BIOINFORMATICS

Vol. 20 no. 16 2004, pages 2702–2710
doi:10.1093/bioinformatics/bth311

Exhaustive whole-genome tandem repeats search

Arun Krishnan and Francis Tang*

Bioinformatics Institute, 30 Biopolis Street, No. 07-01, Matrix, Singapore 138671, Singapore

Received on February 3, 2004; revised and accepted on April 22, 2004
Advance Access publication May 14, 2004

ABSTRACT

Motivation: Approximate tandem repeats (ATR) occur frequently in the genomes of organisms, and are a source of polymorphisms observed in individuals, and thus are of interest to those studying genetic disorders. Though extensive work has been done in order to identify ATRs, there are inherent limitations with the current approaches in terms of the number of pattern sizes that can be searched or the size of the input length.

Results: This paper describes (1) a new algorithm which exhaustively finds all variable-length ATRs in a genomic sequence and (2) a precise description of, and an algorithm to significantly reduce, redundancy in the output. Our ATR definition is parameterized by a mismatch ratio p which allows for more mismatches in longer tandem repeats (and fewer in shorter). Furthermore, our algorithm is embarrassingly parallel and thus can attain near-linear speed-up on Beowulf clusters. We present results of our algorithm applied to sequences of widely differing lengths (from genes to chromosomes).

Availability: Source and binaries are available on request.

Contact: arun@bii.a-star.edu.sg

Supplementary information: http://web.bii.a-star.edu.sg/~francis/Research/Exhaustive/

INTRODUCTION

Informally, a tandem repeat is defined to be two adjacent (approximate) copies of the same sequence of nucleotides. The occurrence of several adjacent copies is referred to as a variable-length 'approximate' tandem repeat (ATR). Tandem repeats range from micro- and mini-satellites (few tens of base pairs) to larger satellite repeats spanning megabases. They occur frequently in the genomes of organisms. Their function and origins are not truly understood, though it has been proposed that they are the result of a mutational event resulting from the exact duplication of a stretch of DNA, followed by some random mutations over time (Benson, 1999).

What is known is that such repeats can be a source of polymorphism observed in individuals. Thus there are active investigations into potential correlations between lengths of tandem repeats and genetic diseases [e.g. attention deficit hyperactivity disorders (Qian *et al*., 2003), multiple sclerosis (Guerini *et al*., 2003), Alzheimer's (Licastro *et al*., 2003), Autism (Cohen *et al*., 2003) and androgen insensitivity syndrome (Fogu *et al*., 2003)].

A lot of work has been done in order to identify tandem repeats, both exact and approximate. From the literature on tandem repeat searching programs, we find that the solutions fall broadly into two categories: those that define tandem repeats in a way that leads to a simple constructive characterization allowing for an algorithm to find all tandem repeats directly, and those that use a more abstract definition of tandem repeat, e.g. a probabilistic definition, and which give a heuristic for narrowing down the set of potential tandem repeats.

The following approaches fall into the first category. Collins *et al*. (2003) give a vectorizable algorithm to find tandem repeats. Their algorithm finds exact tandem repeats (i.e. the copies must be exactly the same) of short pattern lengths (2–16).

Sagot and Myers (1998) present an exhaustive algorithm which finds ATRs defined by a fixed edit distance εk from a consensus sequence. The performance can be improved by pre-filtering using a heuristic. The asymptotic performance is bounded by $O(N\mathcal{N}(\varepsilon k, k))$, where $\mathcal{N}(e, k)$ is the combinatorial number of k-length strings within a ball of radius e.

REPuter (Kurtz *et al*., 2001) addresses the more general problem of finding repeats (not necessarily tandem). Their algorithm can exhaustively find ATRs using either Hamming distance or edit distance definitions. The Hamming distance solution has $O(N + z\varepsilon)$ complexity, where N is the sequence length, ε is the distance parameter and z is the 'number of seeds', the estimate of which is bounded by $O(N^2/|\Sigma|^s)$, where $s = \lfloor(1 + \varepsilon)^{-1}\rfloor$. Since $|\Sigma| > 1$, this asymptotic bound tends to $O(N^2)$ as the mismatch ε tends to infinity.

The approaches of Sagot and Myers (1998), and REPuter use a fixed distance ε, regardless of the pattern length: a small

*To whom correspondence should be addressed.

ε would be too stringent for long pattern lengths, and a large ε would be too lenient for short pattern lengths. Of course, it is possible to run their algorithms repeatedly for different ε, but this gives worse overall performance. Our basic algorithm finds all tandem repeats of a specific pattern length k in $O(kN)$ time, independent of distance bound ε; thus, we can search all possible k (i.e. for $k = 2..\lfloor N/2 \rfloor$) in $O(N^3)$ time.

Kolpakov and Kucherov (2003), Landau *et al*. (2001), Groult *et al*. (2002, 2004) also use Hamming distance-based definitions of ATR. Groult *et al*. consider a strictly more general problem, but their solution is asymptotically slower than the one presented here. A more detailed comparison of the two can be found in the Supplementary information.

Falling in the latter category, Benson (1999) proposes a probabilistic definition of tandem repeat which uses a model whereby it is assumed that an ATR was, at some point in the past, an exact repeat that has undergone mutations (in the form of nucleotide substitutions, insertions and deletions). However, such a probabilistic definition of tandem repeat is difficult to use directly: one must find candidates to check against the definition. The set of all candidates is too large, and so must be pruned to a set of likely candidates by way of a heuristic. In loc. cit., Benson presents a heuristic which uses k-tuples.

By definition, a heuristic is incomplete and a fine balance must be made between completeness and performance. Wexler *et al*. (submitted for publication) present a different heuristic which experimentally is shown to improve on that of Benson, at the expense of overall run time.[1] Similarly, Stolovitzky *et al*. (2001) apply a pattern discovery tool TEIRESIAS to find their candidates.

We find that existing literature puts little emphasis on the problem of filtering of tandem repeats. The definitions of tandem repeat used by Benson (1999), Kolpakov and Kucherov (2003), Landau *et al*. (2001) can span a non-integral number of copies, and so maximality already reduces/eliminates redundancy. There is no explicit discussion of filtering in Collins *et al*. (2003), Kurtz *et al*. (2001), Wexler *et al*. (submitted for publication), Stolovitzky *et al*. (2001), Groult *et al*. (2002, 2004). In Sagot and Myers (1998), overlapping repeats (which they call trains) are collected together and the fittest member is selected from each collection.

This article describes the algorithmic aspects of our program which exhaustively finds all variable-length tandem repeats in a genomic sequence. Our approach falls in the former category: we take a definition of tandem repeat that uses the Hamming distance measure, and present an algorithm that provably finds all tandem repeats, i.e. an exhaustive algorithm. Our definition is parameterized by a mismatch ratio p which allows for more mismatches in longer tandem repeats (and fewer in shorter), thereby avoiding the problems of a fixed mismatch count. Additionally, we present a filtering algorithm that prunes the resultant exhaustive set to a smaller one with fewer redundancies. Furthermore, our algorithms are embarrassingly parallel and well suited for Beowulf-class clusters. As a result, the algorithm can be applied to whole genomes and for any pattern size.

[1] However, since it finds more tandem repeats, they show that their new heuristic is faster per tandem repeat than that of Benson's.

METHODS

DEFINITION 1. (*Candidate*) *Given a string s, a tuple $(x, \alpha^{(1)}, \ldots, \alpha^{(l)})$, consisting of an offset*[2] *x and strings $\alpha^{(i)}$, is said to be a candidate, if, for $i = 1 \ldots (l-1)$*

(1) $|\alpha^{(i)}| = |\alpha^{(i+1)}|$ *and*

(2) $\alpha^{(i)}$ *is equal to the substring of length* $|\alpha^{(1)}|$ *starting at* $x + (i-1) \cdot |\alpha^{(1)}|$.

We let C denote the set of all candidates:

$$C \stackrel{\text{def}}{=} \{(x, \alpha^{(1)}, \ldots, \alpha^{(l)}) \mid x, \alpha^{(i)} \text{ satisfy the two conditions above}\}.$$

For each non-negative integer k, we define C_k to be the subset of C consisting of those candidates of pattern length k:

$$C_k \stackrel{\text{def}}{=} \{(x, \alpha^{(1)}, \ldots, \alpha^{(l)}) \in C \mid k = |\alpha^{(1)}|\}.$$

It follows that $|\alpha^{(1)}| = |\alpha^{(2)}| = \cdots = |\alpha^{(l)}|$, from the definition of C. A tandem repeat is a candidate $(x, \alpha^{(1)}, \ldots, \alpha^{(l)})$ such that the strings $\alpha^{(i)}$ are approximate copies of each other. We define 'approximate copy' using Hamming distance,[3] $D(-,-)$, by:

DEFINITION 2. (*Approximate tandem repeat*) *For string s, and parameter p, an approximate tandem repeat is a candidate $(x, \alpha^{(1)}, \ldots, \alpha^{(l)})$ such that $D(\alpha^{(i)}, \alpha^{(i+1)}) \leq \lfloor p.|\alpha^{(i)}| \rfloor$ for all positive $i < l$.*

When there is no ambiguity, we write 'repeat' to mean approximate tandem repeat. We let R denote the set of all tandem repeats. That is

$$R \stackrel{\text{def}}{=} \{(x, \alpha^{(1)}, \ldots, \alpha^{(l)}) \in C \mid D(\alpha^{(i)}, \alpha^{(i+1)}) \leq \lfloor p \cdot |\alpha^{(i)}| \rfloor \text{ for } 1 \leq i < l\}.$$

For each non-negative integer k, we define R_k to be the set of tandem repeats of pattern length k:

$$R_k \stackrel{\text{def}}{=} R \cap C_k.$$

Note that we simply insist that adjacent copies are pair-wise approximate copies. One can consider stronger definitions of

[2] For the purposes of this paper, we assume that the beginning of the string has offset 1.

[3] Recall for equal length strings u and v, the Hamming distance $D(u, v)$ is defined to be the number of mismatches, i.e. the number of unique indices i such that $u_i \neq v_i$.

approximate copy, for example, we might insist

$$D(\alpha^{(i)}, \alpha^{(j)}) \leq \lfloor p \cdot |\alpha^{(i)}| \rfloor$$

for all positive $i, j \leq l$. This is a strictly stronger definition since any candidate satisfying this definition also satisfies that of Definition 2, but not vice versa. However, the latter definition is clearly computationally more expensive to check. We settle for the weaker definition since for typical choices of p, the difference in results is not significant.

Algorithm

Searching for general tandem repeats We know from the definition of a tandem repeat that for any pattern size k, $(x, \alpha^{(1)}, \ldots, \alpha^{(l)})$, is a tandem repeat, if and only if $(x + k\cdot(i-1), \alpha^{(i)}, \alpha^{(i+1)})$ is a tandem repeat, for $i = 1, \ldots, (l-1)$. We define a vector T as follows: entry $T[j]$ is the number of matches between subsequences u and v for the candidate (j, u, v). That is,

$$T[j] = k - D(u, v) \quad \text{for candidate } (j, u, v) \tag{1}$$

For each offset j, starting from the beginning, the algorithm eagerly finds maximal tandem repeats, i.e. tandem repeats that cannot be extended to the left or right to give another tandem repeat, by comparing successive candidates $(j, \alpha^{(1)}, \alpha^{(2)})$, $(j + k, \alpha^{(2)}, \alpha^{(3)})$, $\ldots$, $(j + (l-1)k, \alpha^{(l-1)}, \alpha^{(l)})$. For each offset at which two candidates are compared, we update $T[\cdot]$.

For any candidate (j, u, v), in general, the calculation of $T[j]$ requires $|u|$ comparisons. However, suppose we already know $T[j-1]$. By definition, since u is the substring of S, starting at x, we know that $u_i = S_{x+i-1}$. We have a similar definition for v, namely $v_i = S_{x+k+i-1}$. Now consider candidates (x, u, v) and $(x+1, u', v')$. By referring to the definitions of $D(u, v)$ and $D(u', v')$, and the facts that $u'_i = S_{i+x} = u_{i+1}$ and $v'_i = S_{i+k+x} = v'_{i+1}$, we can derive

$$T[j] = T[j-1] - \delta_0 + \delta_k$$

where

$$\delta_0 \stackrel{\text{def}}{=} \begin{cases} 1 & \text{if } u_1 = v_1 \\ 0 & \text{otherwise} \end{cases}$$

$$\delta_k \stackrel{\text{def}}{=} \begin{cases} 1 & \text{if } u'_k = v'_k \\ 0 & \text{otherwise} \end{cases}$$

Thus $T[j]$ can be computed from $T[j-1]$ using only two more comparisons. So, to take advantage of this observation, we also maintain vector V to remember which entries of $T[\cdot]$ have been computed:

$$V[j] = \begin{cases} 1 & \text{if candidate } (j, u, v) \text{ has been checked} \\ 0 & \text{otherwise} \end{cases}$$

This allows us to use the optimization when we have already computed $T[j-1]$, and allows us to avoid computing $T[j]$ more than once.

The algorithm is presented in its entirety in the supplementary information, but, to demonstrate, we present it here with an example. Let $p = 0.25$ and $k = 4$. Consider the following sequence

ATAA AGAA ATAA GTAA GT

(Spaces have been inserted to separate the letters into blocks of four for ease of reading.) We build vectors T and V as follows. We start with $j = 1$. Then, $T[1] = 3$, since $D($ATAA, AGAA$) = 1$. As $T[1] \geq \lceil (1-p)k \rceil = 3$,[4] we try to extend this tandem repeat to the right. Hence, we skip ahead to $j = j + k = 5$ and calculate $T[5] = 3$. Since this again meets the criterion for being accepted as a tandem repeat, we extend this to the right ($j = j + 2k = 9$) and calculate $T[9] = 3$. However, we cannot extend this to the right anymore (having reached the end of the sequence) and hence accept the tandem repeat (1, ATAA, AGAA, ATAA, GTAA). We set $V[j] = 1$ for all j visited by the algorithm. Hence, in the previous cases $V[1] = V[5] = V[9] = 1$.

We next assign $j = 2$ and calculate $T[2] = 3, V[2] = 1$. Note that since in this case, $T[1]$ has already been calculated, calculation of $T[2]$ requires only two comparisons (the first and the last). Since this satisfies the condition for $(2, u, v)$ to be a tandem repeat, we try and extend this to the right by setting $j = j + k = 6$ and calculate $T[6] = 2$. Since this does not satisfy the criterion for $(6, u', v')$ to be a tandem repeat, we accept the tandem repeat (2, TAAA, GAAA). V is updated in the same way as before.

Continuing in the same manner, we calculate $T[3] = 3$ and, extending to the right again, we calculate $T[7] = 3$; $T[11] = 4$. Since we cannot extend this any further, we again accept (3, AAAG, AAAT, AAGT, AAGT). Similarly, we calculate $T[4] = 3$ and extend it to $T[8]$ and $T[12]$ (note that we cannot compute this as $j > N - 2k + 1$ for $j = 12$), and accept (4, AAGA, AATA, AGTA) as a repeat. Again, it must be noted here that these involve only two comparisons per offset since $T[j-1]$ has already been calculated while calculating $T[j]$. We then skip $T[5]$, $T[6]$, $T[7]$, $T[8]$ and $T[9]$ since these have already been computed and calculate $T[10] = 4$. However, since we cannot extend this to the right, we accept (10, TAAG, TAAG) as a repeat, and skip $T[11]$. We stop the calculations at this point, since $j > N - 2k + 1$ for $j = 13$. The computed vectors T and V and the procedure are summarized in Figure 1.

Using T, we accept the following repeats: (1, ATAA, AGAA, ATAA, GTAA), (2, TAAA, GAAA), (3, AAAG, AAAT, AAGT, AAGT), (4, AAGA, AATA, AGTA) and (5, TAAG, TAAG). Note that the repeats are stored in start ascending order; this is a required assumption for correctness of the filtering algorithm to follow.

[4] From Definition 2 and Equation (1), (j, u, v) is a repeat iff $T[j] \geq \lceil (1-p)k \rceil$.

(a)

j	$T[j]$	$V[j]$	Accepted Repeats
1	3	1	ATAA, AGAA, ATAA, GTAA
2	0	0	
3	0	0	
4	0	0	
5	3	1	-
6	0	0	
7	0	0	
8	0	0	
9	3	1	-
10	0	0	
11	0	0	
12	0	0	

(b)

j	$T[j]$	$V[j]$	Accepted Repeats
1	3	1	ATAA, AGAA, ATAA, GTAA
2	3	1	TAAA, GAAA
3	0	0	
4	0	0	
5	3	1	-
6	2	1	!
7	0	0	
8	0	0	
9	3	1	-
10	0	0	
11	0	0	
12	0	0	

(c)

j	$T[j]$	$V[j]$	Accepted Repeats
1	3	1	ATAA, AGAA, ATAA, GTAA
2	3	1	TAAA, GAAA
3	3	1	AAAG, AAAT, AAGT, AAGT
4	3	1	AAGA, AATA, AGTA
5	3	1	-
6	2	1	!
7	3	1	-
8	3	1	-
9	3	1	-
10	4	1	TAAG, TAAG
11	4	1	-
12	0	1	!

Fig. 1. The growth of vector T for example. The 'minus' indicates that the corresponding offset is part of a larger repeat. The 'exclamation' indicates that no tandem repeat could be found at the corresponding offset. Figure 1a–c denote the state of the vector and the repeats found after each offset ($j = 1, 2$ and 10 respectively).

Filtering

The algorithm presented so far finds all maximal tandem repeats. Now consider the example sequence ACGACGAG, which has three maximal repeats ACGACG, CGACGA and GACGAG. Clearly it is unnecessary to list all three, since most of the information is redundant. We now consider filtering of repeats.

DEFINITION 3. (*Overlap*) *Given two candidates* $\alpha = (x, \alpha^{(1)}, \alpha^{(2)}, \ldots, \alpha^{(l)})$ *and* $\beta = (y, \beta^{(1)}, \beta^{(2)}, \ldots, \beta^{(m)})$, *define* c *as the cardinality of the set of offsets that are common to* α *and* β. *Then* α *and* β *are said to overlap if* $c \geq k \cdot min\{l, m\} - k$.

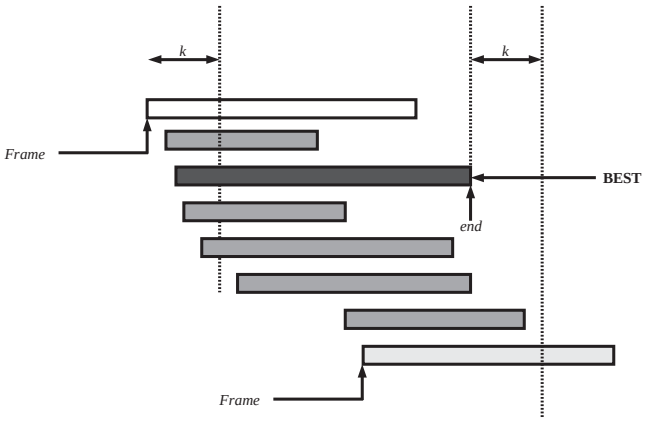


Fig. 2. Schematic representation of the filtering process.

DEFINITION 4. (*Score*) *For any given candidate* α, *let* B *define the set of offsets at which the repeats* $\alpha^{(1)}, \alpha^{(2)}, \ldots, \alpha^{(l)}$ *start, i.e.* $B = \{x, x+k-1, x+2k-1, \ldots, x+(l-1)k-1\}$. *Then, the score* s_x, *for a given candidate* α *is defined as the total number of matches, i.e.* $s_x = \sum_{j \in B} T(j)$.

We can then define the notion of 'better', for overlapping candidates α, β, as the lexicographical ordering formed by first comparing their lengths (i.e. l vs. m) and then their scores (i.e. s_x versus s_y). Redundancy among candidates can now be defined as follows:

DEFINITION 5. (*Redundant*) *A candidate* α *is said to be redundant with respect to a candidate* β *if and only if* α, β *overlap, and* β *is better than* α *with respect to the lexicographical ordering of their lengths and scores.*

Alternatively, and equivalently, candidates α and β are said to be redundant if α is easily recoverable from β. It is clearly a desirable property that any filtering algorithm should be conservative in the sense any repeat filtered out can be easily recovered from the remaining repeats. More formally, we define the safety property as follows:

DEFINITION 6. (*Safety*) *Given a set of repeats* R'_k. *A set* $A \subset R'_k$ *is said to fulfill the safety property, if given any* $r \in R'_k$ *we can find some* $r' \in A$ *such that* r *is redundant with respect to* r'.

A schematic representation of the filtering algorithm is shown in Figure 2. As before, we present the filtering (Fig. 3) with the example from the previous section. Given the sequence

$$S = \text{ATAA AGAA ATAA GTAA GT}$$

we filter the repeats as follows (we only show those offsets for which the algorithm finds accepted repeats).

Given an offset x, we introduce the notion of a frame as a set of repeats that start within k of the reference x. The algorithm

j	s_j	Considered Repeats
1	9	ATAA, AGAA, ATAA, GTAA
2		
3		
4		
10		

(a) $j = 1$

j	s_j	Considered Repeats
1	9	ATAA, AGAA, ATAA, GTAA
2	3	~~TAAA, GAAA~~
3		
4		
10		

(b) $j = 2$

j	s_j	Considered Repeats
1	9	~~ATAA, AGAA, ATAA, GTAA~~
2	3	~~TAAA, GAAA~~
3	10	AAAG, AAAT, AAGT, AAGT
4		
10		

(c) $j = 3$

j	s_j	Considered Repeats
1	9	~~ATAA, AGAA, ATAA, GTAA~~
2	3	~~TAAA, GAAA~~
3	10	AAAG, AAAT, AAGT, AAGT
4	6	~~AAGA, AATA, AGTA~~
10		

(d) $j = 4$

j	s_j	Considered Repeats
1	9	~~ATAA, AGAA, ATAA, GTAA~~
2	3	~~TAAA, GAAA~~
3	10	AAAG, AAAT, AAGT, AAGT
4	6	~~AAGA, AATA, AGTA~~
10	4	~~TAAG, TAAG~~

(e) $j = 10$

Fig. 3. The filtering process for the example sequence: the striken repeats indicate those that have been discarded. Figure 3a–e denote the state of the vector s_j and the filtered repeats found after each offset ($j = 1, 2, 3, 4$ and 10 respectively).

essentially finds the best repeat within each frame, that makes every other repeat in that frame redundant. We start with offset $j = 1$ as our reference offset and initially accept the first repeat we find, viz. (1, ATAA, AGAA, ATAA, GTAA) as our best repeat in that frame (Fig. 3a). We store the value of $s_1 = 9$ and also remember the offset for this repeat ($j = 1$). We then consider the next repeat (offset $j = 2$) and compare this with the previous repeat using the lexicographical ordering defined earlier. Since the number of repeats for offset $j = 2$ is two as compared to four for $j = 1$, we discard the repeat (2, TAAA, GAAA) (Fig. 3b). We then continue with offset $j = 3$ and obtain (3, AAAG, AAAT, AAGT, AAGT) and $s_3 = 10$. Since the offset in this case is also within k of the offset for the reference repeat, we compare the two repeats using the lexicographical ordering. We find that the number of repeats is the same in both cases and hence compare the scores for the two repeats. Since (3, AAAG, AAAT, AAGT, AAGT) has a higher score (implying fewer mismatches) than (1, ATAA, AGAA, ATAA, GTAA), we accept (3, AAAG, AAAT, AAGT, AAGT) as the best repeat (Fig. 3c).

The algorithm then continues with $j = 4$ and using the methodology outlined earlier, discards the repeat with that offset (Fig. 3d). At this point, the algorithm has obtained the best repeat within the current frame as (3, AAAG, AAAT, AAGT, AAGT). The algorithm then seeks for the reference offset for the next frame. We examine the repeat with offset $j = 10$. Since this is not within k of the reference, we check to see if this repeat has an overlap with the best repeat for the current frame. Since this is the case, we discard this repeat (Fig. 3e). Hence, the only accepted repeat for the example sequence is (3, AAAG, AAAT, AAGT, AAGT). This repeat, by definition, causes all the other repeats in the frame (since in this case there is only one frame) to be redundant.

IMPLEMENTATION AND RESULTS

We have implemented the algorithm described above and applied it to several test sequences. The user can specify the range of pattern sizes by providing lower and upper bounds.

Sensitivity

Our search algorithm is known to be exhaustive, and our filtering algorithm is also known to be conservative, hence we can precisely describe which repeats have been omitted. For comparison, we studied the results from Benson's TRF. Our results were obtained by running TRF v3.21 (linux binary as obtained from the web). The input parameters are summarized in Table 1.

Table 2 displays some repeats found by our algorithm but not by TRF. Of these, the first and the last repeats are confirmed to be tandem repeats, with respect to the probabilistic definition as used by TRF, since the web interface identifies them as such when each subsequence was pasted into the web version of TRF. The alignment of the first repeat is shown in Figure 4.

Performance

When measuring the performance of the program for searching with various pattern lengths, we found that the total run time (i.e. wall-clock time) was near-proportional to the size of the output, namely the number of tandem repeats found (see plot in Fig. 5). What this means is that writing the output is significantly more expensive than the other computational operations performed during the run. So plotting the total run time of the program at best reveals information about the

Table 1. Parameters used for Benson's Tandem Repeats Finder, v3.21

Parameter	Value
(match, mismatch, indels)	(2, 7, 7)
Match prob.	80
Indel prob.	10
Min. align. score to report	50
Max. period size	500

The reader is referred to Benson (1999) for an explanation of these parameters.

Table 2. Some tandem repeats in *H. sapiens* chr. 1 which are not found by TRF

Period	Copies	Contig	Indices
37		3NT_019273.15	6338192–6338302
84		2NT_021877.15	5031665–5031832
176		2NT_021877.15	413526–413877
111		2NT_021937.15	1425446–1425667
118		2NT_021937.15	526994–527229
1410		4NT_032962.4	2607662–2613301

```
Reading Sequence Data...Done
Hs1_19429 NT_019273.15 Homo sapiens chromosome 1
genomic contig
Sequence length 6517873 base pairs
37:3:6338192
REPEAT 37:3:6338192
              TGGTGGCTGGACAGAGGCGCTCCCCACCTCCCAGATG
  34 matches  #  #      #
              GGGCGGCTGGGCAGAGGCGCTCCCCACCTCCCAGATG
  34 matches     #           #       #
              GGGTGGCTGGGCAGAAGCGCTCCTCACCTCCCAGATG
score == 68 / 74
====================
```

Fig. 4. Repeat in genomic contig NT_004391 of period 48, count 10.

input sequence itself (namely something about the inherent redundancy) rather than any characteristic of the algorithm. Therefore, the following timings were made on a variant of program whose output is disabled (i.e. the timing no longer counts the disk I/O operations).

The time required to search for tandem repeats, of pattern lengths from 2 to 500, is displayed in Figure 6. The graph, and a linear-regression analysis, suggest that for a given pattern size range, run time is proportional to the length of the input sequence.

Figure 6b shows the time required for various pattern size ranges. The graph, and a linear-regression analysis, suggest that the run time is proportional to the size of pattern size range. Figure 7 shows the time spent searching for tandem repeats of specific pattern lengths, omitting the time required to read the input sequence into memory. The graph

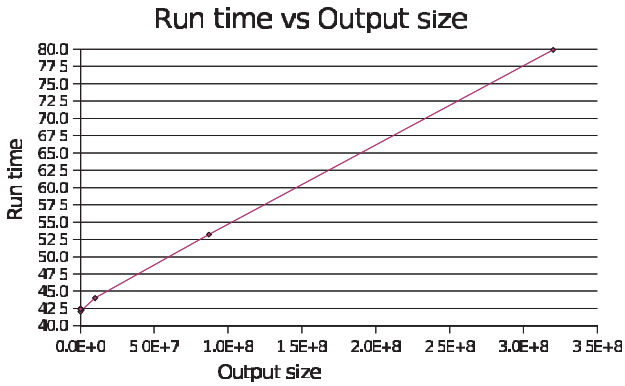


Fig. 5. Plot of run time against output size for specific pattern lengths. Input sequence: *H.sapiens* chromosome 1.

roughly shows that the performance improves as the pattern size increases. In practice, we can find all tandem repeats in *Homo sapiens* chromosome 1 (about 229 million nucleotides, i.e. $N = 229 \times 10^6$) for pattern sizes from 2 to 500 in <90 min (this timing is for the original program with output). Scanning the same sequence for pattern sizes from 2 to 10 000 takes <29 h. These figures were obtained from code compiled with Intel Compiler 7.1, and execution runs on an Itanium 2, 900 MHz processor. The machine has 8 GB of RAM installed, though the program requires ∼1 GB to process this particular sequence. The graph coincides with our theoretical prediction of linear asymptotic performance.

Figure 8 shows the performance of the filtering algorithm for a particular pattern size (1866) for a small section of *H.sapiens* chromosome 1. The figures show the unfiltered and filtered outputs. The filtering algorithm was able to decrease the number of repeats reported from 13 733 belonging to 11 distinct groups (or frames) to 11 repeats, corresponding to one from each group (frame).

Parallelization

Other than initially reading the sequence data, our algorithm is essentially embarrassingly parallel, since we can independently search for tandem repeats for different pattern lengths k. For example, we can parallelize the search across two processing elements by searching with the following pattern length ranges: 2–5001 and 5002–10001. We generalized this and partitioned the pattern length range 2–10001 evenly between 1, 5, 10, 25 and 50 processors. We achieve near-linear speed-up, (see Fig. 9 for a plot of speed-up against the theoretical maximum of linear-speed-up). The parallel computer was a Beowulf-class cluster consisting of 64 compute nodes with dual 1.4 GHz Pentium III processors (i.e. a total of 128 processors) and at least 2 GB of RAM. The input sequence was stored on a shared partition on a NAS (disks with a dedicated NFS server). It is likely that disk bandwidth contention is a significant factor explaining why, for 50 CPUs, we only

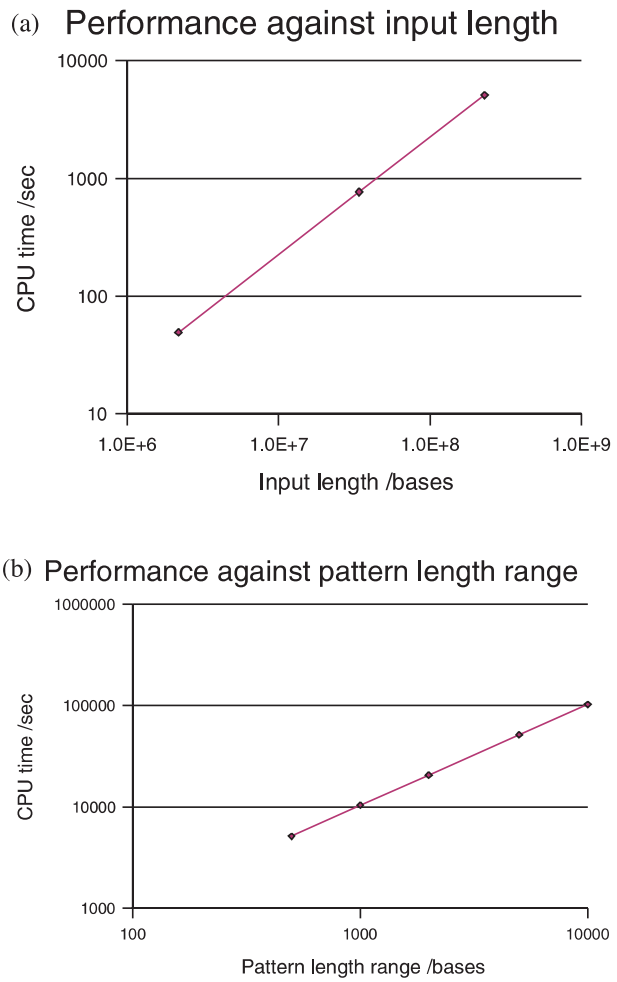


Fig. 6. Performance with respect to input sequence length. System: HP rx2600 (dual Itanium 2 900MHz), 8GB RAM, Intel C Compiler 7.1 for Itanium. **(a)** Run time against input sequence length. The log–log plot has slope 1, and thus run time is linearly dependent on input length. Times displayed are for pattern size 2 to 500 nt, in: *A. fulgidus* DSM 4304; *H.sapiens* chromosome 21; *H.sapiens* chromosome 1. **(b)** Run time against pattern range size. The log-log plot has slope 1, and thus run time is linearly dependent on pattern length range. Input sequence: *H.sapiens* chromosome 1 (229M nt).

attain 97% of the theoretical maximum speed-up, since, for *H.sapiens* chromosome 1, each of the 50 processes initially has to read almost 230 MB from disk.

DISCUSSION AND SUMMARY

We have presented an exhaustive algorithm for finding all maximal, approximate tandem repeats. Additionally, we have also presented a filtering algorithm that reduces the exhaustive set of maximal ATRs to a smaller set with fewer redundancies.

Our definition of ATR uses Hamming distance which cannot take into account insertions and deletions, unlike the definition used by, e.g., Benson's TRF. However, since we have

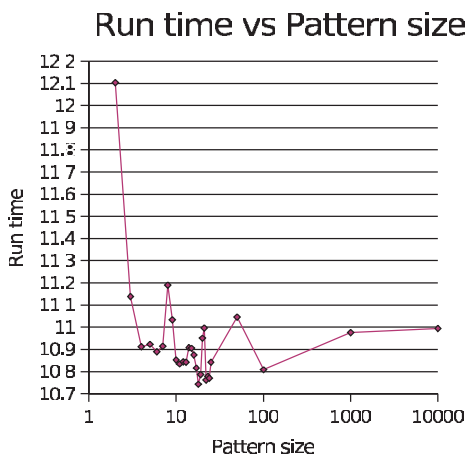


Fig. 7. Run time against pattern size. Timings omit time-spent initial reading input sequence into memory.

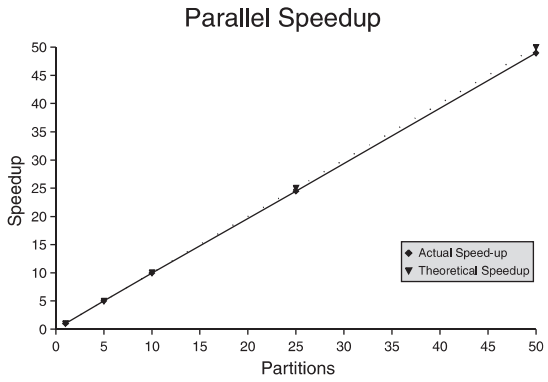
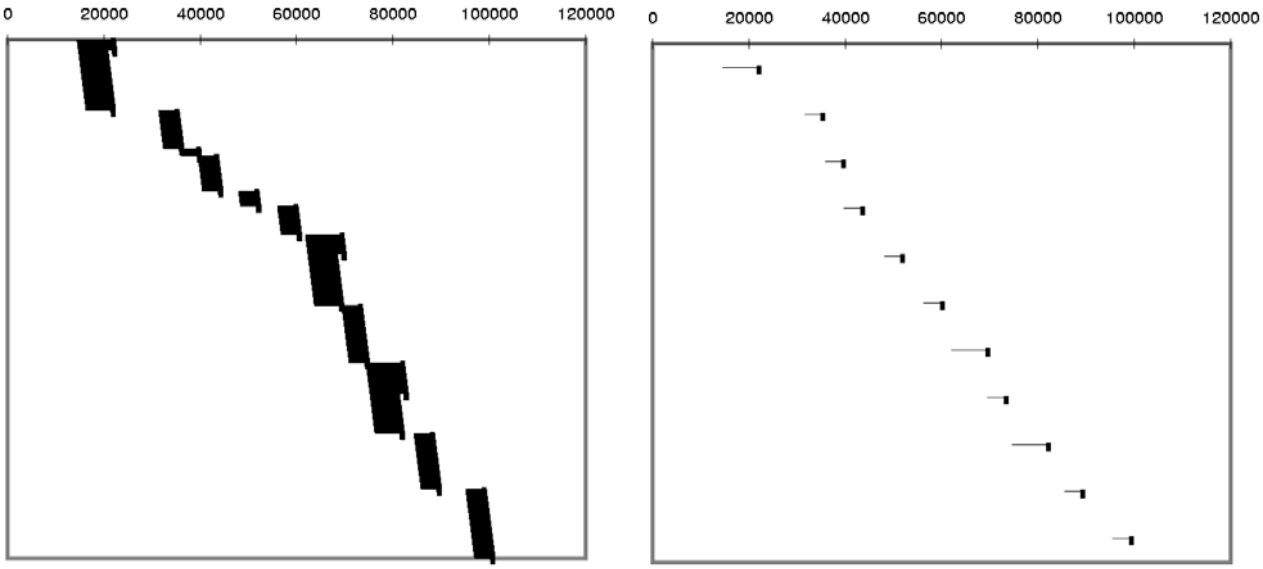


Fig. 9. Performance with parallelization. Times obtained cluster consisting of 32 HP Proliant DL360-G2 (dual Pentium III 1.4 GHz) compute nodes, each with at least 2 GB of RAM.

shown that our algorithm can find ATRs missed by TRF, we believe that our program should augment those currently available to the bioinformatician.

We can show that for a given k, the complexity of our algorithm has worst-case upper bound of $O(kN)$, thus for all possible k, the complexity is $O(N^3)$. However, the constructed input sequence used to obtain this bound is very specific and, we believe, is unlikely to be observed in typical organisms. In fact, empirical evidence suggests, for typical inputs and a given pattern size, run time is almost linearly proportional to input length (Fig. 6a). Furthermore, since the cost of the disk writes is relatively expensive, the real-world performance is ultimately more dependent on the number of tandem repeats present in the input sequence.

Modifying the algorithm to use a measure which can account for indels between copies, e.g. edit distance, will likely adversely affect worst case performance, and significantly increase the real-life run time. This is because dynamic programming will likely be required to determine whether two copies are similar, and this has $O(k^2)$ complexity,



(a) Unfiltered output (b) Filtered output

Fig. 8. Outputs from a section of *H.sapiens* chromosome 1 for a pattern size of 1866. The figures show the unfiltered and filtered outputs for the algorithm. The filtering algorithm decreased the number of repeats from 13 733 (belonging to 11 distinct groups as can be seen from Fig 8a) to 11 tandem repeats (corresponding to one for each group) as seen in Figure 8b.

whereas in the current algorithm, similarity can be determined in at best $O(1)$ and at worst $O(k)$ time.

The filtering algorithm that we have described is more conservative than aggressive.[5] Nevertheless, as shown in Figure 7, the filtering algorithm helps to pare down the exhaustive set to more manageable proportions. However, we do not filter repeats across pattern sizes. This implies, e.g., that a repeat which occurs at a higher pattern size (e.g. 60), could occur again for a pattern size that is a factor of the larger pattern size (e.g. 30).

Moreover, the algorithm that we have described is embarrassingly parallel. As a result, it is very easy to parallelize the algorithm on a compute cluster or even on the grid. Another advantage of our algorithm is that there are no inherent restrictions on the input file sizes or on the pattern sizes being searched.

Not described in this article, our implementation allows the threshold p and the number of repeats l to be parameterized by pattern size k. This allows for the algorithm to account for the fact that a small k, in general, gives repeats of large l, and for large k even low similarity scores are significant.

ACKNOWLEDGEMENTS

The authors would like to thank Li Kuo-Bin, Stephen Wong and Alywin Ng for fruitful discussions on the topic. Thanks again to Li Kuo-Bin for suggesting the problem.

[5] We suspect that there might not necessarily exist a completely non-redundant and safe set.

REFERENCES

Benson,G. (1999) Tandem repeats finder: a program to analyze DNA sequences. *Nucleic Acids Res.*, **27**, 573–580.

Cohen,I., Liu,X., Schutz,C., White,B., Jenkins,E., Brown,W. and Holden,J. (2003) Association of autism severity with a monoamine oxidase A functional polymorphism. *Clin. Genet.*, **64**, 190–197.

Collins,J.R., Stephens,R.M., Gold,B., Long,B., Dean,M. and Burt,S.K. (2003) An exhaustive DNA micro-satellite map of the human genome using high performance computing. *Genomics*, **82**, 10–19.

Fogu,G., Bertini,V., Dessole,S., Bandiera,P., Campus,P.M., Capobianco,G., Sanna,R., Soro,G. and Montella,A. (2003) Identification of a mutant allele of the androgen receptor gene in a family with androgen insensitivity syndrome: detection of carriers and prenatal diagnosis. *Arch. Gyn. Obstet.*, **269**, 25–29.

Groult,R., Léonard,M. and Mouchard,L. (2002) Evolutive tandem repeats using Hamming distance. In *Proceedings of 27th Symposium on Mathematical Foundations of Computer Science*, Warsaw, Poland, August 2002, Springer, pp. 292–304.

Groult,R., Léonard,M. and Mouchard,L. (2004) Speeding up detection of evolutive tandem repeats. *Theor. Comp. Sci.*, **310**, 309–328.

Guerini,F.R., Ferrante,P., Losciale,L., Caputo,D., Lombardi,M.L., Pirozzi,G., Luongo,V., Sudomoina,M.A., Andreewski,T.V., Alekseenkov,A.D. *et al*. (2003) Myelin basic protein gene is associated with ms in DR4- and DR5-positive Italians and Russians. *Neurology*, **61**, 520–526.

Kolpakov,R. and Kucherov,G. (2003) Finding approximate repetitions under Hamming distance. *Theor. Com. Sci.*, **303**, 135–156.

Kurtz,S., Choudhuri,J.V., Ohlebusch,E., Schleiermacher,C., Stoye,J. and Giegerich,R. (2001) REPuter: the manifold applications of

repeat analysis on a genomic scale. *Nucleic Acids Res.*, **29**, 4633–4642.

Landau,G.M., Schmidt,J.P. and Sokol,D. (2001) An algorithm for approximate tandem repeats. *J. Comp. Biol.*, **8**, 1–18.

Licastro,F., Grimaldi,L.M., Bonafe,M., Martina,C., Olivieri,F., Cavallone,L., Giovanietti,S., Masliah,E. and Franceschi,C. (2003) Interleukin-6 gene alleles affect the risk of Alzheimer's disease and levels of the cytokine in blood and brain. *Neurobiol. Aging*, **24**, 921–926.

Qian,Q., Wang,Y., Li,J., Yang,L., Wang,B. and Zhou,R. (2003) Association studies of dopamine D4 receptor gene and dopamine transporter gene polymorphisms in Han Chinese patients with attention deficit hyperactivity disorder. *J. Peking Univ. Hea. Sci.*, **35**, 412–418.

Sagot,M. and Myers,E.W. (1998) Identifying satellites in nucleic acid sequences. In *Second Annual International Conference on Research in Computational Molecular Biology (RECOMB)*. ACM Press, New York, pp. 234–242.

Stolovitzky,G., Gao,Y., Floratos,A. and Rigoutsos,I. (2001) Tandem repeat detection using pattern discovery, with applications to identification of yeast satellites. *Technical Report RC 21508*. IBM T. J. Watson Research Center, Cambridge.