ON HEURISTIC METHODS IN NEXT-GENERATION SEQUENCING DATA ANALYSIS

Ivan Vogel
Doctoral Degree Programme (1), FIT BUT
E-mail: xvogel01@stud.fit.vutbr.cz

Supervised by: Jaroslav Zendulka
E-mail: zendulka@fit.vutbr.cz

Abstract: This paper presents an advanced workflow for repetitive element reconstruction from the sequenced genomes . This workflow is good motivation for optimizations concerning the fast similarity sequence search among NGS data. Some hashing techniques such as locality-sensitive hashing and positional hashing for fast similarity candidate estimations are proposed, as well as the algorithm BayesLSH which combines Bayesian inference and locality-sensitive hashing. A new method that combines Bayesian inference and positional hashing is proposed.

Keywords: transposable elements, graph-based clustering, Bayesian inference, locality-sensitive hashing, positional hashing

1 INTRODUCTION

Next-generation sequencing (hereinafter NGS) methods provide us nowadays with a huge amount of data usually consisting of millions of sequence reads relatively short in length (100-300 kilobases) [1]. By applying advanced analysis techniques, we are able to reconstruct de-novo a great variety of genome sequences (especially repetitive ones), though just with low genome sequencing coverage (usually about 5%). First, an advanced workflow for obtaining repetitive element families will be presented as a motivation for optimizing certain steps of the presented analysis. Finally, algorithms for fast similarity search will be described and future research will be presented.

2 BIOLOGICAL BACKGROUND

A transposable element (hereinafter TE) is a DNA sequence able to change its relative position (self-transposition) within the genome of a single cell. The mechanism of self-transposition is directly connected with its division into two main classes: **Class I (retrotransposons)** - these use a so-called "copy-and-paste" mechanism, which means they copy themselves in two stages. First, they are transcribed from DNA to RNA through transcription, and consequently from RNA back to DNA by reverse transcription. The DNA is then inserted into the genome in a new position. Reverse transcription is often catalyzed by a reverse transcriptase, which is often coded by the TE itself; **Class II (DNA transposons)** - these, in contrast to retrotransposons, do not use the RNA intermediate for their transposition, but rather the so-called cut-and-paste principle.

Thanks to the aforementioned copy mechanisms, the transposable elements are often referred to as so-called repetitive sequences, as there are hundreds of more or less divergent copies of the same transposable element families represented in the genome of a specific organism. In the following subsequences, mostly Class I transposons reconstruction will be discussed in a more detailed manner.

3 NGS WORKFLOW FOR TRANSPOSABLE ELEMENT RECONSTRUCTION

In order to get concrete transposable element families from the genome, the following steps must be performed:

1. Genome Sampling – the high-throughput NGS machine scans the genome randomly and outputs several millions of short sequence reads. Although the number of received reads from an NGS run is actually insufficient for full genome assembly, it is mostly intended for detecting elements which are fully or partially copied across the genome in many copies.

2. Data Cleaning – sometimes identical reads occur in the experiment, which is a technical artefact. Redundant reads are removed in this step.

3. Data Transformation – one-to-one pairwise comparison is performed and all read pairs with significant sequence overlaps meeting the specified threshold are recorded (often at least more than 90 % identical on at least 55 % of the longer sequence [2]). Consequently, a graph is constructed, on which the vertices correspond to sequence reads, and overlapping reads are connected with edges and their similarity score is expressed as an edge weight. Singletons in this graph structure point to single-copy sequences, sequence reads of repetitive elements build groups of mutually connected nodes.

4. Graph-Based Clustering –involves optimal graph partitioning into communities. The algorithm works in an agglomerative hierarchical manner, builds a tree, and points out an optimal tree cut to obtain well-partitioned clusters [2].

5. Sequence Reconstruction (contig assembly and consensus estimation) - from the previous step, we have clusters of reads representing concrete repetitive element families. The remaining task is to assemble concrete nucleotide sequences from the cluster reads. This involves multiple steps which will not be covered in this paper. The idea is to detect sequence overlaps in the reads within one cluster and make so-called contigs (assembled sequences) from them.

Notice that points 2, 3 and 5 include one-to-one pairwise comparison (detecting identical reads, comparing reads to build a graph structure and finally performing a multiple sequence alignment as part of the final sequence assembly). In this workflow, algorithm BLAST [4] is used as a standard. Although BLAST already contains heuristic methods, this is still unefficient for one-to-one pairwise comparison. There is a need for a fast and accurate enough method for building so-called similarity candidates, that is pairs of sequences very likely to be similar according to the specified similarity threshold. In the next subsection, various methods for selecting candidate sequence pairs will be presented.

4 SEQUENCE SIMILARITY ESTIMATION

The idea is to create minimized disjunct subsets of the original dataset (candidate pairs). Each subset consists of elements which meet the specified similarity threshold criterion and will be analyzed further.

4.1 LOCALITY-SENSITIVE HASHING

Consider two strings s_1 and s_2 of the common length d over an alphabet Σ (actually, $\Sigma = \{A, T, C, G, -\}$). We call s_1 and s_2 similar, if there is r, $r < d$ and s_1 and s_2 differ at most r single-character substitutions [6].

Let's define the function $f : \Sigma^d \rightarrow \Sigma^k$ by

$$f(s) = \langle s[i_1], s[i_2], \ldots, s[i_k] \rangle. \tag{1}$$

The function f is called a locality-sensitive hash function. It samples the nucleotide sequence by randomly choosing k characters and building the so-called k-mer or an LSH value. There is a direct relationship between the similarity of two sequences and the probability that they hash to the same LSH value and is given by the following equation:

$$Pr[f(s_1) = f(s_2)] \geq (1 - \frac{r}{d})^k \tag{2}$$

False positives (that is, s_1 and s_2 are dissimilar, but $f(s_1) = f(s_2)$) can be distinguished from true positives by counting the number of positions at which s_1 and s_2 disagree. False negatives cannot be detected efficiently, but prevention of this phenomenon can by made applying multiple independent random hash functions [6].

4.2 Positional Hashing

Positional hashing [5] is a generalized locality sensitive function in 2D space. It consists of a so-called sampling template in a two-dimensional matrix, as illustrated in Figure 1B, which means each sequence in the dataset is sampled both in a horizontal and vertical sampling position. This can clearly produce two different hash keys per sequence, depending on the hashing strategy. A positional hash table is made, including every key from the hashing scheme and pointing to the sequences that share the same key (horizontal and vertical, vertical and horizontal, or both). According to this table, a pairwise similarity hash table is made, where each sequence pair sharing at least one substring is present as a hash key, pointing to the common substrings of the particular pair . Browsing the pairwise similarity hash table brings sufficient information about the sequence similarity among the dataset.

4.3 Bayes Locality Sensitive Hashing

The similarity estimation using locality hash functions is as follows - having n hashes and observed m agreements in hash values, the maximum likelihood estimator is $\hat{s} = \frac{m}{n}$. The variance for the estimator is $\frac{s*(1-s)}{n}$. This indicates that in order to obtain the same level of accuracy for different similarities, it is necessary to use a different number of hashes (see Figure 1A for illustration).

The algorithm BayesLSH [3] solves the abovementioned problem. It combines Bayesian inference with LSH. To be more precise in explanation, it has the following advantages over the classic LSH approach: the algorithm can prune away many false positive candidate pairs by examining only the first few hashes; the number of hashes for which each candidate pair is compared is determined automatically by the algorithm, completely eliminating the need to set the number of hashes manually

The formulas for the proper construction of the BayesLSH algorithm are derived in [3]. In this paper, we simply apply mathematical solutions to three crucial questions.

Let us consider a pair (x,y) and say that m out of the first n hashes match for this pair. Let us denote this event as $M(m,n)$. We therefore pose three questions:

1. When $M(m,n)$, what is the propability that the similarity is greater than the threshold t? Answer: $Pr[S \geq |M(m,n)] = \int_t^1 p(s|M(m,n))ds$
2. What is the similarity value with the highest posterior probability? Answer: $\hat{S} = arg\,max_s p(s|M(m,n))$
3. What is the probability that $\hat{S}$ is within δ of the true similarity? Answer: $Pr[|S - \hat{S}| < \delta \,|\, M(m,n)] = \int_{\hat{S}-\delta}^{\hat{S}+\delta} p(s|M(m,n))ds$

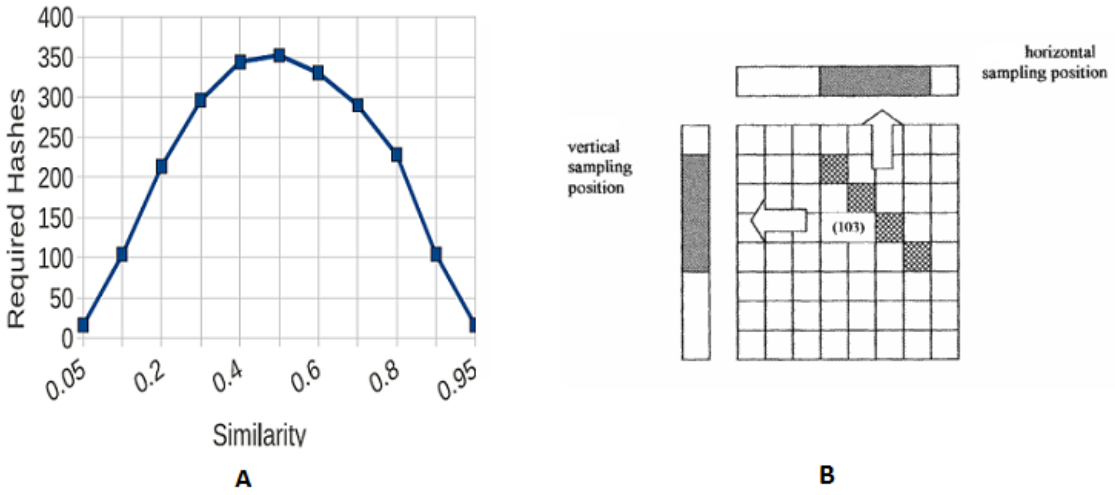

Figure 1: A:Hashes vs. Similarity; B: Positional Hashing

5 APPLICATION TO THE SEQUENCE READS ANALYSIS

In the future analysis and algorithm design, we will constrain on the following conditions: unlike in BLAST, there is no need for an exact alignment, only a similarity score is questioned; the sequence reads are of the same length; as the genome is sequenced, sequence overlaps between the reads occur (i.e. last 50 nucleotide bases of one sequence are identical to the first 50 nucleotide bases of another one), this needs to be considered as a particular similarity as well.

5.1 PROPOSED METHOD

We are looking for a solution that fulfills all the above-mentioned criteria. The proposed algorithm combines positional hashing with BayesLSH. Consider sequence reads each of the length l. Let's define function $Gen(l)$, that randomly generates a positional hashing function, that is, a pair (x,y). The x and y are randomly generated numbers, whereby $0<x,y \leq l$. Let's define another function $Select(x,y,k)$ that returns a positional hash (actually two hashes, each of length k). The modified algorithm, BayesPH, is described as follows (algorithm structure adapted from [3]):

Input: Set of candidate pairs C; Similarity treshold t; recall parameter ε; accuracy parameters δ,γ; length of k-mer to select with function $Select()$ k

Output: Set O of pairs (x,y) along with similarity estimates $\hat{S}_{x,y}$

for all $(x,y) \in C$ **do**
  $n,m \leftarrow 0$ {Initialization}
  $isPruned \leftarrow False$
  while $True$ **do**
    Generate k positional hash functions using $Gen(l)$
    Select the corresponding positional hashes using function $Select(x,y,k)$
    Put $m = m+1$ for every identical positional hash between sequence x and y
    $n = n+k$
    if $Pr[S \geq t|M(m,n)] < \varepsilon$ **then**
      $isPruned \leftarrow True$
      break {Prune candidate pair}
    end if
    $\hat{S} \leftarrow argmax_s p(s|M(m,n))$

```
        if Pr[|S − Ŝ|M(m,n) < δ] < γ then
            break {Similarity estimate is sufficiently concentrated}
        end if
    end while
    if isPruned == False then
        O ← O ∪ ((x,y), Ŝ)
    end if
end for
```

6 CONCLUSION AND FUTURE RESEARCH

This paper presented a reasoning for using optimized algorithms in regards to NGS data analysis. There were proposed some points of the concrete NGS data processing workflow that needs an improvement concerning the computational efficiency. One of the heuristics that can be applied is fast similarity search by locality sensitive hashing and its slightly different generalization - the positional hashing. On the other hand bayesian inference can bring fairly better results estimating the parameters that cannot be properly chosen by classic LSH approach (rejecting false positives at the very beginning of the analysis and optimizing the number of needed hashes). The aim of my future research is to experimentally proove the proposed concept of the BayesPH algorithm and to compare the results with some previously published tools like PASH [6] or well known BLAST algorithm, which is often used for the purpose of the one-to-one pairwise comparison in the practical analysis [2]. The method seems to be promising and acurrate enough for a fast similarity candidate search. Although it is crucial to choose a good hashing template strategy (Subsection 4.2) to effectively scan the compared sequences.

ACKNOWLEDGEMENT

This work was partially supported by the research plan MSM0021630528, the specific research grant FIT-S-11-2 and the IT4Innovations Centre of Excellence CZ.1.05/1.1.00/02.0070.

REFERENCES

[1] KIRCHER, M. a J. KELSO. High-throughput DNA Sequencing - Concepts and Limitation. BioEssays. 2010, Nr. 32, p. 524-536. ISSN 1521-1878.

[2] NOVAK, P., P. NEUMANN a J. MACAS. Graph-Based Clustering and Characterizaction of Repetitive Sequences in Next-Generation Sequencing Data and Medical Sciences. BMC Bioinformatics. 2010, Nr. 1, p. 524-536. ISSN 1471-2105.

[3] SATULURI, V. and PARTHASARATHY, S. Bayesian Locality Sensitive Hashing for Fast Similarity Search. CoRR. 2011.

[4] ALTSCHUL, S.F., GISH, W., MILLER, W., MYERS, E.W., LIPMAN, D.J.: Basic Local Alignment Search Tool. J Mol Biol. 1990, Nr. 3, p. 403-10

[5] MILOSAVLJEVIC, A.: Positional Hashing Method for Performing DNA Sequence Similarity Search. United States Patent, Woodlands, TX (US), 2010

[6] BUHLER, J. Efficient Large-Scale Sequence Comparison by Locality-Sensitive Hashing. Bioinformatics. 2001, Nr. 5. ISSN 1367-4803.