BIOINFORMATICS APPLICATIONS NOTE Vol. 22 no. 6 2006, pages 762–764
doi:10.1093/bioinformatics/btk041

Sequence analysis

GUUGle: a utility for fast exact matching under RNA complementary rules including G–U base pairing

Wolfgang Gerlach and Robert Giegerich*

Faculty of Technology, Bielefeld University, 33615 Bielefeld, Germany

Received on December 21, 2005; revised and accepted on January 4, 2006
Advance Access publication January 10, 2006
Associate Editor: Martin Bishop

ABSTRACT

Motivation: RNA secondary structure analysis often requires searching for potential helices in large sequence data.

Results: We present a utility program GUUGle that efficiently locates potential helical regions under RNA base pairing rules, which include Watson–Crick as well as G–U pairs. It accepts a positive and a negative set of sequences, and determines all exact matches under RNA rules between positive and negative sequences that exceed a specified length. The GUUGle algorithm can also be adapted to use a precomputed suffix array of the positive sequence set. We show how this program can be effectively used as a filter preceding a more computationally expensive task such as miRNA target prediction.

Availability: GUUGle is available via the Bielefeld Bioinformatics Server at http://bibiserv.techfak.uni-bielefeld.de/guugle

Contact: robert@TechFak.Uni-Bielefeld.DE

1 MOTIVATION

Exact sequence matching under DNA complementarity rules is easy, as each sequence has a unique reverse Watson–Crick complement. When it comes to efficient matching on a genomic scale, indexing methods such as REPuter (Kurtz *et al*., 2001) and Vmatch (Abouelhoda *et al*., 2002) simply compute the suffix tree or suffix array for the genome plus its reverse complement. Reverse complementary matches can then be read from the index, independent of the genome size. For example, the approach in Horesh *et al*. (2003) computes potential helices from a suffix tree, but ignores G–U pairs. Matching under RNA base pairing rules is different because of the presence of G–U and U–G pairs. A sequence like AGUCAAGG not only matches its (reverse) Watson–Crick complement CCUUGACU, but also seven others that include one or more G–U pairs. It is still straightforward to write a naive program that finds such matches. When it comes to large data sizes, an efficient algorithm appears to be missing. The STAN program (Nicolas *et al*., 2005) also uses suffix trees to efficienlty match a pattern (of quite general type) against a target sequence and allows pairing errors that can be G–U pairs. All-against-all matching of two sequence sets is not supported.

*To whom correspondence should be addressed.

Here we present an algorithm and a utility program for this task, both called GUUGle. The algorithm accepts a set of target sequences and a set of query sequences, and a length threshold k. It reports all matches (under RNA rules) beween the target and the reverse of the query sequences that have k or more consecutive base pairs. The algorithmic idea of GUUGle can be applied to a precomputed index (suffix tree or array). To be useful as a standalone utility, our implementation of the GUUGle program starts from non-indexed sequences and mimicks a partial suffix tree construction in parallel with the matching phase.

The application scenario of this approach is the quick determination of potential regions of inter- or intramolecular hybridization to speed up the prediction of secondary structure or complex formation. To demonstrate the efficacy of GUUGle as a filter, we report on its combination with the popular miRNA target prediction program RNAhybrid (Rehmsmeier *et al*., 2004).

2 ALGORITHM

The idea of GUUGle can be explained as running, in an interleaved fashion, two copies of the write-only, top-down (WOTD) suffix tree construction (Giegerich *et al*., 2003). One copy is the regular WOTD algorithm applied to the target sequences. This part can be replaced when using a precomputed index for the target. The other copy is a modified WOTD applied to the reversed queries. We therefore start our explanation with a review of the WOTD construction.

2.1 Review of WOTD

WOTD constructs the suffix tree in a top–down fashion. As it does so by recursive internal sorting of an array of suffixes, it also constructs the suffix array.

Let t be a string over the alphabet Σ and $\bar{u}$ be a node in the suffix tree of t. Then u denotes the concatenation of all edge labels on the path from the root to $\bar{u}$. Each node $\bar{u}$ in a suffix tree represents the set of all suffixes that have the prefix u in common, i.e. $R(\bar{u})$. To evaluate the children of $\bar{u}$, one has to divide the set $R(\bar{u})$ into four groups according to the first character of each suffix. For each character c let group $(\bar{u}, c) := \{w \in \Sigma^* \mid cw \in R(\bar{u})\}$ be the c-group of $R(\bar{u})$. Groups containing only one suffix correspond to a leaf. The labeling of an edge going from u to child $\overline{up}$ begins

with a c and is evaluated by computing the longest common prefix (lcp) p of the suffixes contained in the c-groups.

The datastructure used to represent the suffixes is a global array called suffixes. It contains pointers to all suffixes of t sorted in increasing suffix length. By performing countingsort and looking only at the first character of each suffix, the pointers are sorted in such way, that all suffixes starting with the same character are grouped together in one interval. By computing the lcp for each interval of suffixes that start with the same character $c \in \Sigma$, one retrieves all c-groups.

To evaluate all nodes, the procedure is recursive: For each interval from the first phase the algorithm performs countingsort again. The sort key now is the character after the lcp of the c-group. This is done recursively, as long as intervals contain more than one suffix.

Although its average case runtime is $O(n \log n)$, WOTD compares well with $O(n)$ suffix tree algorithms. In Giegerich *et al*. (2003), it is shown to perform similar to the McCreight algorithm for sequences up to length 4 MB. This results from the simplicity and good locality behaviour on modern cached CPU architectures. Our new algorithm inherits these attributes.

2.2 GUUGle

Let $\Sigma = \{A, G, C, U, \#, N\}$ be the alphabet. The elements A, G, C, U represent the standard RNA bases, arginin, guanin, cytosine and uracil. N denotes any character. Every unknown character will be substituted by N. # denotes the separation character between neighbouring sequences. We define the following order on our alphabet: $A < G < C < U < \# < N$.

The idea is to develop two suffix arrays, target and query, in parallel. Intervals in the target are only built when the corresponding complementary interval in query is not empty and vice versa. Those complementary intervals represent reverse complementary matches between both the sequences. To account for the 'reverse', the offset chosen for suffixes in the query suffix array is just the negative of the offset in the target.

In case of an A-interval in the target suffix array, the complentary interval in query is the U-interval with corresponding negative offset. In the same way, the C group in the target is matched to the G group in the query. For G and U it is a bit more complicated, since both of them have two complementary bases. If the suffix array was ordered lexicographically, the complementary bases of G, C and U, would be split into two intervals. For each interval in the target we would have to keep track of an exploding number of intervals in the query as we proceed recursively. By choosing the order $A < G < C < U$ the groups $A + G$, and $C + U$ are contiguous intervals and we avoid this splitting.

While the recursion in WOTD can proceed either depth-first or breadth-first, it is essential for GUUGle to work in depth-first mode (Fig. 1). All matches derived from the A/U group in phase 1 must be completely determined before phase 3 is entered. Since each interval is reordered in place, phase 3 shuffles suffixes in the joint $C + U$ interval, destroying their original separation in C groups and U groups.

In each recursion step, suffixes with #s or Ns at the actual offset will no longer be considered, since #s represent the end of a suffix and matching with Ns would produce too many artefacts. This is done by sorting these suffixes to the right end of the interval and excluding them from the next recursion step. The matches are reported when the offset reaches the threshold k. Then, for each pair of suffixes the program checks for left- and right-maximality, as only matches that cannot be extended are reported.

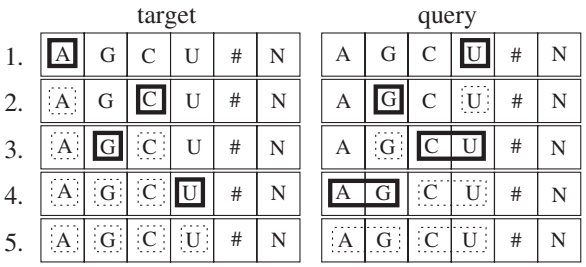

Fig. 1. Development of the suffix arrays target (left) and query (right). The thick rectangulars denote performing of countingsort and recursion on intervals. Dotted rectangulars denote already sorted intervals.

3 EXPERIMENTAL RESULTS

MicroRNAs are known to play an important role in gene-regulation and cell-differentiation. Many microRNAs are known but some of their target sites in the genome still have to be discovered. Energy based folding algorithms [like Stark *et al*. (2003); Enright *et al*. (2003); Rajweski and Socci (2004) and RNAhybrid (Rehmsmeier *et al*., 2004)] are a promising way to find such binding sites. It has been observed that the binding sites are conserved in their 3′-site (target) and form alpha-helices with the microRNA 5′-ends, owing to the exact base-pairing in this region mostly from nucleotide 2 to nucleotide 7 or 8 (Rehmsmeier *et al*., 2004). Thus, for some algorithms such a 'seed' of perfect helix is necessary for a pair of microRNA and possible target site. It has been shown that this structural constraint increases statistical significance by reducing the search space (Lewis *et al*., 2003).

Using this fact, it should be possible to speedup RNAhybrid by filtering with GUUGle. GUUGle searches for all matches in the target sequences and reports them, including 20 bases before and after the match. This output is much smaller than the original sequence, as it contains only regions with matches. We compared the time RNAhybrid needed to find microRNA target sites in GUUGle-output and in the original Fasta file.

The test has been performed under a Sun Fire V20z with 1.8 GHz AMD Opteron 244-processors and 6 GB RAM. A total of 27 692 human downstream UTRs (∼200 MB in total) have been searched against a 2–8 seed heptamer 'GAGGUAG' from the hsa-let-7a microRNA.

For the original fasta file RNAhybrid took 2860 s. In the second run, we filtered the data with GUUGle, which took ∼20 s. For the filtered data RNAhybrid now took ∼163 s. This is more than 15 times faster than without filtering.

ACKNOWLEDGEMENTS

The authors thank John Mattick, who first pointed out the lack of such a program and Marc Rehmsmeier who helped with the measurements. Funding to pay the Open Access publication charges was provided by Bielefeld University.

Conflict of Interest: none declared.

REFERENCES

Abouelhoda,M.I., Kurtz,S. and Ohlebusch,E. (2002) The Enhanced Suffix Array and its Applications to Genome Analysis. In *Proceedings of the Second Workshop on Algorithms in Bioinformatics, LNCS 2452,* Springer Verlag, pp. 449–463.

Enright,A.J. *et al*. (2003) MicroRNA targets in *Drosophila*. *Genome Biol*., **5**, R1.

Giegerich,R. *et al*. (2003) Efficient implementation of lazy suffix trees. *Software Pract. Exp*., **33**, 1035–1049.

Horesh,Y. *et al*. (2003) A rapid method for detection of putative RNAi target genes in genomic data. *Bioinformatics*, **19** (Suppl. 2), II73–II80.

Kurtz,S. *et al*. (2001) REPuter: the manifold applications of repeat analysis on a genomic scale. *Nucleic Acids Res*., **29**, 4633–4642.

Lewis,B.P. *et al*. (2003) Prediction of mammalian microRNA targets. *Cell*, **115**, 787–798.

Nicolas,J. *et al*. (2005) Suffix-tree analyser (STAN): looking for nucleotidic and peptidic patterns in chromosomes. *Bioinformatics*, **21**, 4408–4410.

Rajweski,N. and Socci,N.D. (2004) Computational identification of microRNA targets. *Dev. Biol*., **267**, 529–535.

Rehmsmeier,M. *et al*. (2004) Fast and effective prediction of microRNA/target duplexes. *RNA*, **10**, 1507–1517.

Stark,A. *et al*. (2003) Identification of *Drosophila* MicroRNA targets. *PLoS Biol*., **1**, 1–60.