BIOINFORMATICS APPLICATIONS NOTE

Short read alignment

Error-Tolerant Short Read Mapping using the Burrows-Wheeler Transform (BWT)

Sam Tkach[1] and Kenneth Xiao[2]

[1]Undergraduate in Department of Computer Science, Columbia University, [2]Undergraduate in Department of Computer Science, Columbia University

Supervising Professor: Itshack Pe'er

Received on May 9, 2016

Abstract

Summary: This application implements error-tolerant backward search building off the base algorithm of BWA, proposed in Heng Li and Richard Durbin's original paper (see references). We use different search pruning techniques in an attempt to tolerate higher error rates efficiently.

Results: Tested on reference genomes of various viruses, our application aligns short reads with sufficient accuracy and efficiency.

Availability: Implemented in Python 2.7, code freely available at www.github.com/st2794/cs4761-team-y

Contact: st2794@columbia.edu, ksx2101@columbia.edu

1 Introduction

Since next generation genomic sequencing techniques provided an application of the Burrows-Wheeler Transform in bioinformatics, many algorithms and tools have been proposed to utilize it for read-mapping and sequencing, especially for the purpose of reduction in memory costs (e.g.- BWA, Bowtie). This is in large part due to the high computational cost of aligning many millions of short reads.

The Burrows-Wheeler Transform is composed of the last column of the matrix of all possible suffixes of a reference string in lexicographic order. The first column can be reconstructed by sorting, and index ranges in the first column correspond to sets of substrings. The Backwards Search algorithm works by narrowing down the indices of this range at each iteration/character in the match string.

We have implemented an expanded version of the base inexact backwards search algorithm originally proposed by Li and Durbin in their 2009 paper and implemented in their alignment software tool, BWA. We seek to improve on the ability of the aligner to tolerate errors by including certain metrics with which we can effectively prune the set of substrings being considered as matches. The proposed applications of this method center around short-read mapping on genome data that is either incomplete, high-error, or from a related species.

2 Algorithm

As previously stated, our search algorithm is a variant of BWA's base inexact search. This search relies on FM-indexing (Ferragina, Manzini; 2000), which is similar to suffix array indexing, but allows for a faster and memory efficient search. The scheme is effectively as follows: we define $C(a) := count(c)\ in\ BWT\ for\ all\ c < a$, and $O(a, k) := count(a)\ in\ BWT[1, k]$, where *a* and *c* are bases/characters and *BWT* is the Burrows-Wheeler Transform of some data. We can now define the last-to-first mapping as $LF(i) = C(BTW[i]) + O(BWT[i],i)$. This allows the implementation of exact matching with backwards search. To extend this to inexact matching, we effectively keep track of the number of mistakes present in each substring (i.e. at each index of the BWT) within the search. This is functionally equivalent to traversing the prefix tree of the reference sequence. Li and Durbin outline two methods for 'pruning' the execution tree of this recursive program. First, there is a vector D[*i*] that contains estimates (heuristically calculated) of the number of errors in the substring match up to index *i*. We can loosely assume that paths that have a number of errors greater than D[*i*] before it reaches the index *i* in the search are too costly. Second, they propose a hard limit on the number of mistakes (substitutions or indels) that are allowed in the whole match.

We have extended this algorithm by adding a method for scoring the quality of matches, as well as accounting for gaps in a more intelligent way. We implement affine gap penalties in an attempt to replicate some of the success BWA-SW (Li, Durbin; 2010) sees in this area. We considered also applying convex gap penalties and interpolating these two methods, but it has been shown that convex gaps are generally not nearly as useful as affine ones in this setting (Cartwright, Reed; 2006). Affine gap penalties are implemented with little overhead by simply having a state for each recursive call, and conditions that check that state on each call. Next, adding a custom substitution matrix that is calculated as the maximum likelihood of each mismatch over each possible non-gapped alignment, which approximates the probability of each transition as well as possible without further reference material or information. We addi-

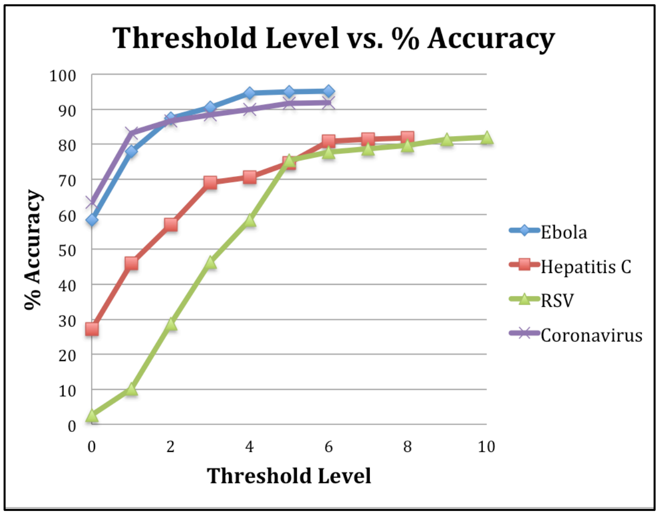

Figure 1: Relationship between threshold level and percent accuracy for various viruses.

tionally tested some random pruning techniques, where an error threshold would be generated from a discrete normal distribution, such that it effectively cut some paths off early, and gave some a "second chance." We noticed no significant improvement in accuracy or performance, so this feature was scrapped.

3 Results

All of our data was obtained from the National Center for Biotechnology Information's GenBank and Sequence Read Archive (SRA).

Figure 1 illustrates that as the threshold level increases, the percent accuracy of our aligner increases, because more "branches" are explored to a greater depth. Percent accuracy is calculated as the percentage of reads that our aligner predicted to have the same position as their true position. These tests were conducted on viral genomes and reads.

It is very important to note that these tests were run with the following consistent parameters: gap open penalty = 3, gap extension penalty = 1, mismatch penalty = 1, match reward = 0. These parameters were clearly very favorable for Ebola and Hepatitis C alignment. Other parameters are more suitable for RSV and MERS-Coronavirus, but this graph is only aiming to measure changes in threshold level. The maximum threshold level tested differs among viruses due to constraints on running time (see Figure 2); thus, threshold levels were increased until percent accuracy plateaued for each virus.

Figure 2a illustrates the relationship between threshold level and time. The algorithmic complexity is exponential in the threshold level. Analogously, Figure 2b illustrates the relationship between threshold level and the number of nodes pruned.

When testing our algorithm, we started by using references and reads of various error rates, but that were crucially from the same species, and in some cases the same organism. In these test cases on average, we found the error in selecting the correct best match to be 5.1%. Of this fraction, ~60% had the best alignment appear in the top 5 matches returned, suggesting that with a more sophisticated method for estimating parameters, the performance could be improved.

We noticed that, as evident in Figure 1, there is a decreasing marginal return on accuracy above a certain error threshold value. We found there to be a plateau in accuracy vs. allowed mistakes, because of the relative limitations to this method in contrast to something like dynamic programming. Exact search of BWT data is fast, but inexact search is an

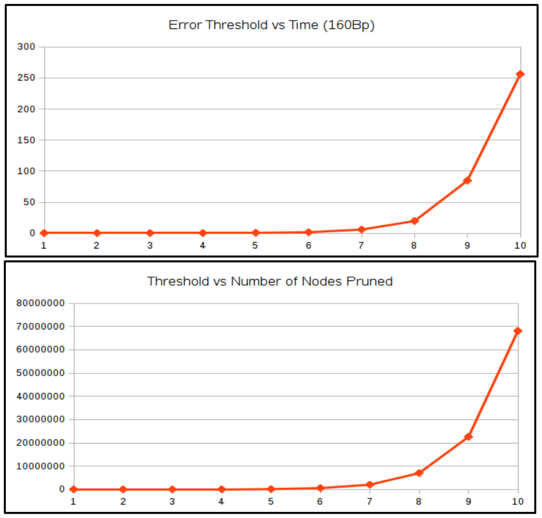

Figure 2a (top): Relationship between threshold level and time.
Figure 2b (bottom): Analogously, relationship between threshold level and number of nodes pruned. Tested on Hepatitis C data.

exponential process with respect to error threshold value, and it seems that the threshold could not be pushed high enough to overcome the plateau of accuracy we experienced. Testing this method on Dengue virus reads and the Ebola virus genome, we experienced no success. So although this method works relatively well (considering the lack of optimization, the implementation language, and the timeframe), we posit that it will not surpass other methods of sequence alignment.

There are many further improvements that could be made to this algorithm. Organizing the recursive calls into a heap structure would allow us to do inexact searching more quickly and efficiently. Analysis of reference reads and genomes could be used to estimate the penalty parameters for matching to a similar organism. Incorporation of a seed string matching method might allow for more flexibility in the context of the matches we were finding. This was the main challenge of this project—it was difficult to know which changes we could make (to an already excellent algorithm) that would improve performance the most.

Acknowledgements

We would like to thank Professor Itshack Pe'er and Shuo Yang for their hard work and a great semester. As graduating seniors, we are grateful to the Columbia University Department of Computer Science for providing us with an excellent undergraduate educational experience.

Conflict of Interest: none declared.

References

Cartwright, Reed. (2006) Logarithmic gap costs decrease alignment accuracy. *BMC Bioinformatics.* Department of Genetics, University of Georgia.

Ferragina Paolo and Giovanni Manzini. (2000) Opportunistic Data Structures with Applications. *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, 390.

Langmead, B., Trapnell, C., Pop, M., and S.L. Salzberg. (2009) Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biology*. Center for Bioinformatics and Computational Biology, University of Maryland.

Li, Heng and Richard Durbin. (2009) Fast and accurate short read alignment with Burrows-Wheeler transform. Wellcome Trust Sanger Institute, Cambridge, UK.

Li, Heng and Richard Durbin. (2010) Fast and accurate long read alignment with Burrows-Wheeler transform. Wellcome Trust Sanger Institute, Cambridge, UK.