High throughput sequence alignment with suffix arrays and q-grams

Justin Wilson, Manhong Dai, Stanley Watson and Fan Meng

Molecular and Behavioral Neuroscience Institute and Department of Psychiatry
University of Michigan
mengf@umich.edu

Abstract

High throughput sequencing is without a doubt one of the most influential technological advances in biomedical research. Analyzing the large volume of sequence data generated by high throughput sequencing is a major challenge. Current solutions may miss up to 30% of the unique matches and are not suitable for longer high throughput sequences.
In response, we developed AQUESA (Aligning Q-grams Using Ehanced Suffix Arrays); a sequence alignment algorithm designed for aligning high throughput sequencing results under the assumptions of variable length sequences, total recall and multiple single base mismatches, single base insertions and single base deletions. We unite the "sort and search" approach of enhanced suffix arrays with q-grams and extend perfect seed matches to a specified mismatch tolerance. Between the suffix array construction and alignment, valid q-grams in the reference and query are indexed so that q-grams in the reference overlap and q-grams in the query do not. These inclusion vectors permit the re-use of suffix arrays and can themselves be reused for future alignments. Our method retains the advantages of suffix arrays like favorable memory requirements and time complexities (ie. linear or near-linear) at the expense of disk space for indices. We are currently comparing our implementation in terms of recall, time and machine resources to algorithms such as BLAST, BLAT, MEGABLAST, MOSAIK, Eland, MOSAIK, SOAP and Vmatch.

1. Introduction

Q-gram based sequence alignment is based on the principle that in order for an alignment to exist, then at least q contiguous bases must match perfectly. If L is the length of a query sequence and M is the maximum number of allowed mismatch events (insertion, deletion or mismatch), then q is calculated as $q = \left\lfloor \frac{L}{M+1} \right\rfloor$. The query sequence is divided into $M+1$ segments that are at least q bases long called q-grams. If the maximum number of mismatch events M are distributed among the $M+1$ q-grams, then in all cases at least one q-gram will not have a mismatch event. Sequence alignment is then a matter of finding q-grams in the reference and extending them to check for an alignment.
Suffix arrays are data structures designed as storage efficient replacements for suffix trees [1]. A suffix array (SA) is a permutation of the indices of a string so that $\mathrm{S}[\mathrm{SA}[i]] < S[\mathrm{SA}[i+1]]$ where $\mathrm{S}[j]$ denotes the substring of S starting at j. Another data structure related to the SA is the inverse suffix array (ISA) defined as $\mathrm{ISA}[\mathrm{SA}[i]] = i$. The ISA provides the rank of a substring given its starting index. The longest common prefix (LCP) array is defined as $\mathrm{LCP}[i] = lcp(\mathrm{S}[\mathrm{SA}[i-1]], \mathrm{S}[\mathrm{SA}[i]])$ where lcp denotes the longest common prefix function. Together, the SA and LCP are called an enhanced suffix array.

2. Methods

2.1 Algorithm

2.1.1 Reference Sequence Indexing

A multi-FASTA file containing reference sequences is split into a sequence file, an id file and an info file. The sequence file contains the sequences delimited by the null character and optionally their reverse complements, the id file contains the sequence ids and the info file contains offsets, lengths and reverse complement status for a given sequence. The SA, ISA and LCP are generated from the sequence file using the algorithms described in [2] and [3]. These arrays only need to be generated once.
The next step in the preparation of the reference sequences is the identification of valid q-grams. A valid q-gram is a sequence of q bases that does not contain the unknown base N. If the q-grams of the query sequences are selected to not overlap then the q-grams of the reference sequences must be selected to overlap and vice versa. An inclusion vector that indicates valid q-grams corresponding to the SA can be generated from a linear scan of the reference sequences and ISA. An inclusion vector for the reference sequences must be generated for each q.

2.1.2 Query Sequence Indexing

The query sequences can be prepared in exactly the same way as the references sequences with the exception that the q-grams are selected to not overlap. A more efficient alternative is to only include the indices of valid q-grams in the SA and generate the corresponding LCP and inclusion vector. The fraction of included indices is $f = \frac{1}{q} = \frac{M+1}{L}$. The algorithm for accomplishing this starts with a linear scan of the sequences followed by a merge sort. When using this alternative, the SA, LCP and inclusion vector must be calculated for each q.

2.1.3 Alignment

To align the query sequences to the reference sequences, the SAs, LCPs and inclusion vectors are scanned linearly to find perfect q-gram matches. The first step is to calculate a running longest common prefix (RLCP) as the minimum between the current reference LCP, current query LCP and the previous RLCP. Next the inclusion vectors are checked and if either the reference position or the query position is not included, then the appropriate current index is advanced and the loop is restarted. If both are included, then the reference sequence and query sequence are compared using the indices in the SA starting at an offset of RLCP. If less than q bases match, then the appropriate current index is advanced and the loop is restarted. If q bases match, then the reference arrays are scanned for included indices and added to a set until the LCP is less than q. This is repeated for the query arrays. At this point, the two sets contain indices that are perfect q-gram matches.
The next step is to enumerate all possible pairs from each set and extend the sequences to check for a match. First, the sequences are extended to the left. The extension continues until a maximum mismatch is violated, the end of the query sequence is reached or a q-gram fails to have a mismatch. This last criteria prevents finding duplicate matches. Variable length query sequences are supported because the the maximum mismatch is calculated from a specified mismatch rate and the query sequence length. If the left extension is successful, then the sequences are extended to the right until a maximum mismatch is violated or the end of the query sequence is reached. By default, the extension phase uses dynamic programming to consider insertions and deletions. This type of extension is less efficient than only considering simple mismatches so this behavior can be turned off.
Alignments are recorded in a binary form with the rationale that sorting and filtering is easier. The binary form can be converted to a human-readable representation using the id and info files generated during sequence preparation.

2.2 Materials

The methods described above were implemented using 64-bit integers for all offsets and lengths. For the experiments, human chromosome one was choosen as the reference sequence and 150,000,000 short sequences of length 27 from James Watson's Personal Genome (http://jimwatsonsequence.cshl.edu) were choosen as the query sequences.
Alignments performed using AQUESA were performed on a server with 2 Opteron 2216 (dual-core 2.4 GHz) and 32 GB RAM. All other alignments were performed on a server with 2 Opteron 2218 HE (dual-core 2.6 GHz) and 32 GB RAM. All tests were performed using a single thread.

3. Results

Method	MIS	PCRT	PCQT	AT	EXP
MUMmer [10]	0	0	0	1,920[a]	
AQUESA	0	1,137	475	264	739
PASS [11]	0	361	626	246	872
SOAP [12]	0	0	0	50,024	50,024
Mosaik [13]	0	0	1,712	11,690	13,402
AQUESA	1	1,130	1,089	11,786	12,875
PASS	1	11,867	28,215	4,602	32,817
SOAP	1	0	0	85,807	85,807
Mosaik	1	0	1,712	144,642[b]	
AQUESA	2	1,206	2,194	209,662	211,856
PASS	2	156,311	322,593	184,924	507,517
SOAP	2	0	0	378,122	378,122

Figure 1: *Alignment Times (seconds)*
[a]Estimated. Failed at sequence 41,154,685.
[b]Estimated. Memory exceed.

Table 1 shows the time required for each algorithm to find all possible matches without insertions or deletions. MIS is the number of allowed mismatches, PCRT is the time spent pre-computing on the reference sequences, PCQT is the time spent pre-computing on the query sequences, AT is the alignment time and any post-processing or conversion time and EXP is the sum of PCQT and AT or the time required to align a new batch of short sequences to pre-existing reference sequences. The SWIFT [4], MAQ [5], MEGABLAST [6] and BLAT [7] algorithms did not finish and the Eland [8] algorithm encountered a segmentation fault. The Vmatch [9] algorithm also uses suffix arrays and we are currently in the process of acquiring and testing it. Note that some algorithms, including AQUESA, treat N as a mismatch while others treat N as a match.

4. Discussion

AQUESA has several traits which make it attractive for high throughput sequence alignment. First and foremost are recall and speed. Our algorithm compares favorably in terms of recall (not shown) and speed to the other algorithms tested. Another trait is a small memory requirement when performing the alignment since only the sequence data must be kept in memory due to random accesses. Another feature not supported by all high throughput sequence alignment algorithms is the ability to handle variable length query sequences. Our algorithm is able to accomplish this based on the properties of q-grams and a mismatch rate.
AQUESA is available at http://brainarray.mbni.med.umich.edu/Brainarray/SequenceAlignment/AQUESA/.

Acknowledgments

The authors are members of the Pritzker Neuropsychiatric Disorders Research Consortium, which is supported by the Pritzker Neuropsychiatric Disorders Research Fund L.L.C. This work is also supported in part by the National Center for Integrated Biomedical Informatics through NIH grant 1U54DA021519-01A1 to the University of Michigan.

References

[1] Manber, U. and Myers, E. W. 1990. Suffix Arrays: A New Method for On-Line String Searches. First ACM-SIAM Symposium on Discrete Algorithms. 319-327.

[2] Maniscalco, M. A. and Puglisi, S. J. 2007. An efficient, versatile approach to suffix sorting. ACM J. Exp. Algor. 12, Article 1.2, 23 pages.

[3] Kasai, T. et. al. 2001. Linear-time longest-common-prefix computation in suffix arrays and its applications. In Proceedings of the 12th Annual Symposium (CPM 2001). Lecture Notes in Computer Science, vol. 2089. Springer-Verlag, Berlin, Germany, 181-192.

[4] Rasmussen, K. R., Stoye, J. and Myers, E. W. 2005. Efficient q-gram filters for finding all epsilon-matches over a given length. Proc. 9th Annual International Conference on Research in Computational Molecular Biology (RECOMB 2005). 189-203.

[5] Li, H. Mapping and assembly with quality. Unpublished. http://maq.sourceforge.net.

[6] Zhang, Z. et. al. 2000. A greedy algorithm for aligning DNA sequences. J Comput Biol. 7(1-2):203-14.

[7] Kent, W. J. 2002. BLAT–the BLAST-like alignment tool. Genome Res. 12: 656-664.

[8] Cox, A. ELAND: Efficient Local Alignment of Nucleotide Data. Unpublished.

[9] Vmatch. http://www.vmatch.de.

[10] Delcher, A. L. et. al. 1999. Alignment of Whole Genomes. Nucleic Acids Res. 27: 2369-2376.

[11] Presorted Alignment for Short Sequences (PASS). http://brainarray.mbni.med.umich.edu/Brainarray/pass

[12] Li, R. et. al. 2008. SOAP: short oligonucleotide alignment program. Bioinformatics. 24: 713-714.

[13] Mosaik. http://bioinformatics.bc.edu/marthlab/Mosaik.