Resolution-length distribution is a statistical property of datasets and indexes in random-permutations-based DNA strings analysis. This property also affects other algorithms used for the same purposes.
We report on results of preliminary analysis of basic resolution-length distribution properties of human DNA, with implications for random-permutations-based algorithms. We also discuss some of the implications for other algorithms which are used for DNA analysis.

A Note about the Resolution-Length Characteristics of DNA

Roy R. Lederman[†]
Technical Report YALEU/DCS/TR-1473
April 2, 2013

[†] Applied Mathematics Program, Yale University, New Haven CT 06511

Keywords: *DNA, Resolution-Length, Permutation, Alignment, Mapping.*

1 Introduction

Various algorithms are currently used for different types of analysis of DNA strings, and for processing reads from DNA sequencing. Some of the key processing applications and some of the algorithms used for these applications are described in [1, 2, 3, 4, 5, 6]

In [7], we describe a random-permutations-based approach to read alignment. In other reports, we discuss extensions and additional applications for this approach. For more information and technical reports, see http://alignment.commons.yale.edu

In the description of the approach, we introduced resolution-length, a statistical property of a string in the context of a library of strings. Resolution-lengths and the distribution of resolution-lengths for different strings play a significant role in random-permutations-based search. Analysis of these distributions can also contribute to the analysis and visualization of other algorithms used in DNA read processing.

Here, we consider the resolution-length properties of libraries of substrings of real DNA strings, and of human DNA in particular. We examine how permutation of strings in libraries influences basic resolution-length characteristics of the libraries.

2 Preliminaries

The discussion in this technical report is based on the description of permutations-based-alignment in [7].

Definition 2.1 *The permutation-length is the length of the prefix to which a permutation is applied. The remaining characters in the string remain in the same position. We denote the permutation-length by M_2.*

In other words, given strings of length M_1, we choose $M_2 \leq M_1$. We permute the first M_2 characters of the strings, leaving the last $M_1 - M_2$ characters untouched. When no permutation is applied, we can say that $M_2 = 0$ or $M_2 = 1$.

Definition 2.2 *A neighborhood size, denoted by K, is the maximum size of the items which we allow in a list of "candidate neighbors".*

In the context of random-permutations-based search, this is the size of lists of candidates. This number is also related to sizes bins of seeds in hash-based searches.

Definition 2.3 *The resolution-length of a sting with respect to a library of strings, is the smallest length of the prefix which is shared by no more than K strings in the library.*

In other words, it is the number of characters at which we have to "touch" until we narrow down the number of matches to at most K. The resolution-length is denoted by L.

Different strings can have different resolution-lengths in respect to a given library. Different neighborhood sizes can produce different resolutions-lengths. The same string can have different resolution-lengths with respect to different libraries. Different permutations result in different resolution-lengths of the permuted strings with respect to the library of permuted strings.

3 Methods

We consider a library of substrings of length M_1 from the reference. We define some neighborhood size, denoted by K.

For this library, we are interested in the resolution-length of each string in the library. In other words, had we been looking for a particular string in the library, we would like to know how many characters in the prefix of that string we would have to "touch" before we narrow down the number of possible strings to no more than K.

To do this, we build a lexicographically sorted array of the strings in the library. We then examine the neighborhood of each string, to find the resolution-length of each of them.

We would also like to examine how permutations affect the resolution-length. We build several libraries with different values of M_2, apply a permutation to the M_2 prefixes of the strings in each library (the same permutation to all strings in the same library), and examine all of them.

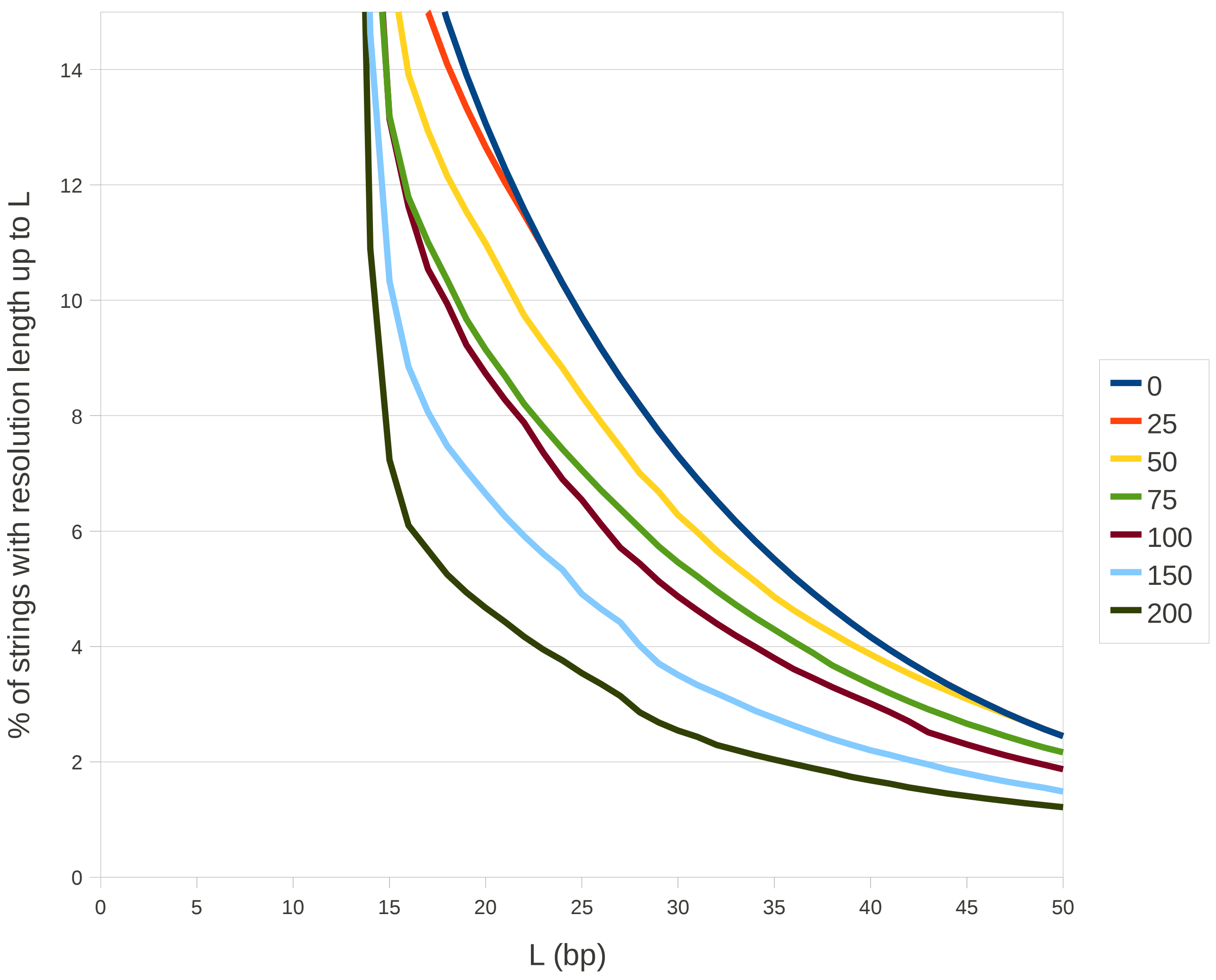
4 Implementation and results

We used indexes generated by a prototype aligner [7] to study the resolution-length characteristics of human DNA.

We chose several values of permutation-lengths M_2. For each value we generated a single random permutation and calculated the resolution-length of each string for different neighborhood sizes K.

A sample of the results is presented in figure 1.

Figure 1: Resolution-length distribution



The % of strings with resolution-length up to L characters.

Human genome, single direction. $K = 10$

5 Conclusions

A preliminary study has been conducted to examine the statistics of DNA substrings in the context of permutations-based search algorithms. The results may also be useful in examining other algorithms.

The results indicate that the permutation of substrings of the DNA produce shorter resolution-length in searches. In other words, when a library of DNA substrings is permuted, fewer characters have to be "touched" to produce a small list of possible search results. Longer permutations produce shorter resolution-lengths.

When performing searches in the presence of mismatches, these shorter resolution-lengths imply higher probabilities of success.

Some aspects of the known problem of optimizing the length of hashed strings in hash-table-based algorithms are also demonstrated here. Very short seeds produce very large bins, whereas long seeds decrease the probability of hashing "error free" segments. Adding a small number of consecutive characters does not reduce the size of the bins as much as one would expect in random i.i.d strings, so even relatively long seeds have many large bins. Gapped seeds still consider a fixed number of characters in each seed, but these character are chosen from a longer range in the substrings. Therefore, gapped seeds can reduce the size of bins, compared to seeds of consecutive characters (with the same seed length).

In view of these results, basic permutations-based-algorithm can be loosely described as a way of removing the rigid seed-length requirement from hash-tables-based algorithms.

More generally, the results reported here demonstrate the strong dependence between characters in the DNA, and demonstrates that the strength of this relationship deceases as the distance between the characters increase.

One possibility that comes out of this analysis, is to consider using "gapped k-mers" as possible elements in de-Bruijn graphs used for assembly. De-Bruijn-graph based assembly uses overlapping short k-mers. Slightly longer k-mers do not reduce the ambiguity considerably. However, "gapped-k-mers" contain information from longer ranges in reads. While they may complicate the graphs and increase the number of theoretical edges, they may reduce the ambiguity in resolving "gapped-de-Bruijn-like" graphs.

6 Acknowledgments

We would like to thank the Yale University Biomedical High Performance Computing Center, NIH grants RR19895 and RR029676-01, for providing some of the computation resources.

References

[1] Paul Flicek and Ewan Birney. Sense from sequence reads: methods for alignment and assembly. *Nature methods*, 6(11 Suppl):S6–S12, November 2009. PMID: 19844229.

[2] Heng Li and Nils Homer. A survey of sequence alignment algorithms for next-generation sequencing. *Briefings in bioinformatics*, 11(5):473–483, September 2010. PMID: 20460430.

[3] Jason R. Miller, Sergey Koren, and Granger Sutton. Assembly algorithms for next-generation sequencing data. *Genomics*, 95(6):315–327, June 2010.

[4] Michael C Schatz, Arthur L Delcher, and Steven L Salzberg. Assembly of large genomes using second-generation sequencing. *Genome Research*, 20(9):1165–1173, May 2010.

[5] Zhenyu Li, Yanxiang Chen, Desheng Mu, Jianying Yuan, Yujian Shi, Hao Zhang, Jun Gan, Nan Li, Xuesong Hu, Binghang Liu, et al. Comparison of the two major classes of assembly algorithms: overlap-layout-consensus and de-bruijn-graph. *Briefings in Functional Genomics*, 11(1):25–37, December 2011.

[6] Nuno A Fonseca, Johan Rung, Alvis Brazma, and John C Marioni. Tools for mapping high-throughput sequencing data. *Bioinformatics (Oxford, England)*, 28(24):3169–3177, December 2012. PMID: 23060614.

[7] Roy R. Lederman. A random-permutations-based approach to fast read alignment. *BMC bioinformatics*, 14(Suppl 5):S8, 2013.

Revised January 7, 2016 (no change in content).