A Fast Look-up Algorithm for Detecting Repetitive DNA Sequences

X. Guan and E. C. Uberbacher

Informatics Group
Computer Science and Mathematics Division
Oak Ridge National Laboratory
Oak Ridge, TN 37831-6364

To be submitted as full article to *The First Pacific Symposium on Biocomputing Ritz Carlton Hotel,* Kona ~~Big Island,~~ Hawaii, January 3-6, 1996.

*Research was supported by the Office of Health and Environmental Research, U.S. Department of Energy under contract No. DE-AC05-840R21400 with Martin Marietta Energy Systems, Inc.

A Fast Look-Up Algorithm for Detecting Repetitive DNA Sequences

X. Guan and E. C. Uberbacher

Computer Science and Mathematics Division
Oak Ridge National Laboratory
Oak Ridge, TN 37831-6364

Abstract

This paper presents a fast linear time algorithm for rapid identification of tandem repeats. It utilizes indices calculated from non-continuous and overlapping k-tuples so that tandem repeats with insertions and deletions can be recognized. It has been made available through the GRAIL (GRAIL@ornl.gov) and GENQUEST (Q@ornl.gov) Internet servers. Its performance is compared with that of another Internet server Pythia.

Introduction

Repetitive sequences are abundant in human DNA sequences. By one estimate, from 30% to 50% human genome consists of repeats of one form or another (Benson and Waterman, 1994). In this paper, we focus on repetitive sequences in which short words are repeated many times, referred to as tandem repeats or microsatellites in the literature. Tandem repeats are highly polymorphic and have been used as polymorphic markers in DNA. A number of genetic diseases are caused by an amplified number of tandem repeats within or adjacent to a gene (Verkerk *et al.*, 1991). Repetitive sequences also present a problem for sequence comparison because they tend to produce non-biologically significant alignments. It is therefore important to have efficient methods to recognize tandem repeats.

Benson and Waterman (1994) developed a method that scans a sequence from left to right looking for a potential repeat pattern (a basic unit that constitutes a repetitive sequence), and then uses a specialized dynamic programming algorithm (wraparound dynamic programming developed by Landau and Schmidt (1993)) to find the boundaries and the alignment of the repetitive sequence. The algorithm requires several input parameters such as the period size (the length of the basic repeat unit or pattern) and the pattern detection parameter (used for initial scanning of a sequence for potential repeat pattern). It is not a linear time algorithm because it uses a dynamic programming algorithm.

Another method proposed by Milosavljevic and Jurka (1993) utilizes a data compression technique to locate tandem repeats. If a pattern is duplicated many times, copies of the pattern can be replaced by pointers to the pattern, thereby reducing the number of bits required to represent the sequence segment covered by the patterns. This may signal a tandem repeat. This algorithm is a linear time algorithm.

In this paper, we describe a fast look-up method that can scan DNA sequences rapidly for tandem repeats. The algorithm uses a new look-up technique and is linear in time.

We use the definition of tandem repeats given by Benson and Waterman (1994). A *pattern* is any particular segment of sequence. A *tandem repeat* is the concatenation of two or more copies of a pattern. These copies may vary from each other in the form of substitutions, insertions, or deletions.

Methods

Look-up table techniques have been used to identify matching words or segments in sequence comparison methods (e.g., FastA, Pearson and Lipman, 1988). Given two sequences,

ACTTGACT
TTGACATA

A look-up table can be calculated from the first sequence that records the position of different nucleotides, i.e., A appears at positions 1 and 6, C appears at positions 2 and 7, etc.

A	1	6	
C	2	7	
G	5		
T	3	4	8

Then the second sequence is scanned from left to right. For each nucleotide, the positions of the nucleotide in the first sequence are fetched from the table and offsets (the differences of the nucleotide's positions in the two sequences) are calculated. For example, the first nucleotide of the second sequence is T which also occurs at positions 3 and 4 in the first sequence. So we get two offsets 3 - 1 = 2 and 4 - 1 = 3. Each nucleotide in the second sequence, if it is also present in the first sequence, produces one or more offsets. For each offset, we maintain a counter which counts how many nucleotides produce this offset. The contents of these offset counters are called offset scores. The offset scores for the second sequence are

offset	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7
score	1	0	1	1	3	1	0	1	2	5	1	0	0	1	1

If these two sequences share a common word w at a certain offset, that offset will get a score $\geq$ the length of w. Clearly the word TTGAC shared by the two sequences produces the highest offset score 5 at offset 2.

Similarly, a k-mer with $k \geq 2$ can also be used to build the table, though it takes more time and space. The technique can be used to compare a sequence with itself to detect tandem repeats. However, the patterns in tandem repeats typically are not duplicated exactly. Substitutions, insertions, or deletions make it difficult to use the lookup technique described above where an insertion or a deletion can interrupt otherwise matching segments.

Recently, a new look-up technique, which uses indices calculated from non-continuous, overlapping tuples, was employed in the FLASH program (Califano and Rigoutsos) for sequence homology search and is described below.

Given a sequence, $a_1a_2...a_n$, where a_i, $1 \leq i \leq n$, can have s different values (for DNA, $s = 4$), let λ be a function that maps a_i to an integer in $[0, s-1]$. A fixed number of indices are calculated for each position in the sequence as follows: given a window size w, and a tuple length of $k \leq w$, for each position i, a k-tuple is formed by selecting k ordered elements from the window starting at i, say $b_1b_2...b_k$, where $b_1 = a_i$. It is important that $b_1 = a_i$ because we are calculating indices for position i. An index can be calculated from this k-tuple as follows:

$$I = \sum_{t=1}^{k} \lambda(b_t) * s^{t-1}$$

Because we require that the first element is always a_i, there are $d = \binom{w-1}{k-1}$ different such k tuples ($\binom{n}{m}$ is a binomial coefficient, i.e., the number of ways to choose m ordered positions from n positions), therefore d indices for each position in a sequence. One way of selecting the k positions from a window is by masking. A vector of length w has k positions set to 1 and other positions set to 0. For $w = 5$ and $k = 3$, there are $d = \binom{5-1}{3-1} = 6$ such vectors. They are

11100

11010

11001

10110

10101

10011

For a sequence segment TGTAT, applying the first vector to the segment results in the 3-tuple TGT. Totally we have six 3-tuples, TGT, TGA, TGT, TTA, TTT, and TAT, that are used for calculating indices.

Indices calculated from non-continuous k-tuples allow insertions and deletions between matching segments, and overlapping k-tuples provide rich indices for each position in a sequence.

The new indexing technique is used here to recognize tandem repeat as follows. We build a lookup table by scanning a sequence from left to right. For each position i in the sequence, we calculate the d indices $I_1, I_2, ..., I_d$ as before. For each I_t, $1 \leq t \leq d$, we first store a pair (t, i) at the lookup table entry I_t, which means an index I_t has been generated from the t-th vector for the current position i, and then we search the same table entry I_t for positions that share the same index I_t calculated from the same vector. Suppose a pair (t, j) is stored in the same table entry I_t, that is, the index I_t has also been generated from the t-th vector for position j. We say that position i has one vote for position j. After all the indices $I_1, I_2, ..., I_d$ have been processed, the position (the window that beginning at that position) that receives the most votes from position i is the one that most resembles the window starting at i.

Note that for each position i, we only need to look for position $j < i$ that most resemble position i, so we can build the lookup table as we scan the sequence. If position i votes for position $j < i$ and position j votes for position $k < j$, then we say that position i votes for k. Therefore, only the most recent position for each index needs to be stored, and for each table entry, we need only to store at most d positions. Now instead of storing a pair (t, i) at a table entry I, we store i at the t-th column at entry I. The lookup table is depicted in Table 1.

	1	2	...	d
1	i...	i...	...	i...
2	i...	i...	...	i...
...	...	...	...	...
k^s	i...	i...	...	i...

Table 1. Lookup table. Given window size w, tuple length k, and the number of values s that a position can have, $d = \binom{w-1}{k-1}$

If there is a tandem repeat in the sequence, the first pattern of the tandem repeat will get the most votes, and the last position that votes for the first pattern marks the end of the tandem repeat.

Space is not as much a problem here as it is in FLASH. Since we are looking for tandem repeat of short pattern, for each position, we only need to look forward (to the left of the position) limited distance (let's called it look-forward-distance), and the table size is at most dk^s. For the parameters that we are using in our present system, $w = 5$, $k = 3$, $s = 4$, and $d = 6$, the table size is 486.

Let f be the look-forward-distance. Our algorithm identify tandem repeats with pattern sizes from 1 to f in one pass. In a comparison, a fixed pattern size is a required input parameter to the method by Benson and Waterman (1994).

Our algorithm is a linear time algorithm. We scan the sequence only once, and for each position, only constant number of indices are calculated and looked up in the table.

Implementation and Results

The algorithm was implemented in C language, and has been incorporated into the GENQUEST and GRAIL Internet email servers. To access GENQUEST, send an email message to Q@ORNL.GOV.

To illustrate the use of the algorithm, we have applied it to the Human Tissue Plasminogen Activator Gene (GenBank Release 89, accession number K03021). The parameters we used were $w = 5$, $k = 3$, and $d = 6$. Two annotated tandem repeats, a $(RY)^n$ (RY repeated n times) run (7170-7225) and a $TGATAGA$ tandem repeat region (23888-24458), were identified. Two other un-annotated tandem repeats, a $(AC)^n$ run followed by a $(TC)^n$ run (16910-16961) and a poly(A) segment (17107-17132), were also identified by our algorithm. These four tandem repeats were reported by the Pythia server (Milosavljevic and Jurka, 1993). Regions identified by our algorithm but not reported by Pythia include several poly(A) segments (1015-1047, 26467-26486, 29083-29104, 19160-19178) and an interesting tandem repeat region (21562-21597)

AAATAATAATAATAAATAAATAATAAATAAATAAAT.

This can be considered as a tandem repeat of pattern AAAT with deletions (underscore means deletion):

AAAT _AAT _AAT _AAT AAAT AAAT _AAT AAAT AAAT AAAT

Pythia also uses a linear time algorithm, but in practice, our algorithm is 5-6 times faster (note, we compare our algorithm with only that portion of the Pythia server which identifies simple repeat regions).

Conclusions

We have presented a fast linear time algorithm for recognizing tandem repeats. Our algorithm is a one pass algorithm. No information about the periodicity of tandem repeats is needed. The use of the indices calculated from non-continuous and overlapping k-tuples allow tandem repeats with insertions and deletions to be recognized.

References

[1] Altshcul,S.F., Gish,W., Miller,W., Myers,E.W., and Lipman, D.J. (1990) "Basic Local Alignment Search Tool," *Journal of Molecular Biology,* vol. 215, 403-410.

[2] Califano,A. and Rigoutsos, I. "FLASH: A Fast Look-up Algorithm for String Homology," *CABIOS.* To appear.

[3] Claverie,J.-M. and States, D.J. (1993) "Information Enhancement Methods for Large Scale Sequence Analysis," *Computers & Chemistry*, vol. 17, 191-201.

[4] Benson, G. and Waterman, M.S. (1994) "A Method for Fast Database Search for all k-nucleotide Repeats," *Proc., Proc., The 2nd International Conference on Intelligent Systems for Molecular Biology*, AAAI Press.

[5] Guan,X. Mural,R. Petrov,S. and Uberbacher,E.C. (1993) "A Sensitive Sequence Comparison Server for DNA and PROTEINS," *Proc., Genome Sequencing and Analysis Conference V*, Hilton Head Island, South Carolina.

[6] Landau,G. and Schmidt,J. (1993) "An Algorithm for Approximate Tandem Repeats," *Fourth Annual Symposium on Combinatorial Pattern Matching*, 120-133.

[7] Milosavljevic,A. and Jurka,J. (1993) "Discovering Simple DNA Sequences by the Algorithmic Significance Method," *CABIOS,* vol. 9, 407-411.

[8] Pearson,W.R. and Lipman,D.J. (1988) "Improved Tools for Biological Sequence comparison," *Proc. National Academy of Sciences, USA*, vol. 85, 2444-2448.

[9] Verkerk, A. J. M. H. *et al*, (1991) "Identification of a Gene (FMR-1) Containing a CGG Repeat Coincident with a Breakpoint Cluster Region Exhibiting Length Variation in Fragile X Syndrome," *Cell,* 65,905-914.

[illegible]