequences to justify its practical efficiency.

Previous Work

A detailed, up-to-date survey of previous work on finding approximate repeats can be found in (Kurtz et al., 2000). Here, we only mention results that are directly related to our results. Kurtz *et al.* (Kurtz et al., 2000) have recently presented an algorithm for finding approximate repeats of length P which runs in $O(N + Dz)$ time for hamming distance and in $O(N + D^3 z)$ time for edit distance, where z is the number of seeds and D is the maximum number of mismatches or differences expected in the resulting repeats. Their construction works by first computing all seeds of lengths above a specific threshold and testing whether each seed can be left or right maximally extended to a D-(mismatch or differences) repeat. For the hamming distance, they extend each seed by computing the length of the longest common prefix of two substrings, while for the edit distance, each seed is extended using a combination of dynamic programming algorithm due to Ukkonen (Ukkonen, 1985) and the longest common prefix technique. They estimate that the expected number of seeds required is $O(N^2(1/|\Sigma|^s))$, where s is the length of the seed used. This is suitable for long approximate repeats (that is, P is large), since the size of the pool of seeds decreases as P increases. For short repeats, however, the situation is different. If $P = c \cdot \log N$, where $c > 1$, the running time would be $O(N + D^3 \cdot N^{2-\frac{c}{D+1}})$ for edit distance, which is worse than the running time of our algorithm when D is modestly large.

Sagot (Sagot, 1998) designed an algorithm for finding approximate repeats of length P using suffix trees and the notion of D-neighborhood. Her algorithm runs in time $O(NP^D|\Sigma|^D)$, which is clearly inefficient for moderately large values of P and D. Moreover, her algorithm generates one D-neighborhood for each maximal repeat. By the concept of condensed D-neighborhood and the use of supermaximal repeats as seeds, both drawbacks can be avoided. This will be discussed later more formally. Furthermore, the concept of condensed D-neighborhood even allows us to find approximate repeats using an approximate common seed instead of an exact seed.

We note that maximal exact repeats have been used as seeds in several previous constructions. For example, the work of Kurtz *et al.* (Kurtz et al., 2000), Repeat-finder (TIGR, 1999), MuMmer (Delcher et al., 1999), and RE-Puter (Kurtz and Schleiermacher, 1999) all use maximal repeats. However, in all these constructions, general maximal exact repeats are used as seeds without any distinction from supermaximal repeats. Our construction will focus on supermaximal repeats instead, and use additional (non-super)maximal repeats only when it is necessary. This distinction between supermaximal seeds and general maximal repeats is a key to our design of the sub-quadratic algorithm.

The rest of the paper is organized similar to the structure of the algorithm. In the next section, we give a characterization of the set of seeds from the set of maximal exact repeats based on some properties. The maximal exact repeats can be computed using the suffix tree of the target string S. In section 3, we prove that the expected number of seeds of length $2\log_{|\Sigma|} N$ is $O(N^{2pow(\epsilon)-1-d})$ for some $d < 1$, and discuss how to find all short approximate repeats from these seeds using the sublinear-time algorithm of Myers for approximate keyword search, which is simply called *SAM* in this paper. In section 4, we report some experimental results and discuss the practical efficiency of our approach.

CHARACTERIZATION OF SEEDS

In this section, we present some properties of maximal exact repeats which are useful in the construction of our seeds. It is well-known (Gusfield, 1997) that there exists a linear time algorithm to find maximal and supermaximal exact repeats, since each of them can be represented by an appropriate subtree of the suffix tree for string S.

As a first derivation, we give an estimate of the expected length of a longest exact repeat in a string of length N. Let L denote the expected length of a longest exact repeat.

LEMMA 1. Given a string of length N, the expected length L of the longest repeat is $2\log N/\log|\Sigma| + o(log N)$.

Proof. By standard probabilistic analysis. □

Note that the above expected length bound also holds for maximal and supermaximal repeats. It is clear (Kurtz and Schleiermacher, 1999) that from the maximal repeats, all the non-maximal repeats can be derived. It is also true that from the supermaximal repeats, all the maximal

repeats that are not supermaximal repeats and non-maximal repeats can be derived. This is stated in the following lemma. Note that from now on, when we refer to maximal repeats, we mean those maximal repeats that are not supermaximal repeats.

LEMMA 2. Each maximal or non-maximal repeat is contained in some supermaximal repeat.

The following definitions taken from (Myers, 1994) will lead us to another property concerning the relationship between supermaximal repeats and maximal repeats. Let $\delta(V, W)$ denote the edit distance between two strings V and W.

DEFINITION 1. *(Myers, 1994)* The D-neighborhood of a string W is the set of all strings with edit distance at most D from W, *i.e.,* $N_D(W) = \{V : \delta(V, W) \leq D\}$.

DEFINITION 2. *(Myers, 1994)* The condensed D-neighborhood of W is the set of all strings in the D-neighborhood of W that do not have a prefix in the neighborhood, *i.e.,* $\overline{N_D}(W) = \{V : V \in N_D(W)$ and no prefix of V is in $N_D(W)\}$.

A *repeat family* is defined to be the approximate repeats in a D-neighborhood of some supermaximal (exact) repeat. The only approximate repeats that may not be in any repeat family are those that are extendable from maximal repeats of lengths less than $L - D$. They may be significant, since not all positions where there exist supermaximal repeats represent maximal repeat substrings of a supermaximal repeat. A simple way to detect them is to sequentially fish out maximal repeats whose length are less than $L - D$, but a more rigorous approach is the following further refinement, based on Lemma 2, of a 'supermaximal' repeat family via a covering by the corresponding 'maximal' repeats with a smaller neighborhood of radius $\hat{D}$. A formal definition is given below.

DEFINITION 3. For a supermaximal repeat W, we call the set of approximate repeats in the condensed D-neighborhood $\overline{N_D}(W)$ the extended repeat family $E(D, W)$ for W.

DEFINITION 4. Given a maximal repeat w which is a *substring of a supermaximal repeat* W and distances D and $\hat{D}$ with $\hat{D} \leq D$, we define the set of approximate repeats in $\overline{N_{\hat{D}}}(w)$ to be the condensed repeat family $C(\hat{D}, w)$.

Intuitively, the relationship between $E(D, W)$ and $C(\hat{D}, w)$ can be observed as the covering of the elements of $E(D, W)$ by some smaller sets around some central elements w. In essence, what we need to break $E(D, W)$ into $C(\hat{D}, w)$ is to find these central elements and let $\hat{D}$ be the edit distance between w and the elements of $E(D, W)$ which are the closest to w. We identify these central elements w as the set of maximal repeats in $E(D, W)$. Therefore, W and w, respectively, are called the central elements or representatives of the corresponding repeat families $E(D, W)$ and $C(\hat{D}, w)$. By definition, $C(\hat{D}, w) \subseteq E(D, W)$. The subset property of $C(\hat{D}, w)$'s in $E(D, W)$ implies a coverage property, so that for each supermaximal repeat, we can compute $\hat{D}$ by a binary search on $\hat{D}$, $1 \leq \hat{D} \leq D$, until $\hat{D}$ is minimized but $\cup C(\hat{D}, w) = E(D, W)$. The relationship between D and $\hat{D}$ indicates this new relation between the sets of maximal repeats and supermaximal repeats.

From the above results that show the ability to break $E(D, W)$ into several $C(\hat{D}, w)$'s, we give again another important property that enables our use of all supermaximal repeats as seeds to be robust enough to capture all other significant repeated patterns. Therefore, our set of seeds is a combination of the set of all supermaximal repeats (denoted as U) and the set of some *significant* maximal repeats (denoted as M').

We will show how to obtain these maximal repeats in the next section. Note that the locations of those maximal repeats, whose positions are not covered by supermaximal repeats but have lengths greater than $L - D$, will be located as approximate re-occurrences of patterns extendable from some supermaximal exact repeats. In the following, $\xi(\hat{D}, D)$ denotes the average number of condensed repeat families $C(\hat{D}, w)$ in $E(D, W)$.

FINDING SHORT APPROXIMATE REPEATS

Using the result of Lemma 1 of section 2 and the sublinear time algorithm of Myers for finding approximate keywords, SAM (Myers, 1994), we prove in this section a significantly sub-quadratic time algorithm for finding all short approximate repeats, that is, approximate repeats of length $O(\log N)$. This is derived by executing SAM for each of the seeds with a given edit distance D. To prove this, we begin with an overview of some relevant results in (Myers, 1994). Next, we prove a sublinear bound on the number of supermaximal repeats and, finally, show how to extract from the set of maximal repeats, all significant maximal repeats . The construction of the sub-quadratic algorithm then follows.

An Upper Bound on the Expected Number of Supermaximal Repeats

Here, we prove an upper bound on the expected number of supermaximal repeats using techniques from (Myers, 1994). We state first an upper bound by Myers for

generating the strings in $\overline{N_D}(W)$. Let $Pr(T, D)$ be the maximum probability of matching a string in a condensed D-neighborhood of a string W of length T at a given position of the subject string.

LEMMA 3. *(Myers, 1994)* For $T = \log_{|\Sigma|} N$ and $D \leq T$ and given that $|\overline{N_D}(W)| \leq \frac{c}{(c-1)} Bnd(T, D, c)$ and $Pr(T, D) \leq \frac{c}{(c-1)} \frac{Bnd(T,D,c)}{|\Sigma|^T}$ for $c > 1$, where $Bnd(T, D, c) = (\frac{c+1}{c-1})^T c^D |\Sigma|^D$, $|\overline{N_D}(W)| < 2N^{pow(\epsilon)}$ and $Pr(T, D) < 2N^{pow(\epsilon)-1}$, where $\epsilon = D/T$, $pow(\epsilon) = \log_{|\Sigma|}(c + 1)/(c - 1) + \epsilon \log_{|\Sigma|} c + \epsilon$ and $c = \epsilon^{-1} + \sqrt{1 + \epsilon^{-2}}$.

LEMMA 4. *(Myers, 1994)* For any string $s \in S$, the strings in $\overline{N_D}(s)$ can be generated in $O(DN^{pow(\epsilon)})$ worst-case time, where $\epsilon = D/T$, $pow(\epsilon) = \log_{|\Sigma|}(c + 1)/(c - 1) + \epsilon \log_{|\Sigma|} c + \epsilon$, and $c = \epsilon^{-1} + \sqrt{1 + \epsilon^{-2}}$.

Using this bound and a covering list concept, the sublinear algorithm, SAM, was proved in the following lemma.

LEMMA 5. Given that the string of length N is the result of equi-probable Bernoulli trials and that $pow(\epsilon) < 1$, the approximate keyword search problem can be solved in expectation in $O(DN^{pow(\epsilon)} \log N + P)$ time.

Using the expected bound for the length of the longest repeat proved in section 2, we can extend the lemmas above concerning the size of $\overline{N_D}(W)$, $Pr(T, D)$ and the efficiency of the generation of $|\overline{N_D}(W)|$ for strings W.

LEMMA 6. For $T \approx 2 \cdot \log_{|\Sigma|} N$ and $D \leq T$, $|\overline{N_D}(W)| < 2N^{2pow(\epsilon)}$ and $Pr(T, D) < 2N^{2pow(\epsilon)-2}$ where $\epsilon = D/T$, $pow(\epsilon) = \log_{|\Sigma|}(c + 1)/(c - 1) + \epsilon \log_{|\Sigma|} c + \epsilon$ and $c = \epsilon^{-1} + \sqrt{1 + \epsilon^{-2}}$.

The proof of the lemma closely follows the original proof of Myers. With this information, the following lemma gives the expected number of supermaximal repeats.

THEOREM 7. For $T \simeq 2 \cdot \log_{|\Sigma|} N$ and $D \leq T$, in expectation, the number of supermaximal repeats(R) is $O(N^{(2pow(\epsilon)-1-d)})$ for some $d < 1$.

Proof.(Sketch) Note that the number of occurrences of any words w in the neighborhood of a given word W is at most $NPr(T, D)$. Then, the number $(\frac{N}{NPr(T,D)})$ of such words W in the database is less than $1/Pr(T, D)$, assuming that neighborhoods are kept disjoint. Hence, based on the results of Lemma 2 and definitions 3-4 of section 2, assuming that $E(D, W)$ neighborhoods are kept disjoint, the number of neighborhoods $R = O(1/Pr(T, D))$.

To overcome the assumption of disjointness, because the generation of words in the neighborhood of W is tailored towards an arbitrary word and does not take into consideration similarity or edit distance between words W, we give the following proof. Let $I_{EP}(\delta(W_1, W_2), |l_2 - l_1|)$ (henceforth I_{EP}) be the expected number of elements that form the intersection subset between any two extended repeat families $E(D, W_1)$ and $E(D, W_2)$. We estimate that $I_{EP} = c \cdot N^d$ for suitable constants c and $d \leq 1$. A more precise analysis of I_{EP} is deferred to the full version of the paper.

In expectation, the number of substrings of length T of string S where the substrings are counted more than once because they are in the intersection subsets is

$$N + \frac{R^2}{2} I_{EP}.$$

Hence, each word is counted $\frac{N+\frac{R^2}{2}I_{EP}}{N}$ times on average. Therefore, R can be estimated by

$$R = \frac{N}{NPr(T,D)} \cdot \frac{N+\frac{R^2}{2}I_{EP}}{N}.$$

Simplifying the above leads to a quadratic equation in the unknown variable R. That is,

$$I_{EP}R^2 - 2NPr(T, D)R + N = 0.$$

Solving this quadratic equation shows that $R \leq \frac{2NPr(T,D)}{I_{EP}}$. Thus,

$$R = O(N^{(2pow(\epsilon)-1-d)}) \text{ for some } d < 1.$$

□

Observing that $\xi(\hat{D}, D)$ is in fact a constant, $\xi(\hat{D}, D)R$ (which represents the expected number of maximal repeats) is also $O(N^{(2pow(\epsilon)-1-d)})$.

Extracting Significant Maximal Repeats

Next, we show how to use the approach given in the previous subsection to precisely compute all elements of M', the set of significant maximal repeats. The procedure to find significant maximal repeats FINDSIGMR(U, M) is encapsulated in Fig. 1 below, given the sets of supermaximal repeats (U) and maximal repeats (M) in a string of length N.

The first three sub-procedures in FINDSIGMR(U, M) implement the specification given in definitions 3-4 above, where we need to partition maximal repeats into classes, *i.e.*, $E(D, W)$'s, where the members (*i.e,* maximal repeats) of each class are proper substrings of the supermaximal repeat representing the class. Specifically, in sub-procedure IDCLASSMR(U, M), we concatenate supermaximal repeats, separated by special symbols for

1. procedure FINDSIGMR(U,M)
2. { $M \longleftarrow IDCLASSMR(U, M)$
3. Sort M
4. $E_i(D, W_i) \longleftarrow PARTMR(M), i = 1...R$
5. for $W_i, i = 1...R$ **do**
6. $M' \longleftarrow COLTSIGMR(E_i(D, W_i))$ }

Fig. 1. Finding maximal repeats whose lengths are less than $L - D$ and whose positions are not covered by a supermaximal repeat.

1. procedure FINDSAPPR(D, P, $U \cup M'$, N)
2. { **For** $W_i, i = 1...|U \cup M'|$ **do**
3. W_l,W_r $\longleftarrow$SPLITW(W_i)
4. H $\longleftarrow$SAM(D_l, W_l)
5. **for** each $H_i, i = 1...|H|$ **do**
6. SA $\longleftarrow$ADDWR(D_r, H_{ir}, P) }

Fig. 2. Finding short approximate repeats by extenping seeds.

recognition, to form a string and build a suffix tree for this string. We call this suffix tree the supermaximal repeats suffix tree (*SST*). The next task is to fish out the indices where each maximal repeat exists in *SST*. From these indices, we can find out which supermaximal repeat(s) contains a particular maximal repeat. Note that a maximal repeat may occur in more than one supermaximal repeats but, in this analysis, we make use of only one of its occurrences. In PARTMR(M), we partition maximal repeats into classes $E_i(D, W)$, $i = 1...R$. The last sub-procedure in FINDSIGMR(U, M) is COLTSIGMR($E_i(D, W)$), $i = 1...R$. This sub-procedure collects into M' all maximal repeats whose length are less than $L - D$ and whose positions are not covered by a supermaximal repeat.

LEMMA 8. Given the sets of supermaximal repeats (U) and maximal repeats (M) in a string of length N, the running time of FINDSIGMR(U, M) in expectation is $O(N^{2pow(\epsilon)-1-d} \log N)$ for some $d < 1$.

Proof. The procedure FINDSIGMR(U, M) include subprocedures IDCLASSMR(U, M), Sort M, PARTMR(M), and COLTSIGMR($E_i(D, W_i)$), $i = 1...R$.

The running time of IDCLASSMR(U, M) includes the time to concatenate supermaximal repeats to form a string and building a suffix tree, *SST*, in $O(N^{(2pow(\epsilon)-1-d)} \log_{|\Sigma|} N)$ and decide for each maximal repeat, for which supermaximal repeat is the maximal repeat a substring in $O(N^{2pow(\epsilon)-1-d} \log_{|\Sigma|} N)$ for some $d < 1$.

The time to sort M and for PARTMR(M) is at most $O(N^{2pow(\epsilon)-1-d} \log_{|\Sigma|} N)$ for some $d < 1$.

To estimate the running time of COLTSIGMR($E_i(D, W_i$)), $i = 1...R$, requires that we know how many maximal repeats are in each supermaximal repeat based class $E(D, W)$. Extending the proof of Lemma 6 and using the estimation for the length of supermaximal repeat, a technical analysis shows that the running time of COLTSIGMR($E_i(D, W_i)$), $i = 1...R$, is

$$O(N^{2pow(\epsilon)(\log_N \log_{|\Sigma|} N^2+1)-3-d} \log N) \text{ for some } d < 1.$$

Noting the dominating term, we can write that the running time of FINDSIGMR(U, M) is $O(N^{(2pow(\epsilon)-1-d)} \log N)$ for some $d < 1$. □

Procedure FINDSAPPR($D, P, U \cup M', N$) finds all short approximate repeats using SAM that are obtainable by expanding seeds in the set $U \cup M'$. Noting that $|U \cup M'|$ is the number of seeds found in string S and $|H|$ is the number of locations returned into H, FINDSAPPR($D, P, U \cup M', N$) and its complexity analysis are described below.

We assume here that $P = c \cdot \log N$ for a constant $c > 0$. If $c \leq 1$, a direct application of SAM solves the problem. For $c > 1$, we do the following:

Basically, given D, for each seed W, we split W into a left and right part W_l and W_r, where $|W_l| = \log_{|\Sigma|} N$. Next, using SAM, we find all locations of all approximate repeats W_l' of W_l. Their right indices are stored in set H. Next, we check whether we can extend W_l' to the right to get approximate repeat for $W = W_l W_r$. Therefore, for every index H_i in H corresponding to a word W_{li} with differences D_i, we consider the word W_{ri} starting from H_i and having length $P - \log_{|\Sigma|} N$. Let E_W denote the set of possible rightward extensions of W_r to a word of length $P - \log_{|\Sigma|} N$. Note that $|E_W|$ is the number of occurrences of the seed W in string S, which is, fortunately, at most $|\Sigma|$ for supermaximal repeats and even constant in expectation, if we restrict W to be of length approximately $2 \log N$. Therefore, $|E_W|$ will be neglected in the following analysis. Furthermore, note that for $c = 2$, no extension is necessary. We just take W_r.

We compute the pairwise alignments of W_{ri} with all elements E_j of E_W and, if the difference between two strings is at most $D_i' = D - D_i$, then the extension works and we can add $W_i = W_{li} W_{ri}$ to the output list SA for the short approximate repeats. We call this procedure ADDWR(D_i', L_i, P). A pseudo-code for FINDSAPPR is given in Fig. 2.

LEMMA 9. Given the sets of seeds $U \cup M'$, where the length of each seed is approximately $2 \log_{|\Sigma|} N$, all short

1. **procedure** LISTSAPPR(D, P, N)
2. { build a suffix tree for S
3. S, M $\longleftarrow$FINDSMR(N)
4. //Compute$\hat{D}(D)$//
5. M' $\longleftarrow$FINDSIGMR(U, M)
6. FINDSAPPR(D, P, $U \cup M'$, N) }

Fig. 3. The complete algorithm for finding all short approximate repeats.

approximate repeats of size $O(\log N)$ can be found in an $O(DN^{(3pow(\epsilon)-1)} \log N)$ expected time algorithm for $D \leq 2\log_{|\Sigma|} N$.

Proof. The subprocedure SPLITW can be done for each seed in constant time. SAM running time for each seed is $O(DN^{pow(\epsilon)} \log N)$.

Note that using Lemma 3, the size of indices in set L is at most $O(N^{pow(\epsilon)})$. Therefore, since in expectation the length of W_r is $\log_{|\Sigma|} N$, ADDWR requires for all indices in L at most $O(N^{pow(\epsilon)} \log^2_{|\Sigma|} N)$ time. Hence, the running time of FINDSAPPR(D, $U \cup M'$, N) for all seeds where $|U \cup M'|$ is at most $O(N^{2pow(\epsilon)-1})$ is at most $O(DN^{3pow(\epsilon)-1} \log N)$, at least for all practical values of N and Σ. □

Using all of the details above, the algorithm LISTSAPPR(D, P, N) that lists all short approximate repeats and its expected time is encapsulated in Fig. 3 and the theorem below.

THEOREM 10. Given a string of length N, the expected time require to find all short approximate repeats is $O(N + DN^{3pow(\epsilon)-1} \log N)$ for $D \leq 2\log_{|\Sigma|} N$.

Proof. The time to build a suffix tree for string S and find supermaximal and maximal repeats in a string of length N, which is done in FINDSMR(N), is $O(N + m)$, where m is the number of the maximal exact repeats found. Note that we have shown in Lemma 7 that $m = \xi R = \Theta(N^{(2pow(\epsilon)-1-d)})$ for some $d < 1$. The supermaximal and maximal repeats are output into sets U and M.

In procedure FINDSAPPR(D, P, $U \cup M'$, N), we need the significant approximate repeats as seeds. As noted earlier on, to locate maximal repeats whose length are less than $L - D$ and whose positions are not covered by a supermaximal repeat, we make use of procedure FINDSIGMR(U, M) and store these significant exact maximal repeats in variable M'. The expected running time of FINDSIGMR(U, M) is given in Lemma 8.

Note that the running time of FINDSAPPR(D, P, $U \cup M'$, N) proved in Lemma 9 is within the bound of the theorem. Therefore, the overall expected time for procedure LISTSAPPR(D, P, N) is $O(N + DN^{(3pow(\epsilon)-1)} \log N)$. □

PRACTICAL EXPERIMENTS

We have implemented our algorithm in C to determine its practical efficiency. We find the supermaximal and maximal repeats using suffix tree. Based on some practical observations, we implemented FINDSIGMR(U, M) using only one of the two filtering constraints for significant maximal repeats discussed before. Recall that from the set of all maximal repeats, we extract the maximal repeats whose length are less than $L - D$, where L is the length of the longest supermaximal repeat, and whose positions are not covered by a supermaximal repeat. We found in practice that the first condition is sufficient to extract reasonable maximal repeats (see Table 1, column 4, #max sig). To avoid the multiple extension of the same approximate repeats from a supermaximal repeat (W) and the maximal repeats in $E(D, W)$, we simply ignore using as seed an instance occurrence of a maximal repeat whose left boundary is at most D away from the left boundary of W; this combination is faster in practice (see Table 1, column 5, for the number of maximal repeats #max usd, used as seed by our algorithm on some genomes). In procedure FINDSAPPR(D, P, $U \cup M'$, N), we direct the search for an approximate repeats in the extension phase in subprocedure ADDWR(D_r, H_{ir}, P) using the idea of progressive computation of the corresponding rows of the dynamic programming matrix. This idea is also used by (Myers, 1994) to tackle the problem of generating the words in the condensed D-neighborhood of a word. We also extracted maximal repeats using the filtering constraint given by Kurtz *et al.* (Kurtz et al., 2000). This constraint is stated in the lemma below:

LEMMA 11. *(Kurtz et al., 2000)* Every maximal D-difference repeat R of length P contains a maximal exact repeat of length $\geq \lfloor \frac{P}{D+1} \rfloor$, which is used as a seed.

Note that this notion of seeds is different from the one we used.

We report some test results on our algorithm, in comparison with that of Kurtz *et al.*, on some real genomes taken from the NCBI database in the following table. Because our algorithm, partially, and the algorithm of Kurtz *et al.* run in time proportional to the number of seeds, we focus foremost on the number of seeds required by each algorithm. Note that we do not need to consider all supermaximal or maximal repeats, but just those of lengths larger than or equal to $\lceil 2 \cdot \log_{|\Sigma|} N \rceil$ (see Table 1, column 4). This shows the significance of our theoretical estimation of $2 \cdot \log_{|\Sigma|} N$ as the expected maximum length of maximal repeats.

Table 1. Experimental results on the number of supermaximal, maximal repeats, the number of seeds and the number of approximate repeats found with the corresponding running time used. In this table, $2 \cdot \log_{|\Sigma|} N < P \leq 30$ and $D \leq 4$. The number in the second column is the number of bases in each genome in megabases (MB). Column 3 give the numbers of supermaximal and maximal repeats in each genome and the total number of supermaximal and maximal repeats used as seeds in REPfind. Column 4 shows the total number of supermaximal used as seeds in our algorithm. #max sig is the number of maximal repeats, whose lengths are $\geq \lceil 2 \cdot \log_{|\Sigma|} N \rceil$. On the long run, the number of maximal repeats used in our algorithm as seeds is given in column 5. Column 6 and 7 give the number of approximate repeats found in each genome and the running time in seconds used to find them. Our program was run on a SUN-Sparc computer under Solaris 2.5.1 with a 366 MHz-Processor and 360Mbytes of main memory.

		REPfind	Ours	Ours	Ours	Ours
Genome	N (MB)	#sup usd / #max usd	#sup usd / #max sig	#max usd	#appr.rep	run time (sec)
pNGR234	.46	90996 / 37471	442 / 0	0	329	1003.35
Mgen	.50	95008 / 38055	380 / 3	2	513	867.61
Uure	.64	120060 / 46417	202 / 1	0	201	525.51
Mpneu	.70	132181 / 54932	944 / 16	10	1629	1914.66
Bbur	.78	152881 / 59800	105 / 0	0	94	260.63
Ctra	.89	187080 / 76619	92 / 0	0	91	214.44
CtraM	.92	192460 / 77997	63 / 0	0	66	155.68
Rpxx	.95	191575 / 75925	54 / 0	0	29	215.73
Tpal	.98	203499 / 83244	245 / 1	0	260	613.84
CpneuA	1.05	220777 / 89634	183 / 0	0	201	576.63
Cpneu	1.05	219967 / 90282	124 / 2	1	128	373.94
Cjej	1.41	266843 / 105161	629 / 0	0	482	2617.57
Aaeo	1.43	295649 / 119698	183 / 13	3	325	683.25
Mjan	1.43	277985 / 107694	574 / 39	19	993	2609.25
Hpyl	1.43	276973 / 111685	609 / 1	1	674	2263.31
Mthe	1.50	305620 / 124248	577 / 46	14	865	2086.84
Pabyssi	1.51	317188 / 128042	191 / 8	3	167	1143.23
Hinf	1.57	314207 / 127524	795 / 5	3	964	3038.26
Aful	1.9	383030 / 155826	430 / 27	7	603	1848.13
Drad1	2.27	434270 / 170850	751 / 22	14	1398	3385.79
Synecho	3.06	626017 / 255746	887 / 22	15	1577	4176.41
Bsub	3.61	752332 / 305426	818 / 1	1	1251	4160.21
Mtub	3.78	744838 / 300892	1795 / 64	37	4368	13001.30
Ecoli	4.6	934209 / 380851	683 / 54	36	1488	4892.36

Column 6 and 7 of the table below shows the number of short approximate non-tandem repeats found in each genome we considered and the running time in seconds that our algorithm spent to find them. RePfind compute also all approximate, non-tandem repeats, but uses every available seeds (supermaximal and maximal repeats) to do so. As we stated earlier on, this is practical for finding long approximate repeats but quite impractical for find short approximate repeats. We ran REPfind on genome pNGR234. It used 80552.73 secs $\approx$ 22 hours to find the approximate repeats, whose lengths are between 19 and 30, while $D \leq 4$. We ran also REPfind on genomes Mgen, Uure, Mpneu and Bbur. We broke the operation of REPfind on each genome after it ran for 10 hours. The other genomes even have more maximal repeats such that REPfind is expected to have an even higher runtime in these cases. The foregone arguments clearly conclude the practical efficiency of our approach.

We give some observations based on the experiments:

1. For finding short approximate repeats, the algorithm of Kurtz*et al.* requires a lot more seeds than ours, see columns 3 and 4 of the table above. Therefore, our algorithm is much more efficient.
2. In the extension phase, the algorithm of Kurtz *et al.* actually considers pairs of maximal repeats, different from our approach. Hence, their running time might actually grow quadratic in the number of maximal repeats, but our algorithm's running time is a linear function of the number of supermaximal repeats.
3. Their algorithm might consider the same approximate repeats of a specific length up to D times, namely if it contains D exact repeats of appropriate length. Our method avoids this by extending the seeds only in one direction.

One would expect the running time of our algorithm to increase as the size of the genome increase as it occurred in REPfind. Instead it depends majorly on the number of seeds found in each genome, which does not always

increase as the genome size increases. There is a natural explanation for this. For example, the number of genes in a genome does not always increase with the genome size, as evidenced in genomes pNGR234 and Pabyssi or Mtub and Ecoli. The second factor on the running time of our algorithm is the number of instances, $|E_W|$, of a maximal repeat used as seed. This also does not depend in any way on the size of the genome, as observed in genomes pNGR234 and Mgen for example. The last factor is the total number of elements of set H considered for extension in the attempt to extend a seed to an approximate repeat. This is the prominient factor on the running time of our algorithm on genomes Mpneu and Cjej, where we used 1914.66 secs to find 1629 approximate repeats in one genome but used 2617.57 secs to find only 482 approximate repeats in the other. These are some of the dynamic attributes of our algorithm that makes it efficient in practice.

CONCLUSION

We proved a significantly sub-quadratic algorithm for finding non-tandem approximate short repeats. The improved performance of the algorithm is not only theoretically proved, but also shown to be true in practice. This is because of the small number of seeds the algorithm chooses to use. Our theoretical characterization shows that such a small number of seeds are enough for finding all short approximate repeats. Several further improvements are possible as observed in the comparision of our work with that of Kurtz et al. (Kurtz et al., 2000), and will be attempted in our future work.

Acknowledgments

We thank Jop Sibeyn for some discussions on the analysis of the expected length of the longest repeat. We also thank Stefan Kurtz and Gene Myers for providing us with their program codes for building the suffix tree, for the executable codes of REPfind, and for mailing the code of the sublinear algorithm, which we named SAM in this paper. Ezekiel F. Adebiyi was supported by DAAD Scholarship Grant. Tao Jiang has been partially supported by a UCR startup grant and NSF grants CCR-9988353 and ITR-0085910.

REFERENCES

Bairoch, A. (1992). Prosite: a dictionary of sites and patterns in proteins. *Nucleic Acids Res. 20(Suppl)*, 2013–2017.

Blumer, A., A. Ehrenfeucht, et al. (1989). Average size of suffix trees and dawgs. *Discrete Applied Mathematics 24*, 37–45.

Delcher, A., S. Kasif, et al. (1999). Alignment of whole genomes. *Nucleic Acids Research 27(11)*, 2369–2376.

Gusfield, D. (1997). Algorithms on strings, trees and sequences. *Cambridge University Press, New York*.

Hodgman, T. (1989). The elucidation of protein function by sequence motif analysis. *Comput. Applic. Biosci. 5*, 1–13.

Kurtz, S., E. Ohlebusch, et al. (2000). Computation and visualization of degenerate repeats in complete genomes. *ISMB Conf.*, 228–238.

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

Myers, E. (1994). A sub-linear algorithm for approximate keyword matching. *Algorithmica 12(4-5)*, 345–374.

Sagot, M.-F. (1998). Spelling approximate repeated or common motifs using a suffix tree. *LNCS 1380*, 111–127.

TIGR (1999). Repeat-finder. *http://www.tigre.org/tdb/rice /repeatinfo-MUMmer.shtml*.

Ukkonen, E. (1985). Algorithms for approximate string matching. *Information and Control 64*, 100–118.