Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Identifying Differences Between Sequencing Data Sets

Diploma thesis

2016
Vladimír Macko

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Identifying Differences Between Sequencing Data Sets

Diploma thesis

Study programme:	Informatika
Field of study:	2508 Informatika
Department:	Katedra informatiky
Supervisor:	Mgr. Tomáš Vinař, PhD

Bratislava, 2016
Vladimír Macko

acknowledgment:

Abstrakt

Zjednodušene môžeme povedať, že sekvenačný beh je množina krátkych podreťazcov vzorkovaných z dlhého neznámeho reťazca. Pre dva zadané sekvenačné behy je úlohou identifikovať rozdiely medzi ich zodpovedajúcimi neznámymi reťazcami.

Tento problém bol v minulosti riešený kombinatorickými metódami pre prípady s veľkými sekvenačnými behmi, napr. keď množstvo dát v sekvenačnom behu bolo viac ako desať násobok dĺžky neznámeho reťazca. Cieľom tejto práce je vytvoriť nové metódy založené na pravdepodobnostnej interpretácii dát. Takýto prístup umožné riešiť problém v ďalších prípadoch, napr. pre malé alebo hybridné sekvenačné behy.

Kľúčové slová:

Abstract

In the simplest possible formulation, a sequencing data set is a set of short substrings sampled from an unknown string. Given two sequencing data sets, the goal is to identify differences between the two underlying unknown strings.

This task has been explored by combinatorial methods in scenarios with large sequencing data sets, where amount of data in the set is large (e.g., more than 10x of the length of the underlying string). The goal of this thesis is to develop new methods based on probabilistic interpretation of the data. Such approach may help to solve the problem in other scenarios, including small and hybrid data sets.

Keywords:

Contents

List of Figures

List of Tables

Introduction

Chapter 1

Problem statement and related work

In this chapter we introduce the reader to the problematics of genomes comparison. We discuss biological relevance of this problem and it's theoretical and technical challenges. We finish with an overview of related work.

1.1 Biological motivation

Deoxyribonucleic acid (**DNA**) is a molecule that caries genetic information of a cell, determining it's growth, development, functioning and reproduction. DNA molecules are organized in two facing strands which are composed of simpler monomer units called nucleotides. Each nucleotide is composed of one of four nitrogen-containing nucleobases—either cytosine (C), guanine (G), adenine (A), or thymine (T). The bases of the two separate strands are bound together according to base pairing rules (A with T, and C with G) to form double-stranded DNA.

Natural representation of a genome is a sequence/string of letters A, C, T, G corresponding to four bases. Note that thanks to the pairing rule we can store the genome as only the sequence for a single strand. The usual length of these strings is between 10^6 and 10^{10} of characters.

It is convenient to tell a difference between multiple genome sequences. We can ask about simple characteristics like number of C bases and compare those characteristics across multiple genomes. Moreover we can ask about specific small sequences that were changed. Note figure X where we can see an absence of a sequence <placeholder> in <placeholder> genome.

TODO real world example REDO!!!

These specific differences are called variants. Variants are responsible for

differences in individuals. If we know the difference between two individuals, we can determine the variant that is responsible for this difference. Vice versa by detecting a variant in two genomes we, can determine otherwise undetectable difference. So if we know a variant between cell with cancer and healthy cell we can try to find this variant in different cells and determine, if they started to mutate dangerously.

1.2 Genome sequencing

Determining the genome sequence for a physical DNA is done by DNA sequencing. Currently most used sequencing techniques cannot read the whole sequence at once, instead they produce set of short, overlapping substrings of sequence called reads. These techniques also provide a measure of read quality, that is how certain we could be about a given base. It is assumed that locations of these substrings are sampled uniformly across the original string. Collection of reads for a given genome is called library.

Other important form of sequencing data are paired reads. Paired reads are two reads, for which we know their approximate distance in the genome. The total reads length plus the approximated distance is referred to as insert size.

The most important read characteristics are number of errors and length of reads. Various read characteristics are determined by used DNA sequencing method. Note that there is a lot of other attributes that are very important for practical sequencing. These attributes are throughput, reads, runtime, instrument cost, resources needed for data post-processing. They explain some surprisingly high or low costs of some methods.

As current sequencing techniques are rapidly improving, we did a basic overview of currently most popular ones. We only considered their attributes relevant for our problem. Our data are based on overview from year 2012 by Quail[14] and Liu[10], and on overview from year 2016 by Goodwin [6]. The first one is important even if it is outdated, because we do not necessary have data sampled with the latest methods.

This shows, we are going to encounter $150b$ long paired reads with an error rate at around 1%.

Table 1.1: Overview of sequence technologies for year 2012

Technology	Read length (bp)	Paired reads	Error rate (%)	Cost per million bases ($)
454 GS FLX Titanium XLR70	700	Yes	0.1	10
Illumina HiSeq 2000	150	Yes	0.8	502
Illumina GAIIx	150	Yes	0.76	148
Illumina MiSeq	1500	Yes	12.86	2000
Ion Torrent PGM	200	Yes	1.71	1000
PacBio RS	1500	No	12.86	2000
Sanger 3730xl	400-900	No	0.001	2400
SOLiDv4	50	Yes	0.04	0.13
Oxford Nanopore MK 1 MinION	200000	No	12	750

1.3 Genome alignment and assembly

One common approach to variant detection is to reconstruct the sequence based on reads and then compare reconstructed sequences. In some cases, one sequence is already known and it is called reference sequence. Methods, that do not rely on the reference sequence are called de novo.

The sequence reconstruction problem is referred to as sequence assembly. Common formulation for this problem is as a shortest common superstring problem.

Definition 1.3.1 (Shortest common supersting) *Given set of strings $\mathcal{P} = S_1, S_2, ..., S_k$, the shortest common superstring is a shortest string S, that contains every string from $\mathcal{P}$ as a substring.*

This problem proves to be NP-hard [4].

Other common formulation is to find string by maximal likelihood.

Definition 1.3.2 (Maximum likelihood supersting) *Given set of strings $\mathcal{P} = S_1, S_2, ..., S_k$, the Maximum likelihood supersting is a string S, that maximize probability of observing $\mathcal{P}$ as a substring.*

These definitions could be extended to work with paired reads as well.

If we could reliably find the original DNA sequence, we could find variants by aligning those sequences.

Informally, sequence alignment is a way of arranging the sequences to identify regions of similarity. Aligned sequences of are typically represented as

rows within a matrix. Gaps are inserted between the letters so that identical or similar characters are aligned in successive columns.

TODO example variant aligning

Reconstructing the original sequence as defined above is often very hard. Moreover most of the times the sequence contains a lot of errors, ambiguities and regions of low reliability. Sequences also contain so called repeats, sequences that are repeated multiple times, which are hard to correctly assembly, but could be very important variants.

In practice there are even more problems like reverse complement, diploid genomes, errors in reads, paired reads. TODO CITE and REF These can make the retrieved sequence even more ambiguous.

Most genome assembly algorithms focus only on the most probable assembly. However, if want to find the most probable variant, we must consider all possible assemblies. This is not feasible in an context of standard assembly and alignment.

1.4 Problem statement

TODO variant specification/definition. This will be added after we will see, what specific variants we can detect and describe.

TODO formal definition. We are not sure if we want to define variants as aWb - aXb 1.5.2 or something different.

As seen in chapter above, for the purpose of variant detection it might be better to get rid of the assembly and alignment steps.

For this problem, some work has been already done 1.5. However proposed methods have a lot of shortcomings. They lack concrete probabilistic foundation, do not incorporate paired reads or do not consider combinations of reads from different sources.

We would like to overcome as much of thees shortcomings and present de novo probabilistically grounded method for variant calling using paired reads and multiple sequencing technologies.

1.5 Related work

1.5.1 Bidirected de Bruijn graph

A common method to represent reads is a de Bruin graph. This structure serves as a foundation for many assembly and alignment algorithms. (TODO ref)

It's mutations (TODO ref) are used for variant callings as well.

A de Bruijn graph [3] is a structure for representing overlaps between strings.

Definition 1.5.1 (de Bruijn graph) *A de Bruijn graph is directed graph. For given k, vertices of De Bruijn graph represent sequences of length k (k-mers). The edges of de Bruijn graph represent possible overlaps of length $k-1$, i. e. if have have an edge (u, v) and vertex u represents k-mer $a_1, a_2, \ldots, a_k$, then vertex v represents k-mer $a_2, a_3, \ldots, a_k, x$ and k-mer in vertex v can follow k-mer in vertex u.*

Intuitively an edge (u, v) in de Bruijn graph represents, that k-mer u is followed by k-mer v in the DNA sequence.

Constructing this graph for set of reads is easy. For each read from $S_x \in \mathcal{P}$ we construct a set of all its $k+1$-mers. Each of this $k+1$-mers consist of two overlapping k-mers k_1, k_2, hence we can add new vertices k_1, k_2 with an edge (k_1, k_2) to the da Bruijn graph. Note that each edge can be in a graph multiple times. This is called coverage, and it represents the number of observations that k_1 if followed by k_2.

De Bruijn graphs are widely used in genome assembly and underlie many popular algorithms, including AllPaths-LG, SOAPdenovo, Abyss and Velvet. (TODO refs) This approach can be extended for comparing two libraries. We create one de Bruijn graph out of both of them, but each vertex and each edge will be colored based on the library they are from.

If both libraries are the same, than each node and each vertex has both colors.

Definition 1.5.2 (variant) *Let a, w, w^*, b, A, B be strings over alphabet Σ. Strings $a.w.b$ and $a.w^*.b$ are variants of sequences A and B, if A contains $a.w.b$ and B contains $a.w^*.b$.*

If the initial sequences A and B contain variants, those variants will show as bubbles[8] in the resulting de Bruijn graph.

(TODO draw better pictures)

There is a number of ways to analyze these bubbles. (a) The simplest $a.w.b$ and $a.w^*.b$ variant is showed as two parts that diverge and merge together forming a bubble. (b) Repeats also form bubbles, but note that these bubbles have both colors. (c) When the graph does not form a clean bubble, we can identify variant sites by tracking the divergence of paths. On finding a breakpoint, we take the longest contig in the sample (that is, the path as far

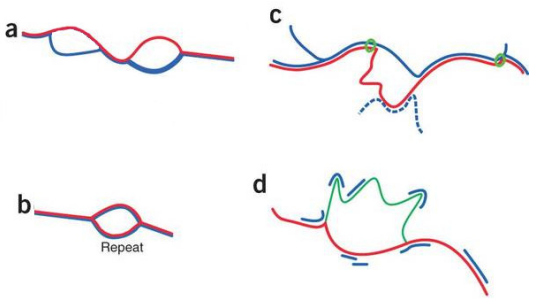

Figure 1.1: Schematic representation of methods for variation analysis using colored de Bruijn graphs. Coverage is represented by line width.

as the next junction) and ask whether the red path returns before this point (green circles show the anchoring sequence). (d) The likelihood of any given genotype can be calculated from the coverage (blue) of each allele (green, red), accounting for contributions from other parts of the genome. In this example, the sample is heterozygous and therefore has coverage of both alleles, although not sufficient to enable full assembly.

Unfortunately, these approaches do not provide concrete probabilistic justification for used methods. Moreover they do not use information about paired reads, which proved to be valuable to decide complicated de Bruijn structures. (TODO cite)

(TODO ref cite Computability of Models for Sequence Assembly)

1.5.2 Bubble calling

As each variant corresponds to a bubble, we just need to find all bubbles.

Definition 1.5.3 (**(s,t) -path**) *Given two vertices s and t in G, an (s,t)-path is a path from s to t.*

Definition 1.5.4 (**(s,t) -bubble**) *is set of two vertex-disjoint (s,t)-paths.*

Definition 1.5.5 (**(s,t,α_1,α_2)-bubble**) *in a weighted directed graph is a (s, t)-bubble with paths p_1 , p_2 satisfying $\|p_1\| \leq \alpha_1$ and $\|p_2\| \leq \alpha_2$.*

Definition 1.5.6 (**$(s,t,\mathcal{A}) - d$-bubble**) *Let d be a natural number and $A = \{\alpha_1, \cdot, \alpha_d\} \subset Q_{\geq 0}$. Given a directed weighted graph G and two vertices s and t, a $(s,t,\mathcal{A}) - d$ - bubble is a set of d pairwise internally vertex-disjoint paths $\{p_1, \ldots, p_d\}$, satisfying $\|p_i\| \leq \alpha_i \forall i \in [1,d]$.*

For finding variants we need to solve problem of having a $(s, t, \alpha_1, \alpha_2)$–bubble. Given a non-negatively weighted directed graph G, a vertex s, a set $A = \{\alpha_1, \ldots, \alpha_d\} \subset (Q)_{\geq 0}$ and $d \in \mathcal{N}$, decide if there exists a $(s, t, \mathcal{A}) - d$-bubble in G , for some $t \in V$. This problem is a generalization of the two disjoint paths problem with a min–max objective function, which is NP-complete. (TODO The complexity of finding two disjoint paths with min–max objective function)

The same goes for problem of enumerating all bubbles in a graph. Sacomoto et al. [16] showed the first polynomial delay algorithm to enumerate all bubbles with length constraints in a weighted directed graph. Complexity of this algorithm in the best theoretical case for general graphs is O(n(m + n log n)), hence an O(n(m + n log n)) delay algorithm, where n is the number of vertices in the graph, m the number of edges. In the particular case of de Bruijn graphs, the complexity is $O(n(m + n \log \alpha))$ where α is a constant related to the length of the skipped part in an alternative splicing event. In practice, an algorithmic solution in $O(nm \log n)$ appears to work better on de Bruijn graphs built from such data. Sacamoto implemented the latter, and showed that it is more efficient than previous approaches and outlined that it allows to discover novel long alternative splicing events.

1.5.3 Variant Calling

Heng Li et al [9] introduced quality score for mapping short reads to the reference genome.

They then extended this approach for calling variants between the library and a reference genome and implemented it as an MAQ algorithm.

Heng Li's work was based on finding the best read alignments to the reference sequence. He discussed use of Eland-like (A.J. Cox, unpubl.) hashing technique and it's probabilistic interpretation. Note that this was a probabilistic interpretation of the hashing, not of sequencing. This hashing is used to find the 28*bp* long beginning (seed) of the read in the reference sequence with some number of acceptable errors.

MAQ then assigns each individual alignment a mapping quality. The mapping quality Q_s is the scaled probability (TODO Ewing and Green 1998) that a read alignment may be wrong.

$$Q_s = -10 \log_{10} \Pr(\textit{read is wrongly mapped})$$

For example, $Q_s = 30$ implies there is a 1 in 1000 probability that the read is incorrectly mapped.

Heng Li considers a simplistic case where all reads are known to come from the reference and an ungapped exhaustive alignment is performed. Practical model for alignment with heuristic algorithms is also considered.

They computed the probability $\Pr(z|x,u)$ of read sequence z coming from the position u at the reference sequence x, on the assumption that sequencing errors are independent at different sites of the read.

To calculate the posterior probability $\Pr_s(u|x,z)$, Heng assumes an uniform prior distribution $p(u|x)$, and applying the Bayesian formula gives $P_s(u|x,z) = \frac{P(z|x,u}{\sum_{v=1}^{L-l+1} P(z|x,v)}$ where $L = \|x\|$ is the length of x and $l = \|z\|$.

In practice, Heng approximates Q_s as

$$Q_s = \min\{q_2 - q_1 - 4.343\log(n_2), 4 + (3-k^*)(\hat{q}-14) - 4.343\log(P_1(3-k^*,28)\}$$

Where q_1 is the sum of quality values of mismatches of the best read alignment, q_2 is the corresponding sum for the second best alignment, n_2 is the number of alignments having the same number of mismatches as the second best alignment, k is the minimum number of mismatches in the $28-bp$ seed, q is the average base quality in the $28-bp$ seed, 4.343 is $\frac{10}{log(10)}$, and $P_1(k,28)$ is the probability that a perfect alignment and a k-mismatch alignment coexists given a $28-bp$ sequence that can be estimated during alignment.

1.5.4 Probabilistic model for sequence assembly

As our problem is closely related to modeling the initial genome, we will base our theory on this modeling.

Several probabilistic models were introduced[15][5] as a measure of the assembly quality. In our work we will build on those, as there is an clear parallel between assembly quality and variant quality. As authors have shown, the likelihood based metrics consistently favors higher quality assemblies, so they should favor higher quality variants as well.

The probabilistic model defines the probability $\Pr(L|A)$ that a library of sequencing reads L is observed assuming that assembly A is the correct assembly of the genome.

This model only evaluates quality of a single assembly. Boža[2] extended this model and used it as an optimization criterion in the own process of finding high likelihood genome assemblies.

The model assumes that individual reads are independently sampled, and thus the overall likelihood is the product of likelihoods of the reads:

$$\Pr(R|A) = \prod_{r \in R} \Pr(r|A)$$

Boža used the log-average probability of a read, what made the resulting value independent of the number of reads in set R.

$$\mathrm{LAP}(A|R) = \frac{1}{|R|}\sum\nolimits_{r\in R} \log \Pr(r|A)$$

Note that maximizing $\mathrm{LAP}(A|R)$ is equivalent to maximizing $\Pr(R|A)$.

If the reads were error-free and each position in the genome was sequenced equally likely, the probability of observing read r would simply be $\Pr(r|A) = \frac{n_r}{2L}$, where n_r is the number of occurrences of the read as a substring of the assembly A, L is the length of A, and thus $2L$ is the length of the two strands combined.

The probability of a single alignment with s errors (substitutions and indels) and m matching positions is defined as $\frac{R(s,m)}{2L}$, where $R(s,m) = \varepsilon^s(1-\varepsilon)^m$ and ε is the sequencing error rate.

Boža noted, that the full dynamic programming is too time consuming, and in practice only several best alignments contribute significantly to the overall probability.

They approximate the probability of observing read r with an estimate based on a set S_r of a few best alignments of r to genome A, as obtained by one of the standard fast read alignment tools:

$$\Pr(r|A) \approx \frac{\sum_{j\in S_r} R(s_j, m_j)}{2L}$$

s_j is the number of mismatches and indels implied by the j^{th} alignment alignment and m_j is the number of matches in this alignment.

Boža also incorporated paired reads. They assumed that the insert size distribution in a set of reads R can be modeled by the normal distribution with known mean μ and standard deviation σ. The probability of observing paired reads r_1 and r_2 can be estimated from the sets of alignments (S_{r_1}) and (S_{r_2}) as:

$$\Pr(r_1, r_2|A) \approx \frac{1}{2L}\sum_{j_1\in S_{r_1}}\sum_{j_2\in S_{r_2}} R(s_{j_1}, m_{j_1})R(s_{j_2}, m_{j_2}) \Pr(d(j_1, j_2)|\mu, \sigma)$$

$(d(j_1, j_2))$ is the distance between the two alignments as observed in the assembly and (s_{j_i}) and (m_{j_i}) are numbers of sequencing errors and matches in alignment j_i.

To model a case where reads are from different libraries obtained by different technologies, they used different model parameters for each set and compute the final score as a weighted combination of log average probabilities for individual read sets $R_1, \ldots, R_k$:

$$\mathrm{LAP}(A|R_1, \ldots, R_k) = w_1\mathrm{LAP}(A|R_1) + \cdots + w_k\mathrm{LAP}(A|R_k).$$

1.5.5 Indexing techniques

A lot of times in genome sequencing, assembly, or aligning we need to search in a set of reads. As discussed at section 1.5.3, we needed to align multiple reads to the reference sequence. As we need to tolerate small mismatches, this problem becomes hard. Various methods based on Blum filters and hashing were proposed (TODO cite).

In fact this problem appears in other fields.

Definition 1.5.7 *Given a set S of d dimensional real value vectors $d_1, d_2 \dots d_n$, retrieve set of k most similar vector query vectors $q_1, q_2, \dots, g_m$.*

Similarity is very vaguely defined, because we are often interested in different similarities in different spaces like L_1, Levenshtein, euclidean or even non-metric ones like KL-divergence, cosine distance or Itakura-Saito.

This problem is called K-nearest neighbor search problem (K-NNS). In its approximated version (K-ANNS) we allow some false positive results.

Authors [12, 13] proposed a proximity graph K-ANNS algorithm called Navigable Small World (NSW), which utilized navigable graphs with long range links constructed by a much simpler model.

Malkov[11] introduced and implemented[1] an algorithm for the K-ANNS based on navigable small world graphs with controllable hierarchy (Hierarchical NSW).

Hierarchical NSW incrementally builds multi-layer structure consisting from hierarchical set of proximity graphs for nested subsets of the stored elements. The maximum layer in which an element is present is selected randomly with exponentially decaying probability distribution. Starting search from the upper layer together with utilizing the scale separation boosts the performance compared and allows a logarithmic complexity scaling.

Additional employment of a heuristic for selecting proximity graph neighbors significantly increases performance at high recall and in case of highly clustered data. Performance evaluation has demonstrated that the proposed general metric space method is able to strongly outperform many previous state-of-art vector-only approaches.

Malkov's[11] algorithm based on idea s close to NSW, very good logarithmic complexity scaling. The main contributions were smart selection of the graph's enter-point node, separation of links by different scales and using a slightly more complicated heuristic to select the neighbors.

1.5.6 A*

Our algorithm will be probably based on some form of de Bruijn graph traversal. Paired reads and long reads provide information about real distance between two nodes in de Bruijn graph.

Incorporating this information in process of graph traversal can be hard. We took an inspiration from a well known A^* algorithm.

Chapter 2

Proposed probabilistic approach 5 pages

To continue we need a few definitions. Let v, w be two strings over alphabet $\Sigma = \{A, C, G, T\}$. Concatenation of these strings is denoted as $v.w$.

We will extend the de Bruijn definition. TODO extended de Bruijn graph definition

Chapter 3

Catchy name for our tool that will be implemented

3.1 Implementation problems 8

3.2 Data 1

write about data generation.

3.2.1 Variant simulation 2-3

Here we closely describe methods used for variant generation, and relevant variant parameters.

3.2.2 Read simulation 2-3

Here we evaluate data for reads and we set our expectations about these data.

3.2.3 Paired read simulation 2-3

write about data sources Here we evaluate data for paired reads and we set our expectations about these data.

3.2.4 Real data 2-3

3.3 Metrics 2-3

Here we discuss what metrics we want to improve and what is the measure of success for our solution.

3.4 Benchmarks

3.4.1 Genome alignments 3-4

Brief mentions of current genome alignments approaches.

3.4.2 Without paired reads 2

3.4.3 With paired reads 2

3.5 Results 12

3.6 Future improvements 5

Chapter 4

Discussion 2

Here we will discuss.

Bibliography

[1] Yury Malkov Bilegsaikhan Naidan, Leonid Boytsov. Non-metric space library (nmslib). `https://github.com/searchivarius/nmslib`, 2016.

[2] Vladimír Boža, Broňa Brejová, and Tomáš Vinař. Gaml: genome assembly by maximum likelihood. *Algorithms for Molecular Biology*, 10(1):1, 2015.

[3] Nicolaas Govert De Bruijn. A combinatorial problem. 1946.

[4] John Gallant, David Maier, and James Astorer. On finding minimal length superstrings. *Journal of Computer and System Sciences*, 20(1):50–58, 1980.

[5] Mohammadreza Ghodsi, Christopher M Hill, Irina Astrovskaya, Henry Lin, Dan D Sommer, Sergey Koren, and Mihai Pop. De novo likelihood-based measures for comparing genome assemblies. *BMC research notes*, 6(1):1, 2013.

[6] Sara Goodwin, John D McPherson, and W Richard McCombie. Coming of age: ten years of next-generation sequencing technologies. *Nature Reviews Genetics*, 17(6):333–351, 2016.

[7] Weichun Huang, Leping Li, Jason R Myers, and Gabor T Marth. Art: a next-generation sequencing read simulator. *Bioinformatics*, 28(4):593–594, 2012.

[8] Zamin Iqbal, Mario Caccamo, Isaac Turner, Paul Flicek, and Gil McVean. De novo assembly and genotyping of variants using colored de bruijn graphs. *Nature genetics*, 44(2):226–232, 2012.

[9] Heng Li, Jue Ruan, and Richard Durbin. Mapping short dna sequencing reads and calling variants using mapping quality scores. *Genome research*, 18(11):1851–1858, 2008.

[10] Lin Liu, Yinhu Li, Siliang Li, Ni Hu, Yimin He, Ray Pong, Danni Lin, Lihua Lu, and Maggie Law. Comparison of next-generation sequencing systems. *BioMed Research International*, 2012, 2012.

[11] Yu A Malkov and DA Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *arXiv preprint arXiv:1603.09320*, 2016.

[12] Yury Malkov, Alexander Ponomarenko, Andrey Logvinov, and Vladimir Krylov. Scalable distributed algorithm for approximate nearest neighbor search problem in high dimensional general metric spaces. In *International Conference on Similarity Search and Applications*, pages 132–147. Springer, 2012.

[13] Yury Malkov, Alexander Ponomarenko, Andrey Logvinov, and Vladimir Krylov. Approximate nearest neighbor algorithm based on navigable small world graphs. *Information Systems*, 45:61–68, 2014.

[14] Michael A Quail, Miriam Smith, Paul Coupland, Thomas D Otto, Simon R Harris, Thomas R Connor, Anna Bertoni, Harold P Swerdlow, and Yong Gu. A tale of three next generation sequencing platforms: comparison of ion torrent, pacific biosciences and illumina miseq sequencers. *BMC genomics*, 13(1):1, 2012.

[15] Atif Rahman and Lior Pachter. Cgal: computing genome assembly likelihoods. *Genome biology*, 14(1):1, 2013.

[16] Gustavo Sacomoto, Vincent Lacroix, and Marie-France Sagot. A polynomial delay algorithm for the enumeration of bubbles with length constraints in directed graphs and its application to the detection of alternative splicing in rna-seq data. In *International Workshop on Algorithms in Bioinformatics*, pages 99–111. Springer, 2013.

Appendix A

TODO