Proposal for diploma thesis

# Accurate Long-Read Alignment using Similarity Based Multiple Pattern Alignment and Prefix Tree Indexing

Astrid Rheinländer

01-09-2010

Supervisor: Prof. Dr. Ulf Leser

## Motivation

Sequence analysis is one of the main challenges in current bioinformatics. Sequence analyses are used for finding structural, functional or evolutionary relationships between two sequences, to identify highly conserved regions in a genome or to align new sequencing data against an existing reference genome [PS08].

The ongoing research in sequencing technology has yielded in machines that are able to produce sequence data in the order of one billion base-pairs (bp) per machine day with an average read length of less than 100 bp per read ("short-reads"). In the past two years, many efficient algorithms have been developed for short-read alignment against a reference genome and for genome assembly, for an overview of and comparison of these tools see [LH10].

However, as the sequencing technology is evolving, the produced read lengths become much longer ("long-reads"). Li and Durbin [LD10] report that the Roche/454 sequencing technology already produces reads of lengths greater than 400 bp and that Pacific Bioscience experiments with read lengths longer than 1000 bp. Therefore, long-read alignment will become a crucial task in the near future.

The existing short-read alignment tools will not be applicable to the alignment of long-reads for several reasons. First of all, long-reads have a bigger chance of containing more errors due to their lengths. Furthermore, Li and Durbin state that tools for long-read alignment should be permissive about alignment gaps because insertions / deletions (indels) will occur more frequently in long-reads and might be the main source of sequencing errors. Therefore, it is essential to have an alignment software that is able to deal with variations and alignment gaps. Most of the short-read alignment tools are only efficient for un-gapped alignment or an alignment with a very small number of gaps. When these tools are used with more allowed errors and gaps, their performance quickly degrades. Accordingly, new strategies for an efficient alignment of long-reads need to be developed.

## Goal

In this diploma thesis, a tool for long-read alignment to a reference genome will be developed and evaluated. We aim at designing a tool that produces accurate alignment results and that is fast. On the basis of an existing implementation of a compressed prefix tree index [RKH+10] we will explore two strategies for long-read alignment. The first strategy is based on an Aho-Corasick automaton [AC75] that is also capable to deal with similarity based searches. This automaton can be used for indexing a set of long-reads and align it to a reference genome with semi-global alignment, which is essentially a global alignment [NW70] that ignores gaps at the start and the end. The second strategy is based on indexing both, the reference sequence and the long-reads in PETER-indices and it computes conceptually the similarity-based intersection of both indices. As we do not follow the seed-and-extend paradigm but rather compute the best alignment for all long-reads directly, we need efficient pruning strategies as well as a parallelized implementation to speed up our algorithms.

## Methods

In a first step, we will use the current implementation of a prefix-tree index for approximate searches (PETER, [RKH+10]) as a baseline for our work. We will improve PETER with selected suggestions, including the following points:

(a) We will remove the internal / external suffixes distinction. All suffixes will be stored directly in the index tree and not, as in PETER, in an external suffix file. This allows us to keep the full information on each long-read in the index in main memory and thus it reduces the necessary number of disk accessions.

(b) As it was shown in [RKH+10], the comparison of q-gram sets of two strings is very helpful in terms of reducing query response time for edit distance [L66] queries, even if the q-grams are computed and evaluated on the fly. We believe that the persistent integration of q-grams will further reduce the query response time. Therefore, we will introduce an additional bitset attribute to each node that stores q-gram information for the prefix represented by this node.

(c) PETER can only deal with strings over an alphabet of 4 characters, internally represented by 2 bits. The character representation will be extended to 8 bits, which allows us to index all strings from the ISO 8859 family of character encodings, e.g. German/English text or proteins.

In the next step, we will explore which one of the following strategies is better suited for fast and accurate long-read alignment.

The first strategy uses an Aho-Corasick algorithm [AC75] as the basic strategy for similarity-based sequence alignment. Long-reads will be stored in a prefix tree index, which is enriched with failure and output links. As [AC75] was natively designed for exact searches, we need to develop functionality for edit distance based similarity searches. This requires on the one hand the costly computation of a distance matrix and on the other hand failure links that are capable to deal with replacements or indels. For exact failure links (these are failure links where one substring from a node x is an exact prefix of some other string), a node x in the compressed prefix tree has at most k outgoing failure links when k is the length of the substring represented by x. For similarity-based searches, the similarity-based construction of failure links will be a critical. Even if we only allow replacements and no indels in the failure links, each node can contain up to $\sum_{i=1}^{k} \Sigma^{i}$ links. This is obviously too much overhead and therefore we will explore whether we can apply some heuristics that reduce the number of similarity based failure links.

The second strategy is based on indexing both, the long-reads and the reference genome in separate PETER indices. For the reference genome, we will index the reverse of the reference in our prefix tree, which essentially gives us the suffix tree of the reference. As this index will require lots of memory, we need to develop strategies for decreasing the index size. This includes a proper partitioning of the reference sequence, e. g., by chromosomes, and we will investigate whether the following strategies reduce the index size:

(a) Two constants, a maximum alignment error rate *e* and a maximum long-read length *l* will be used for downsizing the genome index only. Given the fact that any two strings can only be within edit distance k, if their lengths differ in at most *k* characters, we can now trim down many indexed suffixes. If one suffix starts at position *i* in the sequence *s* of length *n*, we do not need to store *s[i … n]* in the index, it is sufficient to store only the substring *s[i … (i+l+l*e)]* in the index.

(b) As there are often unknown bases, indicated by the character 'N', contained in the reference sequence, we will interpret all occurrences of 'N' as a mismatch. Thus, we will index only those substring of s that contain at most *l*e* times the character 'N'.

When we have indexed both, the reference genome and the long-reads, we will traverse both trees simultaneously and conceptionally compute the intersection of both trees. Given that we have reached a string node in the long-read tree and the current edit distance does not exceed the defined threshold, we can instantly report a match at any position in the reference index.

After having evaluated the pros and cons of each strategy, we will decide for one and implement the necessary algorithms. Plus, we will add the following additional features:

(a) In order to increase the usability of our tool and to compare it to other tools, we will add a functionality to index strings from the FASTA [NCBI] / FASTQ [CFG+10] file formats. The computed alignment will be given in the SAM / BAM output format [HWT+09] that allows us to easily view and check the computed alignment with the help of existing alignment visualization tools, such as [CBO+10].

(b) Prefix pruning: When traversing down the index tree, we will keep track of the current edit distance. Whenever the current distance exceeds our threshold for a given prefix, we will immediately stop the edit distance computation as we will not find a suitable alignment. Thus, a backtracking routine starts and goes on to examine the next parts of the index tree.

(c) Dynamic threshold adjusting: As described in [LD10], long-reads have a great potential of large error rates (Li and Durbin ran experiments with an error rate of up to 10 %). If such a rate is expressed in absolute values, we have to consider the overall length of each long-read . When traversing the index tree, we know at each node x the length of the longest read that occurs somewhere below x. We will use this information to adjust the bandwidth in the k-banded edit-distance alignment algorithm [F84] and to define an edit distance threshold. When traversing to a child y of x, we possibly find that the longest read contained in the sub-tree starting at y is shorter than the longest read of x. This happens, whenever another child z of x holds this longest string of x. In this case, when we investigate the sub-tree y, we can narrow down the bandwidth of the k-banded alignment algorithm and also decrease the allowed edit distance threshold.

(d) An additional lookup table stores the current best alignment and its score. Whenever a better alignment for a pattern is found, we perform an update on this lookup table. Finally, we will report the best alignment for each long-read in the SAM/BAM output format.

In a third phase, we will parallelize our alignment algorithm with map/reduce [DG08]. Therefore, we will partition the long-read set into smaller portions and then index these subsets in several prefix tree indices. Figure 1 shows the main idea for parallelization. In the map phase, each map thread aligns the strings from one long-read index to one chromosome index with our alignment algorithm. Intermediate results, the alignment of each long-read with the best score, are stored in o intermediate files, where o is the number of CPU kernels on our evaluation platform. Each intermediate file will store the alignments for a certain range of long-read IDs (equally spread over the o files). When all map threads have finished, i.e. all long-read indices have been aligned to each chromosome index, the reduce phase starts. In this phase, each reduce thread sorts one intermediate result by long-read ID and by alignment score. If we have more than one alignment for a long-read, we will only keep the the one with the best score. Finally, the alignments will be sorted by chromosome and then SAM/BAM files will be produced as output.

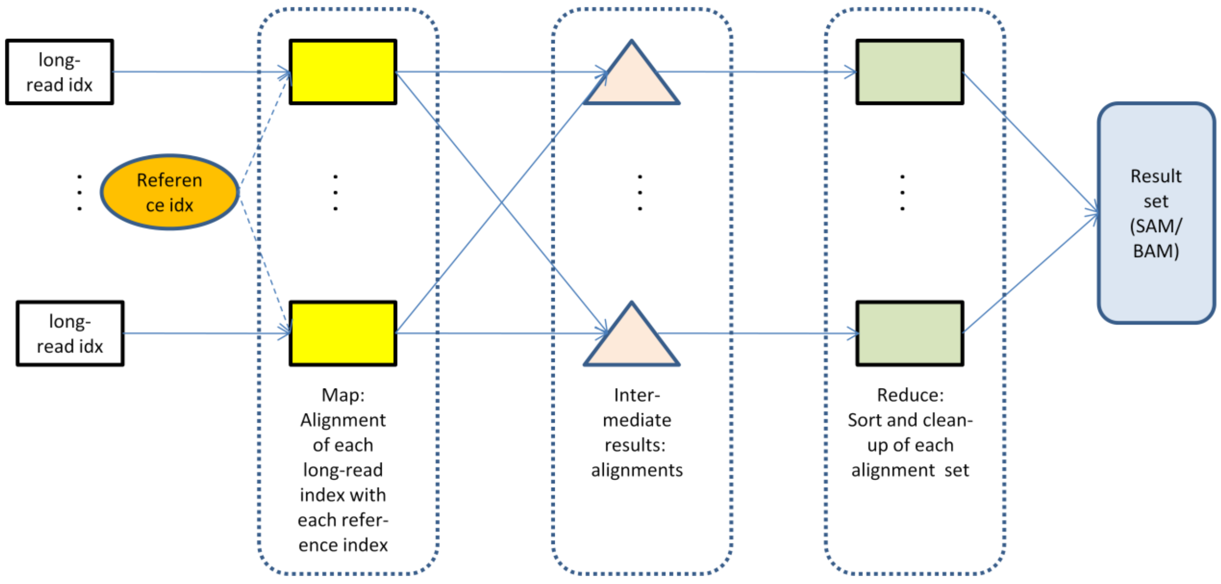

**Figure 1:** Parallelization of the alignment algorithm with map/reduce.

Finally, we will evaluate the accuracy of our software mainly on the basis of simulated long-read data sets which are aligned back to the given reference genome (similar to the evaluation process of [LD10, LD09]). With this procedure, we know the exact coordinates of each read and thus are able to evaluate the number of alignment errors of our tool. Errors might occur when some long-read has more than one best alignment (e.g. the same alignment score at more than one position in the reference genome). We also aim at evaluating our software on real data sets, given that such data will be available. In terms of alignment speed, we will compare our tool to already existing alignment tools such as BLAT [K02] and BWT-SW [LD10].

## References

[AC75] A.V. Aho und M.J. Corasick, "Efficient string matching: an aid to bibliographic search," Communications of the ACM, vol. 18, 1975, p. 333-340.

[CBO+10] T. Carver, U. Bohme, T.D. Otto, J. Parkhill, und M. Berriman, "BamView: viewing mapped read alignment data in the context of the reference sequence," Bioinformatics, vol. 26, March 2010, p. 676-677.

[CFG+10] P.J.A. Cock, C.J. Fields, N. Goto, M.L. Heuer, und P.M. Rice, "The Sanger FASTQ file format for sequences with quality scores, and the Solexa/Illumina FASTQ variants," Nucleic Acids Research, vol. 38, Apr. 2010, p. 1767-1771.

[DG08] J. Dean und S. Ghemawat, "MapReduce: simplified data processing on large clusters," Communications of the ACM, vol. 51, 2008, p. 107-113.

[F84] J.W. Fickett, "Fast optimal alignment," Nucleic Acids Research, vol. 12, Jan. 1984, S. 175-179.

[HWT+09] H. Li, B. Handsaker, A. Wysoker, T. Fennell, J. Ruan, N. Homer, G. Marth, G. Abecasis, und R. Durbin, "The Sequence Alignment/Map format and SAMtools," Bioinformatics (Oxford, England), vol. 25, Aug. 2009, p. 2078-2079.

[K02] W.J. Kent, “BLAT—The BLAST-Like Alignment Tool,” Genome Research, vol. 12, Apr. 2002, p. 656 -664.

[L66] V. Levenshtein, “Binary codes capable of correcting deletions, insertions, and reversals,” Soviet Physics Doklady, vol. 10, 1966, S. 710, 707.

[LD09] H. Li und R. Durbin, “Fast and accurate short read alignment with Burrows-Wheeler transform,” Bioinformatics, vol. 25, Juli. 2009, p. 1754-1760.

[LD10] H. Li and R. Durbin, “Fast and accurate long-read alignment with Burrows-Wheeler transform,” Bioinformatics, Vol. 26, March. 2010, p. 589-595.

[LH10] H. Li and N. Homer, “A survey of sequence alignment algorithms for next-generation sequencing,” Briefings in Bioinformatics Advance Access, May 2010.

[NCBI] National Center for Biotechnology Information, “FASTA format description”. [Online], Available: http://www.ncbi.nlm.nih.gov/blast/fasta.shtml, last access at 29-07-2010.

[NW70] S.B. Needleman und C.D. Wunsch, “A general method applicable to the search for similarities in the amino acid sequence of two proteins,” Journal of Molecular Biology, vol. 48, March 1970, p. 443-453.

[PS08] M. Pop and S.L. Salzberg, “Bioinformatics challenges of new sequencing technology,” Trends in Genetics, Vol. 24, March. 2008, p. 142-149.

[RKH+10] A. Rheinländer, M. Knobloch, N. Hochmuth, and U. Leser, “Prefix Tree Indexing for Similarity Search and Similarity Joins on Genomic Data,” Scientific and Statistical Database Management, M. Gertz und B. Ludäscher (eds.), LNCS 6187, 2010, p. 519-536.