Excel@FIT 2025
http://excel.fit.vutbr.cz

LTR retrotransposon detection via deterministic finite automata

Lucie Klímová*

Abstract
LTR retrotransposons are often inserted into one another, which makes them hard to detect. This paper shows that it is possible to use deterministic finite automata (DFA) to accelerate the computation. Several tools can detect these transposable elements but vary widely in their runtime, sensitivity, specificity, and capability of detecting nesting. We decided to modify TE-greedy nester [1] because it can locate even highly nested retrotransposons. To localize a transposon, it is necessary to detect its structural domains. We introduced a new method to generate DFA-based models representing these domains, which can be used for efficient domain search.

*xklimo04@vut.cz, *Faculty of Information Technology, Brno University of Technology*

1. Introduction

The genetic information of most eukaryotic organisms contains transposable elements (TEs) inserted into the DNA sequence throughout evolution. LTR retrotransposons constitute approximately 8.3% of the human genome [2], as illustrated in Figure 1. It is a significant part of the genetic information, compared to the 1.5% that is constituted by protein-coding genes [3].

Some retrotransposons may be non-functional or have neutral effects; others have been found to play a crucial role in genome evolution and have regulatory functions, such as controlling gene expression [4]. It is, therefore, essential to localize them and determine the order in which they were nested.

The main task is to create a program that, given a nucleotide or amino acid sequence as input, can find most of the TEs in a reasonable time. The main complications are the frequent nesting of TEs and mutations, which make it impossible to use exact matching algorithms.

Existing tools capable of detecting LTR transposons include, for example, LTR finder [5], which is relatively fast but unable to identify nested TEs. Another tool named RepeatMasker [6] first locates fragments of structural elements that could be part of a transposon and then tries to connect closely located fragments to form a whole TE, thus being able to detect some nesting. The tool we found the most interesting is TE-greedy nester. It can detect even deep nesting, but due to the recursive call of the algorithm on the entire input sequence, it appears relatively slow.

Since we found experimentally that more than 80% of the TE-greedy nester's runtime is taken up by calling a tool named BLASTX [7], we decided to use an alternative algorithm based on deterministic finite automata that could replace this slow part and thus speed up the whole process. Replacing the BLASTX tool with the DFA-based algorithm could significantly speed up the TE-greedy nester and enable the search for transposons even in longer sequences.

2. LTR retrotransposons

LTR retrotransposons are a type of TE that make up a significant part of the genome of many species. They consist of two long terminal repeats (LTRs), typically 250–600 bp in length, at both 5' and 3' ends of the retrotransposon, as shown in Figure 2. Between these two LTRs is a coding region, approximately 5–7 kb long, that contains at least two genes, gag and pol, but the number can vary depending on the type of transposon. These genes encode proteins such as protease (PR) and reverse transcriptase (RT), which are necessary for the transposon to replicate and move along the host DNA [8].

2.1 LTR retrotransposon detection using TE-greedy nester

TE-greedy nester [1] is a command-line tool that can detect even deeply nested LTR retrotransposons. Since older transposons are often fragmented by later inserted transposons, as shown in Figure 3, the program first locates the newest TE, which is then cut out of the original sequence, and the algorithm is repeated until no other transposon is found. This algorithm is described in Figure 4

3. Profile hidden Markov models

Profile hidden Markov models (PHMMs) are statistical models widely used in bioinformatics. They can precisely model the character of the searched sequence because they also consider that some positions are more prone to mutations than others. A PHMM structure is shown in Figure 5.

Nevertheless, the main disadvantage of PHMMs is that they are nondeterministic. Pattern matching using a nondeterministic automaton is possible, but it is computationally expensive due to the large number of different runs that must be evaluated. Dynamic programming algorithms, such as the Viterbi algorithm [9], reduce the time complexity to $O(LM^2)$, but compared to the deterministic alternatives, the execution time is still relatively high.

4. Proposed homology search algorithm

The main idea of this work is that if a PHMM, representing the searched sequence, could be transformed into a simplified DFA while minimizing the loss of accuracy, it would be possible to search for homologue sequences with time complexity of $O(L)$.

4.1 Bounded counting automata

For the purposes of the presented algorithm, we introduce a bounded counting automaton (BCA). BCA is an extended version of the classical nondeterministic finite automaton. It has a counter that is updated when a transition is made. The counter is bounded, so transitions that would cause the counter to exceed the bound b are not allowed.

4.2 Conversion into bounded counting automata

Direct determinization of a PHMM is unfeasible as it would result in an unacceptably large automaton. Fortunately, there are several ways to significantly reduce the size of the resulting automaton while maintaining a reasonable accuracy. The first step is to convert the transition and emission probabilities using a threshold t into discrete values that indicate how much a given transition deviates from the sequences in the database. Next, we can take advantage of the regular structure of the profile HMMs, which can easily be divided into p smaller parts. Each part represents a subsequence of the modelled domain and can be determinized separately.

4.3 Homology search algorithm

Each of the created DFAs is used to efficiently search for occurrences of the corresponding subsequence of the domain. The DFAs must localize most of the subsequence occurrences. Thus, they produce a lot of false positive results. However, if we say that for the result to be valid, we need to find at least m closely located subsequences in the correct order, this number is significantly reduced.

4.4 Results

The optimal settings of the four parameters p, t, b, and m are crucial. The higher p, t, and b are, the more distant the searched sequence can be. On the other hand, the less rigid the criteria are, the more false positive results we get and the more time it will take to filter these results.

When evaluating the algorithm, it is necessary to focus on two main aspects: accuracy and execution time. The algorithm speed highly depends on the query sequence length. Figure 7 shows the execution times for different query sequences.

In order to test the sensitivity of the algorithm, artificial sequences with different degrees of identity were generated. The comparison of the sensitivity of different parameter settings and the sensitivity of the BLASTP tool is shown in Figure 8.

5. Conclusions

Even though the implementation of the algorithm is only a prototype, the execution time measurement showed that the proposed algorithm is around 10 times faster than the BLASTP tool. In addition, the algorithm could be further accelerated if the query sequence does not have to be processed p times. It may be possible to continuously switch between the DFAs depending on the current subsequence found.

Also, in future work, we would like to introduce some result evaluation systems that could be used to filter the results and reduce the number of false positives.

Acknowledgements

I would like to thank my supervisors doc. Mgr. Lukáš Holík, Ph.D. and Mgr. Juraj Síč for their help.

References

[1] Matej Lexa, Pavel Jedlicka, Ivan Vanat, Michal Cervenansky, and Eduard Kejnovsky. Te-greedy-nester: structure-based detection of ltr retrotransposons and their nesting. *Bioinformatics*, 36(20):4991–4999, 07 2020.

[2] Richard Cordaux and Mark Batzer. Cordaux r, batzer mathe impact of retrotransposons on human genome evolution. nat rev genet 10: 691-703. *Nature reviews. Genetics*, 10:691–703, 10 2009.

[3] Chris Ponting and Ross Hardison. What fraction of human genome is functional? *Genome research*, 21:1769–76, 08 2011.

[4] Reyad Elbarbary, Bronwyn Lucas, and Lynne Maquat. Retrotransposons as regulators of gene expression. *Science*, 351, 02 2016.

[5] Zhao Xu and Hao Wang. Ltr_finder: an efficient tool for the prediction of full-length ltr retrotransposons. *Nucleic Acids Research*, 35(suppl_2):W265–W268, 07 2007.

[6] AFA Smit, R Hubley, and P Green. Repeatmasker open-4.0, 2013–2015. Available from http://www.repeatmasker.org.

[7] Stephen F. Altschul, Warren Gish, Webb Miller, Eugene W. Myers, and David J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, 1990.

[8] Camille Thomas-Bulle, Mathieu Piednoël, Tifenn Donnart, Jonathan Filee, Didier Jollivet, and Éric Bonnivard. Mollusc genomes reveal variability in patterns of ltr-retrotransposons dynamics. *BMC Genomics*, 19, 11 2018.

[9] Andrew Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE transactions on Information Theory*, 13(2):260–269, 1967.