Published online 25 November 2011 *Nucleic Acids Research, 2012, Vol. 40, No. 3* ***e17***
doi:10.1093/nar/gkr1070

NTRFinder: a software tool to find nested tandem repeats

Atheer A. Matroud[1,2,*], M. D. Hendy[3] and C. P. Tuffley[1]

[1]Institute of Fundamental Sciences, [2]Allan Wilson Centre for Molecular Ecology and Evolution, Massey University, Private Bag 11 222, Palmerston North 4442 and [3]Department of Mathematics and Statistics, University of Otago, PO Box 56, Dunedin 9054, New Zealand

Received April 12, 2011; Revised October 3, 2011; Accepted October 28, 2011

ABSTRACT

We introduce the software tool `NTRFinder` to search for a complex repetitive structure in DNA we call a nested tandem repeat (NTR). An NTR is a recurrence of two or more distinct tandem motifs interspersed with each other. We propose that NTRs can be used as phylogenetic and population markers. We have tested our algorithm on both real and simulated data, and present some real NTRs of interest. `NTRFinder` can be downloaded from http://www.maths.otago.ac.nz/∼aamatroud/.

INTRODUCTION

Genomic DNA has long been known to contain ‘tandem repeats’: repetitive structures in which many approximate copies of a common segment (the ‘motif’) appear consecutively. Several studies have proposed different mechanisms for the occurrence of tandem repeats (1,2), but their biological role is not well understood.

Recently, we have observed a more complex repetitive structure in the ribosomal DNA of *Colocasia esculenta* (taro), consisting of multiple approximate copies of two distinct motifs interspersed with one another. We call such structures nested tandem repeats (NTRs), and the problem of finding them in sequence data is the focus of this article. Our motivation is their potential use for studying populations: for example, a preliminary analysis suggests that changes in the NTR in taro have been occurring on a 1000 year time scale, so a greater understanding of this NTR offers the potential to date the early agriculture of this ancient staple food crop.

The problem of locating tandem repeats is well known, as their implication for neurological disorders (3,4), and their use to infer evolutionary histories has urged some researchers to develop tools to find them. This has resulted in a number of software tools, each of which has its own strengths and limitations. However, the existing tools were not designed to find NTRs, and consequently do not generally find them. In this article, we present a new software tool, `NTRFinder`, which is designed to find these more complex repetitive structures.

We report here the algorithm on which `NTRFinder` is based and report some of the NTRs it has identified, including an even more complex structure where copies of four distinct motifs are interspersed.

Sequences, edit operations and the edit distance

A DNA sequence is a sequence of symbols from the nucleotide alphabet $\Sigma = \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$. We define a DNA segment to be a string of contiguous DNA nucleotides and define a site to be a component in a segment. For a DNA segment

$$\mathbf{X} = x_1 x_2 \cdots x_n,$$

$x_i \in \Sigma$ is the nucleotide at the i-th site and $|\mathbf{X}| = n$ is the length of $\mathbf{X}$.

Copying errors happen in DNA replication due to different external and internal factors. These changes include substitution, insertion, deletion, duplication and contraction. We refer to these as ‘edit operations’. By giving each type of edit operation some specific weight, we can in principle find a series of edit operations which transforms segment x to segment y, whose sum of weights is minimal. We will refer to this sum as the ‘edit distance’, and denote it by $d(x, y)$. For the purposes of this article, the edit operations allowed in calculating the edit distance are restricted to single nucleotide substitutions, and single nucleotide insertions or deletions (indels).

Classification of tandem repeats

Many classifications of tandem repeat schemas have been introduced in the computational biology literature. We list some which are commonly used:

- (Exact) tandem repeats: an ‘exact tandem repeat’ (TR) is a sequence comprising two or more contiguous copies $\mathbf{XX}\ldots\mathbf{X}$ of identical segments $\mathbf{X}$ (referred to as the *motif*).
- k–Approximate tandem repeats: a *k–approximate tandem repeat* (k–TR) is a sequence comprising two

*To whom correspondence should be addressed. Tel: +6434797989; Fax: +6434798427; Email: a.a.matroud@massey.ac.nz or more contiguous copies $\mathbf{X}_1\mathbf{X}_2\ldots\mathbf{X}_n$ of similar segments, where each individual segment $\mathbf{X}_i$ is edit distance at most k from a template segment $\mathbf{X}$.

- Multiple length tandem repeats: a multiple length TR is a TR where each repeat copy is of the form $\mathbf{Xx}^n$, where n is a constant larger than one and $d(\mathbf{X},\mathbf{x})$ is greater than some threshold value k.

Examples

- **TR:**
  AGG AGG AGG AGG AGG. The motif is AGG.
- **1 − TR**:
  AGG AG$\underline{\text{C}}$ A$\underline{\text{T}}$G AGG $\underline{\text{C}}$GG. The motif is AGG.
- **MLTR**:
  GACCTTTGG ACGGT ACGGT ACGGT
  GACCTTTGG ACGGT ACGGT ACGGT.
  The motifs are x = ACGGT and X = GACCTTTGG, with $n = 3$.

NTRs

In this section, we introduce a more complex repetitive structure, the NTR, also referred to as a 'variable length tandem repeat' (5). Let $\mathbf{X}$ and $\mathbf{x}$ be two segments (typically of different lengths) from the alphabet $\Sigma = \{\text{A}, \text{C}, \text{G}, \text{T}\}$, such that $d(\mathbf{X},\mathbf{x})$ is greater than some threshold value k.

Definition 1.

An 'exact nested tandem repeat' is a string of the form

$$\mathbf{x}^{s_0}\mathbf{Xx}^{s_1}\mathbf{X}\cdots\mathbf{Xx}^{s_n},$$

where $n > 1$, $s_i \geq 1$ for each $0 < i < n$, and $s_j \geq 2$ for some $j \in \{0, 1, \cdots, n\}$. The motif $\mathbf{x}$ is called the TR and the motif $\mathbf{X}$ is the 'interspersed repeat'. The concatenations of the tandem repeats $\mathbf{x}^{s_i}$ alone, and of the interspersed motifs $\mathbf{X}$ alone, both form exact TRs.

Example

x = ACGGT, X = GACCTTTGG, $n = 7$, $s_0 = 0$, $s_1 = 3$, $s_2 = 5$, $s_3 = 2$, $s_4 = 4$, $s_5 = 1$, $s_6 = s_7 = 2$, so

$$\mathbf{x}^0 \prod_{i=1}^{7} \mathbf{Xx}^{s_i}$$

$$= \mathbf{XxxxXxxxxxXxxXxxxxXxXxxXxx}$$

```
=GACCTTTGG ACGGT ACGGT ACGGT
 GACCTTTGG ACGGT ACGGT ACGGT ACGGT ACGGT
 GACCTTTGG ACGGT ACGGT
 GACCTTTGG ACGGT ACGGT ACGGT ACGGT
 GACCTTTGG ACGGT
 GACCTTTGG ACGGT ACGGT
 GACCTTTGG ACGGT ACGGT.
```

In practice, we expect any NTRs occurring in DNA sequences to be approximate rather than exact. In what follows we will write $\tilde{\mathbf{X}}$ to mean an approximate copy of the motif $\mathbf{X}$, and $\tilde{\mathbf{x}}^s$ to mean an approximate TR consisting of s (not necessarily identical) approximate copies of the motif $\mathbf{x}$.

Definition 2.

A (k_1, k_2)-*approximate NTR* is a string of the form

$$\tilde{\mathbf{x}}^{s_0}\tilde{\mathbf{X}}\tilde{\mathbf{x}}^{s_1}\tilde{\mathbf{X}}\cdots\tilde{\mathbf{X}}\tilde{\mathbf{x}}^{s_n},$$

where n and s_i satisfy the same conditions as in Definition 1, and $\tilde{\mathbf{x}}^{s_0}\tilde{\mathbf{x}}^{s_1}\cdots\tilde{\mathbf{x}}^{s_n}$ is a k_1-approximate TR with motif $\mathbf{x}$, and $\tilde{\mathbf{X}}\tilde{\mathbf{X}}\cdots\tilde{\mathbf{X}}$ is a k_2-approximate TR with motif $\mathbf{X}$.

Examples

- **NTR**:
  AGG AGG CTCAG AGG CTCAG AGG AGG AGG CTCAG.
  The motifs are AGG, CTCAG.
- (1, 2)–**NTR**:
  AG$\underline{\text{A}}$ AGG CT$\underline{\text{TC}}$G AGG CTCAG A$\underline{\text{G}}$ AG$\underline{\text{A}}$ AGG CT$\underline{\text{TC}}$G AGG CTCAG $\underline{\text{AA}}$G.
  The motifs are **x** = AGG, X = CTCAG.

Related work

Various algorithms have been introduced to find exact TRs. Such algorithms were developed mainly for theoretical purposes, namely, to solve the problem of finding squares in strings (6–10). These algorithms are not easily adapted to finding the approximate TRs that usually occur in DNA.

A number of algorithms (11,12) consider motifs differing only by substitutions, using the Hamming distance as a measure of similarity. Others, e.g. (13–17), have considered insertions and deletions by using the edit distance. Most of these algorithms have two phases, a scanning phase that locates candidate TRs, and an analysis phase that checks the candidate TRs found during the scanning phase.

The only algorithm designed to look for NTRs is that of Hauth and Joseph in (5), which searches for tandem motifs of length at most 6 nt.

MATERIAL AND METHODS

In this section, we present the algorithm we have developed to search for NTRs in a DNA sequence. The algorithm requires several preset parameters. These are: k_1 and k_2 which bound the edit distances from the tandem and interspersed motifs; and the motif length bounds $\min_{t_1}, \max_{t_1}, \min_{t_2}, \max_{t_2}$. Other input parameters are discussed below.

Search phase

Our search is confined to seeking NTRs with motifs of length $l_1 \in [\min_{t_1}, \max_{t_1}]$ and $l_2 \in [\min_{t_2}, \max_{t_2}]$. A (k_1, k_2)–NTR must contain a $k_1 - $TR, so we begin by scanning the sequence for approximate TRs. To do this, we have chosen to adapt the TR search algorithm of Wexler *et al.*, in which the sequence is searched for tandem motifs of length l_1 by scanning the sequence with two windows w_1 and w_2 of width w, at distance l_1 apart. This may be adapted to find non-adjacent copies of the tandem motif (as occur in NTRs) by holding w_1 fixed, and moving w_2 further away.

The user may set the k_1, k_2 values, preset with default values

$$k_1 = l_1(1 - p_m) + \sqrt{l_1(1 - p_m)p_m}$$
$$k_2 = l_2(1 - p_m) + \sqrt{l_2(1 - p_m)p_m},$$

following Domaniç and Preparata (2007), with matching probability p_m given the default value $p_m = 0.8$.

Once a TR has been found and its full extent determined, the right-most copy of the repeated pattern is taken as the current TR motif **x**, and further approximate copies of **x** are sought, displaced from the TR up to a distance of $\max_{l_2}$ nucleotides to the right. This is done by moving the second scanning window w_2 to the right, while holding the first fixed in the current copy of **x**. If no further approximate copies of **x** are located, this TR is abandoned, and the TR search continues to the right. If a displaced approximate copy of **x** is observed, then both **x** and the interspersed segment **X** are recorded in a list, as we have found a candidate NTR. Further contiguous copies of **x** are then sought, with the rightmost copy **x** replacing the previous template motif.

The steps above are repeated with successive motifs **x** and interspersed segments copied to the list, until no additional copies of the last recorded motif **x** are found. This search phase is illustrated in Figure 1.

At this point, the algorithm builds consensus patterns for **x** and **X** using majority rule. After constructing the two consensus patterns, the algorithm moves to the verification phase.

Example

An example will help illustrate the procedure. Suppose that S contains an NTR of the form

$$\mathbf{xX_0xxxX_1xxxxxxX_2xxX_3}.$$

The algorithm will scan from the left until it locates the TR consisting of three copies of **x** between $\mathbf{X}_0$ and $\mathbf{X}_1$. It will then start searching for additional non-adjacent copies of **x** to the right, locating the first copy to the right of $\mathbf{X}_1$. Having found this, it will record the intervening segment $\mathbf{X}_1$, and then continue the TR search from this point until the full extent of the TR between $\mathbf{X}_1$ and $\mathbf{X}_2$ is found.

This procedure is repeated once more, locating the TR between $\mathbf{X}_2$ and $\mathbf{X}_3$, recording the segment $\mathbf{X}_2$, and then searching for further copies to the right. At this point, no more copies of **x** are found, and the process of verification begins. The segments $\mathbf{X}_0$, $\mathbf{X}_3$ and the initial copy of **x** are found during this stage.

Verification phase

Each candidate NTR is checked to determine whether it meets the NTR definition. This is accomplished by aligning the candidate NTR region, together with a margin on either side of it, against the consensus motifs **x** and **X**, using the nested wrap-around dynamic programming algorithm of Matroud *et al.* (18). The nested wrap-around dynamic programming parameters are set to be 2 for a match, −5 for a mismatch and −7 for a gap. These parameters were chosen following Wexler *et al.* (17). The nested wrap-around dynamic programming algorithm has complexity $O(n|\mathbf{x}||\mathbf{X}|)$, where n is the length of the NTR region and $|\mathbf{x}|$ and $|\mathbf{X}|$ are the length of the tandem motif and the length of the interspersed motif, respectively.

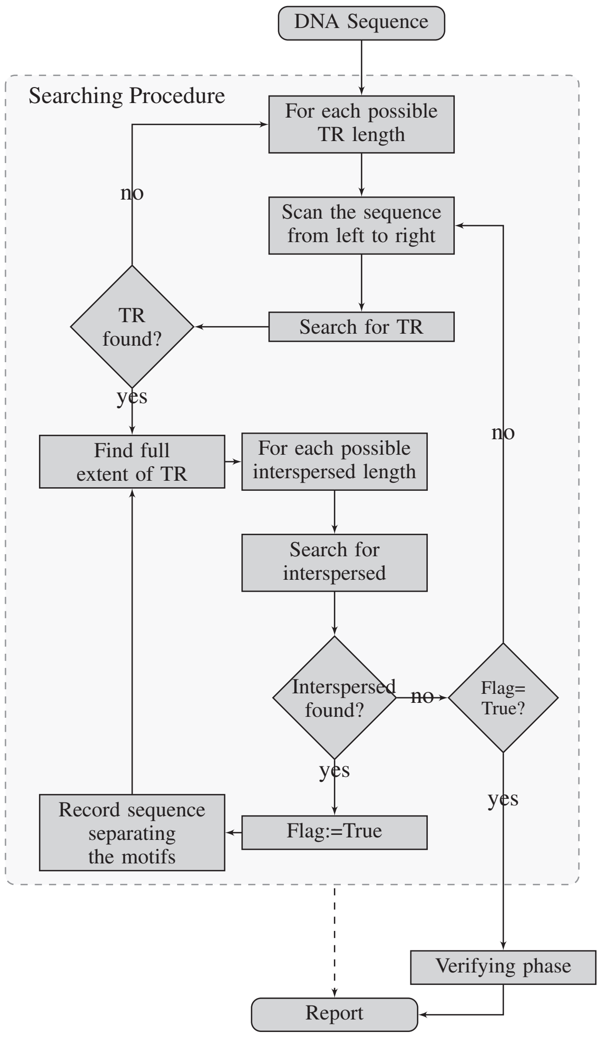

Figure 1. Flowchart of the `NTRFinder` algorithm.

A remark on tandem repeat detection, and the role of verification

The definition of a k-TR requires that each repeat be a distance at most k from some template motif. However, this template is unknown during the search phase. We follow Wexler *et al.*'s algorithm of comparing each repeat copy with the preceding copy. Comparisons between adjacent copies will not miss any TRs, provided the distance threshold is set appropriately, but may result in false positives due to 'drift'. Such false positives are eliminated during the verification phase, when the candidate TR is aligned against the consensus motif.

Suppose that $\mathbf{x}_1\mathbf{x}_2\ldots\mathbf{x}_n$ is a k-TR with motif $\mathbf{x}$. Then since $d(\mathbf{x}, \mathbf{x}_i) \leq k$ we have

$$d(\mathbf{x}_i,\ \mathbf{x}_j) \leq d(\mathbf{x}_i,\ \mathbf{x}) + d(\mathbf{x},\ \mathbf{x}_j) \leq 2k,$$

by the triangle inequality. It follows that a TR search that correctly detects when $d(\mathbf{x}_i, \mathbf{x}_{i+1}) \leq d$ will find all $(d/2)$-TRs.

We note, however, that a segment $\mathbf{x}_1\mathbf{x}_2\ldots\mathbf{x}_n$ satisfying $d(\mathbf{x}_i, \mathbf{x}_{i+1}) \leq d$ for all i need not be a TR, since $\mathbf{x}_j$ may 'drift' away from $\mathbf{x}_i$ as j increases. A simple example is

```
aaaa aaac aacc accc cccc,
```

in which adjacent copies are distance 1 apart, but the first and last copies are distance 4 apart.

RESULTS

Tests on simulated data

In order to measure the accuracy of `NTRFinder`, we generated synthetic sequence data containing NTR subsequences with varying probabilities of substitution and insertion/deletion (indels), and determined the proportion of the NTRs that were found by `NTRFinder`. In our simulation, we first generated one random DNA sequence of 100 000 nt, with each nucleotide occurring with probability 0.25. Within this sequence, we embedded 100 exact NTRs with repeats of randomly generated motifs **X** and **x** of varying lengths. From this sequence, we generated four additional sequences by introducing indels and substitutions. Indels were introduced to each sequence with a constant probability of 1% per site, and substitutions were introduced with varying probabilities of 1, 2, 3 and 4% per site. `NTRFinder` recovered 95, 84, 83, 83 and 80% of the NTRs, respectively. These results are plotted in Figure 2. No false positives were detected.

The first phase of `NTRFinder` uses a modification of ATRhunter's algorithm. In Wexler *et al.* (17), the authors report that ATRhunter has a 74–90% success rate for finding ATRs in synthetic sequences, with average score of an ATR over all sequences being 238 with a standard deviation of 116. These results suggest the accuracy of the Wexler algorithm provides the major limitation on the accuracy of `NTRFinder`.

Tests on real sequence data

To test NTRFinder on real sequence data, we searched all IGS sequences available in GenBank. The IGS sequences were chosen because we already knew of an NTR in the IGS region of *C. esculenta*. We also searched the entire Human Y chromosome from Fujita *et al.* (19).

The size ranges used for this search were $[\min_{t_1}, \max_{t_1}] = [\min_{t_2}, \max_{t_2}] = [2, 100]$, with the parameters k_1 and k_2 set to their default values. NTRs found in IGS sequences are listed in Table 1. We searched 27 IGS sequences and found NTRs in 12 of them.

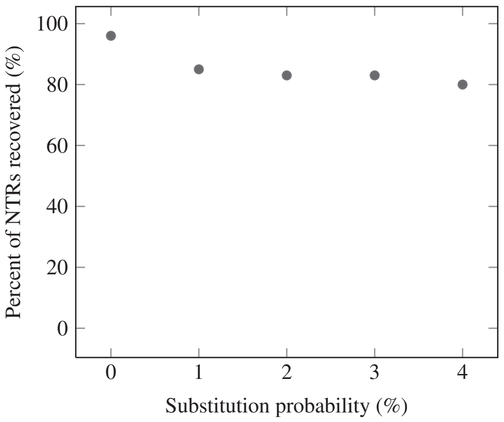

Figure 2. Percentage of NTRs found in the synthetic sequences.

NTRs found in the Human Y chromosome are listed in Table 2. All NTRs found in the Y chromosome appear to be in the pseudoautosomal region.

More complex structures

In addition to the NTRs in Table 2, `NTRFinder` also reported an NTR in *Linum usitatissimum* (accession number gi|164684852| gb|EU307117.1|) which on further analysis by hand turned out to have a more complex structure. The IGS region of the rDNA of this species contains an NTR with four motifs interspersed with each other. The four motifs are `w=GTGCGAAAAT`, `x=GCGCGCCAG GG`, `y=GCACCCATAT` and `z=GCGATTTTG`, and the structure of the NTR has the form

$$\prod_{i=1}^{25} \mathbf{w}^{q_i}\mathbf{x}^{r_i}\mathbf{z}^{s_i}\mathbf{y}^{t_i},$$

where $q_i \in \{1, 2, 3\}$; $r_i \in \{1, 2\}$; $s_i \in \{0, 1\}$; $t_i \in \{0, 1\}$.

Running time

The running time for `NTRFinder` searching some sequences from GenBank is shown in Figure 3. It can be seen that the run time is approximately linear in the length of the sequence. However, it must be noted that the run time depends not only on the length of the input sequence, but also on the number of TRs and NTRs found in the sequence. The program spends most of the time verifying any TRs found.

DISCUSSION

In the last decade, a number of software tools to find TRs have been introduced; however, little work exists on more complex repetitive structures such as NTRs. The problem of finding NTRs is addressed in this study. The motivation for our study is the potential use of NTRs as a marker for genetic studies of populations and of species.

We have done some analysis on the NTR in the intergenic spacer region in *C. esculenta* (taro), noting some variation in the NTRs derived from domesticated varieties sourced from New Zealand, Australia and Japan. Further varieties are currently being analyzed.

Table 1. NTRs found in some IGS sequences searched from GenBank and an additional unpublished sequence (*C. esculenta*)

Species and accession number	NTR structure $(s_0, s_1, \ldots, s_n)$	|x| |**X**|	Start index End index
Nicotiana sylvestris X76056.1	(0,1,2,3,4,2,3,2,3,3,3,5,4,3,2,3,3,1,2,5,3,5,4,2,2,4,3)	10	960
		13	2111
Brassica juncea X73032.1	(9,12,2,6,2,5,4,1)	21	1403
		30	2605
Brassica olerecea	(1,1,1,3,3,1,3,2,1,1,2,1,3,1,1,2,1,2,1,1,2)	30	1031
		44	2902
Brassica olerecea X60324.1	(1,1,1,3,3,1,3,1,3,2,1,1,2,1,3,1,1,2,1,2,1,1,2)	30	1036
		44	3133
Brassica rapa S78172.1	(1,2,2,3,3,2,2,2,3)	12	385
		45	1337
Brassica campestris X73031.1	(6,4,4,7,4,4,4,3,1)	21	1558
		51	2580
Colocasia esculenta Not published	(5,3,1,6,10,5,10,9,13,14,15,4)	11	725
		48	2384
Nicotiana tomentosiformis Y08427.1	(1,1,2,2,1,2,4,2,1)	20	1016
		46	1969
Arabidopsis thaliana CP002685.1	(3,2,1,1)	13	32 189
		17	32 365
Zea mays AJ309824.2	(2,2,1)	19	2984
		52	3113
Olea europea AJ865373.1	(3,1,3,6,5,6,4,3,4)	75	961
		11	3743
Herdmania momus X53538.1	(3,1,1,1,1,0)	107	6363
		91	7642

Table 2. NTRs found in the Human Y chromosome

NTR structure $(s_0, s_1, \ldots, s_n)$	|x| |**X**|	Start index End index
(1,2,2,1,2,1,2,1,1,2,2,2,2,1)	12	143 865
	56	144 880
(7,22,23,12,14,4)	2	234 183
	88	234 767
(1,1,2,1,1,1,1,,1,1,1,1,1,1,1,1,1,1,1)	15	465 369
	14	466 397
(1,1,1,2,1,1,1,1,1,1,1,1,1,2,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1)	11	647 659
	16	649 721
(17,15,31,28,72,62)	2	901 237
	49	902 037
(3,6,8,11,7,6,4,4,5,4,11)	12	1 279 754
	32	1 280 875
(26,27,25,25,25,20,17,13,26)	1	1 397 128
	48	1 397 735
(1,2,1,2,1,2,2,2,2,2,1,2,1,1,1,2,2,2,1,2,2,1,1)	16	1 516 157
	22	1 517 560
(1,1,2,6,2,2,2,1,2)	19	1 626 578
	35	1 627 258
(1,1,1,0,2,1)	19	2 102 194
	56	2 102 594
(2,2,2,1,2,1,1,1,1,2,6)	21	2 164 541
	15	2 165 091

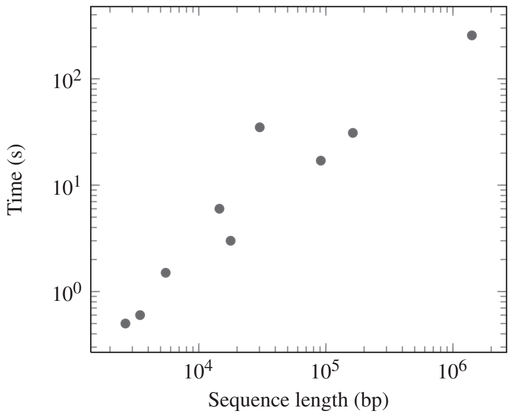

Figure 3. Running time of NTRFinder (on a Pentium Dual core T4300 2.1 GHz) plotted against segment length on a log–log scale. The search was performed on segments of different lengths, with the minimum and maximum TR lengths set to 8 and 50, respectively. The distribution suggests the running time is approximately linear with the sequence length.

CONCLUSION

The NTR structure is a complex structure that requires further analysis and study. The number of copy variants in the NTR region and the relationships between these copies might suggest a TR generation mechanism. In this article, we have introduced a new algorithm to find NTRs. The first phase of the algorithm has $O(n(\max_{t_1})(\max_{t_2}))$ time complexity, while the second phase (the alignment) needs $O(n(\max_{t_1})(\max_{t_2}))$ space and time, where n is the length of the NTR region, and $\max_{t_1}, \max_{t_2}$ are the maximum allowed lengths of the tandem and interspersed motifs.

ACKNOWLEDGEMENTS

We thank Andrew Clarke and Peter Matthews, for providing data and useful background about taro, and Hussain Matawa, for assisting in the development of the program interface. We would also like to thank the anonymous reviewers for their help in improving the manuscript.

FUNDING

The Allan Wilson Centre for Molecular Ecology and Evolution. Funding for open access charge: Allan Wilson Centre.

Conflict of interest statement. None declared.

REFERENCES

1. Weitzmann,M.N., Woodford,K.J. and Usdin,K. (1997) DNA Secondary Structures and the Evolution of Hypervariable Tandem Arrays. *J. Biol. Chem.*, **272**, 9517–9523.
2. Wells,R.D. (1996) Molecular basis of genetic instability of triplet repeats. *J. Biol. Chem.*, **271**, 2875–2878.
3. Macdonald,M.E., Ambrose,C.M., Duyao,M.P., Myers,R.H., Lin,C., Srinidhi,L., Barnes,G., Taylor,S.A., James,M. and Groot,N. (1993) A novel gene containing a trinucleotide repeat that is expanded and unstable on huntington's disease chromosomes. the huntington's disease collaborative research group. *Cell*, **72**, 971–983.
4. Fu,Y.H., Pizzuti,A. Jr, Fenwick,R.G., King,J., Rajnarayan,S., Dunne,P.W., Dubel,J., Nasser,G.A., Ashizawa,T., de Jong,P. *et al.* (1992) An unstable triplet repeat in a gene related to myotonic muscular dystrophy. *Science*, **255**, 1256–1258.
5. Hauth,A.M. and Joseph,D.A. (2002) Beyond tandem repeats: complex pattern structures and distant regions of similarity. *Bioinformatics*, **18(Suppl. 1)**, S31–S37.
6. Apostolico,A. and Preparata,F.P. (1983) Optimal off-line detection of repetitions in a string. *Theor. Comput. Sci.*, **22**, 297–315.
7. Crochemore,M. (1981) An optimal algorithm for computing the repetitions in a word. *Inf. Process. Lett.*, **12**, 244–250.
8. Kolpakov,R., Kucherov,G. and Logiciel,T.G. (2001) Finding approximate repetitions under hamming distance. In *Theoretical Computer Science*. Springer, pp. 170–181.
9. Main,M.G. and Lorentz,R.J. (1984) An *o*(*n log n*) algorithm for finding all repetitions in a string. *J. Algorithms*, **5**, 422–432.
10. Stoye,J. and Gusfield,D. (2002) Simple and flexible detection of contiguous repeats using a suffix tree. *Theor. Comput. Sci.*, **270**, 843–856.
11. Delgrange,O. and Rivals,E. (2004) Star: an algorithm to search for tandem approximate repeats. *Bioinformatics*, **20**, 2812–2820.
12. Landau,G.M., Schmidt,J.P. and Sokol,D. (2001) An algorithm for approximate tandem repeats. *J. Comput. Biol.*, **8**, 1–18.
13. Benson,G. (1999) Tandem repeats finder: a program to analyze DNA sequences. *Nucleic. Acids Res.*, **27**, 573–580.
14. Hauth,A.M. and Joseph,D. (2002) Beyond tandem repeats: complex pattern structures and distant regions of similarity. In *ISMB*, 31–37.
15. Domaniç,N.O. and Preparata,F.P. (2007) A novel approach to the detection of genomic approximate tandem repeats in the levenshtein metric. *J. Comput. Biol.*, **14**, 873–891.
16. Sagot,M.F. and Myers,E.W. (1998) Identifying satellites and periodic repetitions in biological sequences. *J. Comput. Biol.*, **5**, 539–554.
17. Wexler,Y., Yakhini,Z., Kashi,Y. and Geiger,D. (2005) Finding approximate tandem repeats in genomic sequences. *J. Comput. Biol.*, **12**, 928–942.
18. Matroud,A.A., Hendy,M.D. and Tuffley,C.P. (2011) An algorithm to solve the motif alignment problem for approximate nested tandem repeats in biological sequences. *J. Comput. Biol.*, **18**, 1211–1218.
19. Fujita,P.A., Rhead,B., Zweig,A.S., Hinrichs,A.S., Karolchik,D., Cline,M.S., Goldman,M., Barber,G.P., Clawson,H., Coelho,A. *et al.* (2010) The UCSC genome browser database: update 2011. *Nucleic Acids Res.*, **39**, D876–D882.