BRIEFINGS IN BIOINFORMATICS. VOL 11. NO 5. 473–483
Advance Access published on 11 May 2010
doi:10.1093/bib/bbq015

A survey of sequence alignment algorithms for next-generation sequencing

Heng Li and Nils Homer

Submitted: 3rd March 2010; Received (in revised form): 14th April 2010

Abstract

Rapidly evolving sequencing technologies produce data on an unparalleled scale. A central challenge to the analysis of this data is sequence alignment, whereby sequence reads must be compared to a reference. A wide variety of alignment algorithms and software have been subsequently developed over the past two years. In this article, we will systematically review the current development of these algorithms and introduce their practical applications on different types of experimental data. We come to the conclusion that short-read alignment is no longer the bottleneck of data analyses. We also consider future development of alignment algorithms with respect to emerging long sequence reads and the prospect of cloud computing.

Keywords: *new sequencing technologies; alignment algorithm; sequence analysis*

INTRODUCTION

The rapid development of new sequencing technologies substantially extends the scale and resolution of many biological applications, including the scan of genome-wide variation [1], identification of protein binding sites (ChIP-seq), quantitative analysis of transcriptome (RNA-seq) [2], the study of the genome-wide methylation pattern [3] and the assembly of new genomes or transcriptomes [4]. Most of these applications take alignment or de novo assembly as the first step; even in de novo assembly, sequence reads may still need to be aligned back to the assembly as most large-scale short-read assemblers [5, 6] do not track the location of each individual read. Sequence alignment is therefore essential to nearly all the applications of new sequencing technologies.

All new sequencing technologies in production, including Roche/454, Illumina, SOLiD and Helicos, are able to produce data of the order of giga base-pairs (Gbp) per machine day [7]. With the emergence of such data, researchers have quickly realized that even the best tools for aligning capillary reads [8, 9] are not efficient enough given the unprecedented amount of data. To keep pace with the throughput of sequencing technologies, many new alignment tools have been developed in the last two years. These tools exploit the many advantages specific to each new sequencing technology, such as the short sequence length of Illumina, SOLiD and Helicos reads, the di-base encoding of SOLiD reads, the high base quality towards the 5′-end of Illumina and 454 reads, the low indel error rate of Illumina reads and the low substitution error rate of Helicos reads. Short read aligners outperform traditional aligners in terms of both speed and accuracy. They greatly boost the applications of new sequencing technologies as well as the theoretical studies of alignment algorithms.

This article aims to systematically review the recent advance with respect to alignment algorithms. It is organized as follows. We first review the

Corresponding author. Heng Li, Broad Institute, 5 Cambridge Center, Cambridge, MA 02142, USA. Tel: 617 714 7000; Fax: 617 714 8102; E-mail: hengli@broadinstitute.org

Heng Li is a postdoctoral researcher at the Broad Institute and used to work at the Sanger Institute where he developed several popular alignment algorithms.

Nils Homer is a PhD student in Computer Science and Human Genetics departments of UCLA. He developed the BFAST alignment algorithm.

progress on general alignment techniques and then examine their applications in the context of specific sequencing platforms and experimental designs. We will use simulated data to evaluate the necessity of gapped alignment and paired-end mapping, and present a list of alignment software that are actively maintained and widely used. Finally, we will discuss the future development of alignment algorithms.

OVERVIEW OF ALIGNMENT ALGORITHMS

Most of fast alignment algorithms construct auxiliary data structures, called indices, for the read sequences or the reference sequence, or sometimes both. Depending on the property of the index, alignment algorithms can be largely grouped into three categories: algorithms based on hash tables, algorithms based on suffix trees and algorithms based on merge sorting. The third category only consists of Slider [10] and its descendant SliderII [11]. This review will therefore focus on the first two categories.

Algorithms based on hash tables

The idea of hash table indexing can be tracked back to BLAST [12, 13]. All hash table based algorithms essentially follow the same seed-and-extend paradigm. BLAST keeps the position of each k-mer ($k = 11$ by default) subsequence of the query in a hash table with the k-mer sequence being the key, and scans the database sequences for k-mer exact matches, called seeds, by looking up the hash table. BLAST extends and joins the seeds first without gaps and then refines them by a Smith–Waterman alignment [14, 15]. It outputs statistically significant local alignments as the final results.

The basic BLAST algorithm has been improved and adapted to alignments of different types. Nevertheless, the techniques discussed below focus on mapping a set of short query sequences against a long reference genome of the same species.

Improvement on seeding: spaced seed

BLAST seeds alignment with 11 consecutive matches by default. Ma *et al*. [16] discovered that seeding with non-consecutive matches improves sensitivity. For example, a template '1110100101 00110111' requiring 11 matches at the '1' positions is 55% more sensitive than BLAST's default template '11111111111' for two sequences of 70% similarity. A seed allowing internal mismatches is called spaced seed; the number of matches in the seed is its weight.

Eland (A.J. Cox, unpublished results) was the first program that utilized spaced seed in short-read alignment. It uses six seed templates spanning the entire short read such that a two-mismatch hit is guaranteed to be identified by at least one of the templates, no matter where the two mismatches occur. SOAP [17] adopts almost the same strategy except that it indexes the genome rather than reads. SeqMap [18] and MAQ [19] extends the method to allow k-mismatches, but to be fully sensitive to k-mismatch hits, they require $\binom{2k}{k}$ templates, which is exponential in k and thus inefficient given large k. To improve the speed, MAQ only guarantees to find two-mismatch hit in the first 28 bp of each read, the most reliable part of an Illumina read. It extends the partial match when a seed match is found.

RMAP [20, 21], which is based on the Baeza–Yates–Perleberg algorithm [22], applies a different set of seed templates. It effectively uses $k + 1$ templates to find k-mismatch hits. RMAP reduces the number of templates, but for large k, the weight of each template is small. Such a strategy cannot fully take the advantage of hash table indexing in this case as many candidates will be returned.

An improvement is achieved by Lin *et al*. [23] who present the optimal the way to design a minimum number of spaced seeds, given a specified read length, sensitivity requirement and memory usage. For example, their program ZOOM is able to identify all two-mismatch hits for 32 bp reads using five seed templates of weight 14. In comparison, RMAP uses three templates of weight 10; Eland uses six templates of weight 16 but with only 12.5 bases indexed in the hash table to reduce memory requirement. As the time complexity of spaced seed algorithm is approximately proportional to where q is the weight, m the number of templates, n the number of reads and L the genome size, ZOOM has better theoretical time complexity given limited memory.

The memory required by hashing genome is usually bytes where s is the sampling frequency [24]. It is memory demanding to hold in RAM a hash table with q larger than 15. Homer *et al*. [25] proposed a two-level indexing scheme for any large q. They build a hash table for j-long ($j < q$, typically 14) bases. To find a q-long key, they look up the hash table from the first j bases and then perform a binary search among elements stored in the resulting bucket. Looking up a q-long key takes time, only

Table 1: Popular short-read alignment software

Program	Algorithm	SOLiD	Long[a]	Gapped	PE[b]	Q[c]
Bfast	hashing ref.	Yes	No	Yes	Yes	No
Bowtie	FM-index	Yes	No	No	Yes	Yes
BWA	FM-index	Yes[d]	Yes[e]	Yes	Yes	No
MAQ	hashing reads	Yes	No	Yes[f]	Yes	Yes
Mosaik	hashing ref.	Yes	Yes	Yes	Yes	No
Novoalign[g]	hashing ref.	No	No	Yes	Yes	Yes

[a]Work well for Sanger and 454 reads, allowing gaps and clipping. [b]Paired end mapping. [c]Make use of base quality in alignment. [d]BWA trims the primer base and the first color for a color read. [e]Long-read alignment implemented in the BWA-SW module. [f]MAQ only does gapped alignment for Illumina paired-end reads. [g]Free executable for non-profit projects only.

slightly worse than the optimal speed O(1). The peak memory becomes independent of q. A similar idea is also used by Eland and MAQ, but they index reads instead of the genome.

Many other aligners [26–28] also use spaced seed with different templates designed specifically for the reference genome and sensitivity tolerances, making spaced seed the most popular approach for short-read alignment.

Improvement on seeding: q*-gram filter and multiple seed hits*

A potential problem with consecutive seed and spaced seed is they disallow gaps within the seed. Gaps are usually found afterwards in the extension step by dynamic programming, or by attempting small gaps at each read positions [17, 18]. The q-gram filter, as is implemented in SHRiMP [29] and RazerS [30], provides a possible solution to building an index natively allowing gaps. The q-gram filter is based on the observation that at the occurrence of a w-long query string with at most k differences (mismatches and gaps), the query and the w-long database substring share at least $(w+1)-(k+1)q$ common substrings of length q [31–33]. Methods based on spaced seeds and the q-gram filter are similar in that they both rely on fast lookup in a hash table. They are mainly different in that the former category initiates seed extension from one long seed match, while the latter initiates extension usually with multiple relatively short seed matches. In fact, the idea of requiring multiple seed matches is more frequently seen in capillary read aligners such as SSAHA2 and BLAT; it is a major technique to accelerate long-read alignment.

Improvements on seed extension

Due to the use of long spaced seeds, many aligners do not need to perform seed extension or only extend a seed match without gaps, which is much faster than applying a full dynamic programming. Nonetheless, several improvements over BLAST have been made regarding on seed extension. A major improvement comes from the recent advance in accelerating the standard Smith–Waterman with vectorization. The basic idea is to parallelize alignment with the CPU SIMD instructions such that multiple parts of a query sequence can be processed in one CPU cycle. Using the SSE2 CPU instructions implemented in most latest x86 CPUs, [34] derived a revised Smith–Waterman algorithm that is over 10 times faster than the standard algorithm. Novoalign (http://novocraft.com), CLC Genomics Workbench (http://clcbio.com/index.php?id=1240) and SHRiMP are known to make use of vectorization.

Another improvement is achieved by constraining dynamic programming around seeds already found in the seeding step [25, 35, 36]. Thus unnecessary visits to cells far away from seed hits in iteration are greatly reduced. In addition, Myers [37] found that a query can be aligned in full length to an L-long target sequence with up to k mismatches and gaps in $O(kL)$ time, independent of the length of the query. These techniques also help to accelerate the alignment when dynamic programming is the bottleneck.

Algorithms based on suffix/prefix tries

All algorithms in this category essentially reduce the inexact matching problem to the exact matching problem and implicitly involve two steps: identifying exact matches and building inexact alignments supported by exact matches. To find exact matches, these algorithms rely on a certain representation of suffix/prefix trie, such as suffix tree, enhanced suffix array [38] and FM-index [39]. The advantage of using a trie is that alignment to multiple identical copies of a substring in the reference is only needed to be done once because these identical copies collapse on a single path in the trie, whereas with a typical hash table index, an alignment must be performed for each copy.

It should be noted that the choice of these data structures is independent of methods for finding inexact matches. An algorithm built upon FM-index,

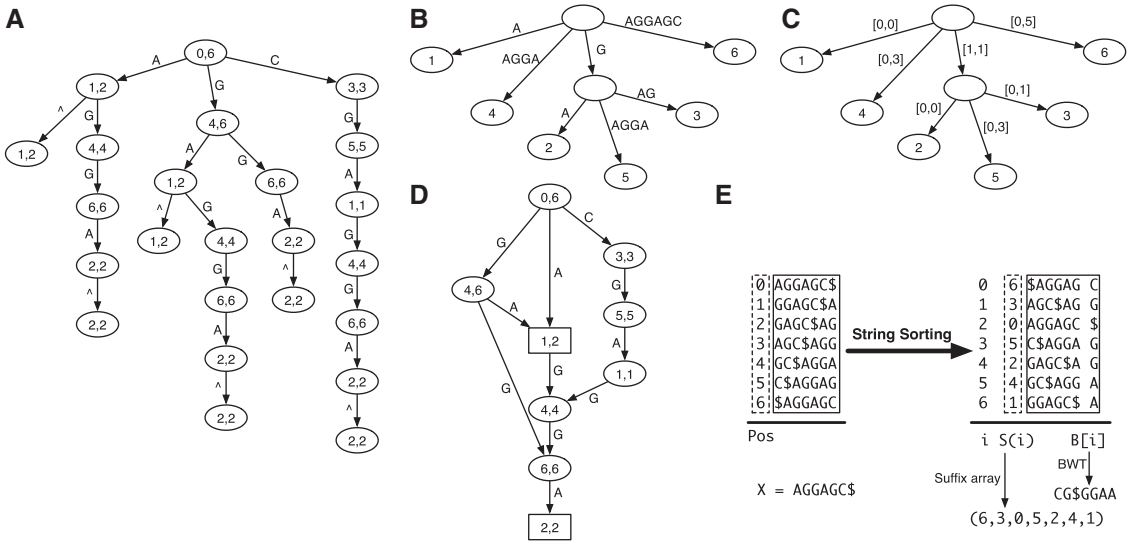

Figure 1: Data structures based on a prefix trie. (**A**) Prefix trie of string AGGAGC where symbol ^ marks the start of the string. The two numbers in each node give the suffix array interval of the substring represented by the node, which is the string concatenation of edge symbols from the node to the root. (**B**) Compressed prefix trie by contracting nodes with in- and out-degree both being one. (**C**) Prefix tree by representing the substring on each edge as the interval on the original string. (**D**) Prefix directed word graph (prefix DAWG) created by collapsing nodes of the prefix trie with identical suffix array interval. (**E**) Constructing the suffix array and Burrows–Wheeler transform of AGGAGC. The dollar symbol marks the end of the string and is lexicographically smaller than all the other symbols. The suffix array interval of a substring W is the maximal interval in the suffix array with all suffixes in the interval having W as prefix. For example, the suffix array interval of AG is [1, 2]. The two suffixes in the interval are AGC$ and AGGAGC$, starting at position 3 and 0, respectively. They are the only suffixes that have AG as prefix.

for example, would also work with suffix tree index in principle.

Trie, prefix/suffix tree and FM-index

A suffix trie, or simply a trie, is a data structure that stores all the suffixes of a string, enabling fast string matching. To establish the link between a trie and an FM-index, a data structure based on Burrows-Wheeler Transform (BWT) [40], we focus on prefix trie which is the trie of the reverse string. All algorithms on a trie can be seamlessly applied to the corresponding prefix trie.

Figure 1A gives the prefix trie of AGGAGC. Finding all exact matches of a query sequence is equivalent to searching for a path descending from the root where each edge label on the path matches a query letter in the reverse order. If such a path exists, the query is a substring. Given a query AGC, for example, the path matching the query is [0, 6]→[3, 3]→[5, 5]→[1, 1].

The time complexity of determining if a query has an exact match against a trie is linear in the length of the query, independent of the length of the reference sequence. However, a trie takes $O(L^2)$ space where L is the length of the reference. It is impractical to build a trie even for a bacterial genome. Several data structures are proposed to reduce the space. Among these data structures, a suffix tree (Figure 1C) is most widely used. It achieves linear space while allowing linear-time searching. Although it is possible in theory to represent a suffix tree in $L\log_2 L + O(L)$ bits using rank-selection operations [41], even the most space efficient implementation of bioinformatics tools requires 12–17 bytes per nucleotide [42], making it impractical to hold the suffix tree of the human genome in memory.

To solve this problem, Abouelhoda *et al.* [38] derived an enhanced suffix array that consists of a suffix array and several auxiliary arrays, taking 6.25 bytes per nucleotide. It can be regarded as an implicit representation of suffix tree, and has an identical time complexity to suffix tree in finding exact matches, better than the suffix array originally invented by Manber and Myers [43].

A further improvement on memory is achieved by Ferragina and Manzini [39] who proposed the

FM-index and found that locating a child of a parent node in the prefix trie can be done in constant time using a backwards search on this data structure. Thus the time complexity of finding exact matches with an FM-index is identical to that with a trie. With respect to memory, the FM-index was originally designed as a compressed data structure such that the theoretical index size can be smaller than the original string if the string contains repeats (equivalently, has small entropy). The FM-index is usually not compressed for better performance during alignment since DNA sequences have a small alphabet. The practical memory footprint of an FM-index is typically 0.5–2 bytes per nucleotide, depending on implementations and the parameters in use. The index of the entire human genome only takes 2–8 GB of memory.

It is worth noting that we only focus on the data structures having been used for DNA sequence alignment. There is a large volume of literature in computer science on general theory of string matching, especially on short string matching. Readers are referred to ref. [44] for a more comprehensive review in a wider scope. Nevertheless, traditional string matching algorithms strive for completeness, while many current aligners sacrifice absolute completeness for speed.

Finding inexact matches with a suffix/prefix trie

Of published aligners that can be used for query-reference alignment, MUMmer [42] and OASIS [45] are based on suffix tree, Vmatch [38] and Segemehl [46] on enhanced suffix array, and Bowtie [47], BWA [48], SOAP2 [49], BWT-SW [50] and BWA-SW [51] on FM-index. As explained above, a program built upon one representation of suffix/prefix trie can be easily migrated to another. The FM-index is most widely used mainly due to its small memory footprint.

As to the algorithms for inexact matching, MUMmer and Vmatch anchor the alignment with maximal unique matches (MUMs), maximal matches, maximal repeats or exact matches, and then join these exact matches with gapped alignment. Similarly, Segemehl initiates the alignment with the longest prefix match of each suffix, but it may also enumerate mismatches and gaps at certain positions of the query to reduce false alignments.

OASIS and BWT-SW essentially sample substrings of the reference by a top–down traversal on the trie and align these substrings against the query by dynamic programming. BWA-SW furthers BWT-SW by representing the query as a directed word graph (DAWG) [52], which also enables it to deploy heuristics to accelerate alignment.

Bowtie and BWA also sample short substrings of the reference, but instead of performing dynamic programming, they compare the query and sampled substrings only allowing a few differences. In addition, as they require the entire read to be aligned, the traversal of the trie can be bounded since it is unnecessary to descend deeper in the trie if it can be predicted that doing so leads to an alignment with excessive mismatches and gaps. Alternatively, Bowtie and BWA can be considered to enumerate all combinations of possible mismatches and gaps in the query sequence such that the altered query can be aligned exactly.

ALIGNING NEW SEQUENCING READS

The algorithms reviewed above are general techniques. Depending on the characteristics of the sequencing technologies and their applications, aligners for new sequence reads also implement extra features.

Effect of gapped alignment

Sequence reads from Illumina and SOLiD technologies were initially 25 bp in length. Performing gapped alignment for such short reads is computationally challenging because allowing a gap in this case will slow down most seeding algorithms. Fortunately, the growing read length makes gapped alignment tractable, although this feature still comes at the cost of efficiency. This raises the question about whether gapped alignment is worth doing.

From Figure 2A, it is clear that gapped alignment (curve 'gap-se') increases sensitivity by a few percent in comparison to ungapped alignment (curve 'ungap-se'), but does not reduce alignment errors. To this end, gapped alignment does not seem to be an essential feature. However, gapped alignment plays a far more important role in variant discovery [53, 54]. When gapped alignment is not implemented, a read containing an indel polymorphism may still be mapped to the correct position but with consecutive mismatches towards the underlying location of the indel. These mismatches can be seen on multiple reads mapped to the same locus, which cause most variant callers to call false SNPs. As a

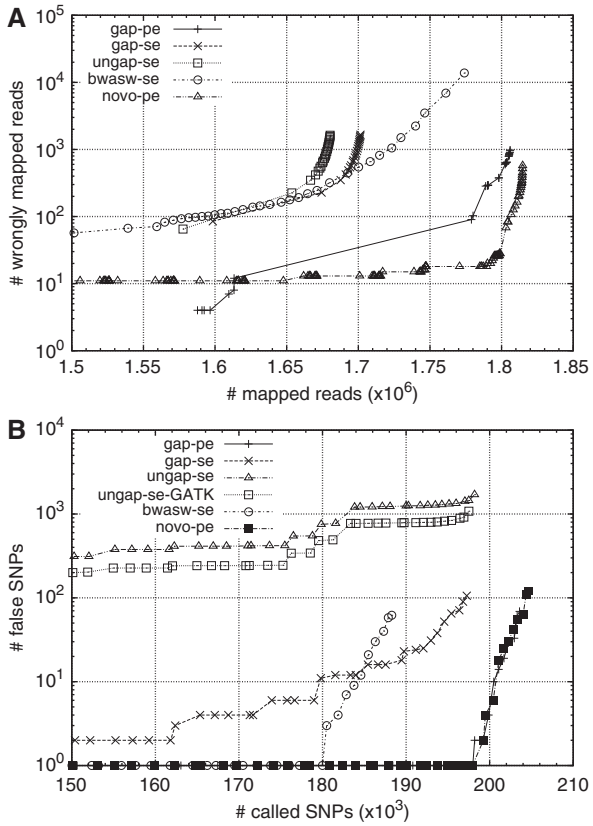

Figure 2: Alignment and SNP call accuracy under different configurations of BWA and Novoalign. (**A**) Number of misplaced reads as a function of the number of mapped reads under different mapping quality cut-off. Reads (108 bp) were simulated from human genome build36 assuming 0.085% substitution and 0.015% indel mutation rate, and 2% uniform sequencing error rate. (**B**) Number of wrong SNP calls as a function of the number of called SNP under different SNP quality cut-offs. Reads (108 bp) were simulated from chr6 of the human genome and mapped back to the whole genome. SNPs are called and filtered by SAMtools. In both figures, 'novo-pe' denotes novoalign alignment; the rest correspond to alignments under different configurations of BWA, where 'gap-pe' stands for the gapped paired-end (PE) alignment, 'gap-se' for gapped single-end (SE) alignment, 'ungap-se' for ungapped SE alignment, 'bwasw-se' for BWA-SW SE alignment, and 'ungap-se-GATK' for alignment cleaned by the GATK realigner.

result, more false SNPs are predicted from ungapped alignment (Figure 2B) and these SNPs cannot be easily filtered out even with the help of sophisticated tools such as the GATK realigner (http://tinyurl.com/broad-gatk); all high-quality false SNPs by 'gap-se' are also around undetected long indels. Furthermore, lacking gapped alignment may also lead to false structural variation calls at least for some algorithms. For example, on the simulated data used in Figure 2B, when the aligner in use does ungapped alignment only, an indel polymorphism is seen causing seven reads mapped to a wrong position with high confidence. BreakDancer [55] predicts a high scoring translocation based on the wrong alignments. Effective gapped aligners such as BWA and novoalign (http://novocraft.com) do not produce this false translocation. Therefore, gapped alignment is essential to the variant discovery, but how ChIP- and RNA-seq [2] may be affected is an open question.

Role of paired-end and mate-pair mapping

Some sequencing technologies produce read pairs such that the two reads are known to be close to each other in physical chromosomal distance. These reads are called paired-end or mate-pair reads. With this mate-pair information, a repetitive read will be reliably placed if its mate can be placed unambiguously. Alignment errors may be detected and fixed when wrong alignments break the mate-pair requirement. Figure 2A shows that paired-end alignment outperforms single-end alignment in terms of both sensitivity and specificity. The gain of sensitivity is also obvious from SNP discovery (Figure 2B).

In addition, it is worth noting here that although curve 'novo-pe' outperforms 'gap-pe' in Figure 2A, the accuracy of SNP calls are similar in Figure 2B. This is possibly because the extra alignment errors from 'gap-pe' are random and thus contribute little to variant discovery.

Using base quality in alignment

Smith *et al.* [20] discovered that using base quality scores improves alignment accuracy because knowing the error probability of each base, the aligner may pay lower penalty for an error-prone mismatch. Figure 3 shows that using base quality score halves alignment errors when the quality score is accurate. In practice, however, accurate quality score is not always available from the base calling pipeline. Recalibration of quality score is recommended to make this strategy more effective.

Aligning long sequence reads

Long reads have greater potential than short reads to contain long indels, structural variations and misassemblies in the reference genome. It is essential for a long-read aligner to be permissive about

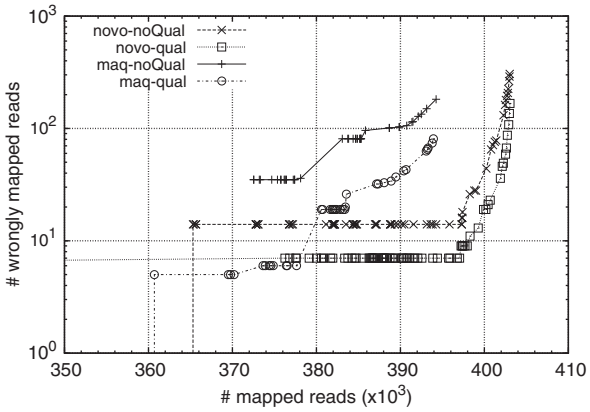

Figure 3: Alignment accuracy of simulated reads with and without base quality. Paired-end reads (51 bp) are simulated by MAQ from the human genome, assuming 0.085% substitution and 0.015% indel mutation rate. Base quality model is trained from run ERR000589 from the European short read archive. Base quality is not used in alignment for curves with labels ended with '-noQual'.

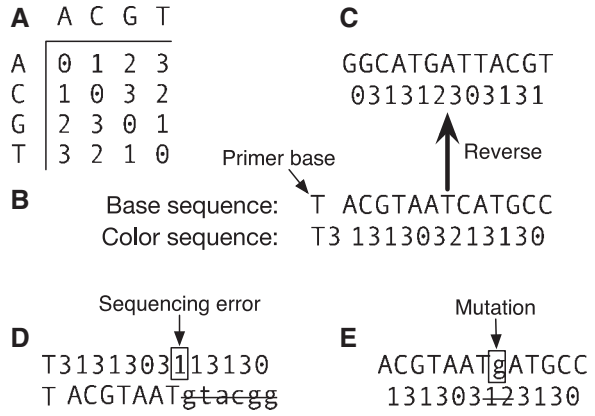

Figure 4: Color-space encoding. (**A**) Color space encoding matrix. (**B**) Conversion between base and color sequence. (**C**) The color encoding of the reverse complement of the base sequence is the reverse of the color sequence. (**D**) A sequencing error leads to contiguous errors when the color sequence is converted to base sequence. (**E**) A mutation causes two contiguous color changes.

alignment gaps and to allow partially aligned read sequences in alignment. At present, all programs capable of genome-wide long-read alignment follow the seed-and-extend paradigm, seeding the alignment using hash table index [8, 9] or more recently FM-index [50, 51], and extending seed matches with the banded Smith–Waterman algorithm. This allows for sensitive detection of indels as well as allowing for partial hits.

Aligning SOLiD reads

The SOLiD sequencing technology observes two adjacent bases simultaneously. Each dinucleotide (16 possibilities) is encoded as one of four possible colors, with the encoding referred to as color space (Figure 4A). Although the known primer base allows for the decoding of a color read to bases (Figure 4B), contiguous errors will arise from a single color sequencing error in this conversion (Figure 4D). Thus algorithms that naively decode a color read will fail. Given the fact that reverse complementing a base sequence is equivalent to reversing the color sequence (Figure 4C), the proper solution is to encode the reference as a color sequence and align color reads directly to the color reference as if they are base sequences with the exception of the complementing rule. After alignment, color sequence can be converted to base sequences with dynamic programming [48].

Performing alignment entirely in the color space may not be optimal, though. With color encoding, one base mutation leads to two contiguous color changes with some restrictions (Figure 4E). Two adjacent consistent color changes are preferred over two discontinuous changes. A better solution is to perform a color-aware Smith–Waterman alignment found in BFAST and SHRiMP [29, 56]. This extension to the standard Smith–Waterman algorithm allows the detection of indels without the aid of post-alignment analysis at the cost of increased computational complexity. Most alignment algorithms described in the previous sections can be applied to SOLiD sequencing reads with few modifications making them color space aware.

Aligning bisulfite-treated reads

Bisulfite sequencing is a technology to identify methylation patterns [3]. From the alignment point of view, unmethylated 'C' bases, or cytosines, are converted to 'T' (sequences 1 and 4 in Figure 5) and 'G' bases complement those cytosines converted to 'A' (sequences 2 and 3). Directly aligning converted sequences against the standard reference sequence would be difficult due to the excessive mismatches. Most aligners capable of bisulfite alignments [24, 57] do the following. They create two reference sequences, one with all 'C' bases converted to 'T' bases (the C-to-T reference) and the other with all 'G' bases converted to 'A' bases (the G-to-A reference). In alignment, 'C' bases are converted to 'T' base for reads and are mapped to the

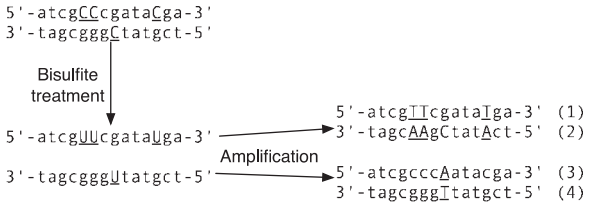

Figure 5: Bisulfite sequencing. Cytosines with underlines are not methylated. Denaturation and bisulfite treatment will convert these cytosines to uracils. After amplification, four different sequences from the original double-strand DNA result.

C-to-T reference (then a C–T mismatch is effectively regarded as a match); a similar procedure is performed for the G-to-A conversion in the next round of alignment. The results from two rounds of alignment are combined to generate the final report. If there are no mutations or sequencing errors, a bisulfite treated read can always be mapped exactly in one of the two rounds.

Aligning spliced reads

Transcriptome sequencing, or RNA-seq [2], produces reads from transcribed sequences with introns and intergenetic regions excluded. When RNA-seq reads are aligned against the genomic sequence, a read may be mapped to a splicing junction, which will fail with a standard alignment algorithm. It is possible to add sequences around known or predicted splicing junctions to the ref. [58] or more cleverly to make the alignment algorithm aware of known splicing junctions [24]. Nevertheless, novel splicing will not be discovered this way.

QPALMA [59] and TopHat [60] were developed to solve this problem. They first align reads to the genome using a standard mapping program and identify putative exons from clusters of mapped reads or from reads mapped into introns at their last few bases, possibly aided by splicing signals learnt from real data. In the next round, potential junctions are enumerated within a certain distance around the putative exons. The unmapped reads are then aligned against the sequences flanking the possible junctions. Novel junctions can thus be found. However, Trapnell *et al.* [60] have reported that only 72% of splicing sites found by ERANGE [58] can be identified by TopHat without using known splicing sites (TopHat is able to consider known splicing), indicating that incorporating known splicing sites in alignment may be necessary to RNA-seq. Readers are also referred to ref. [2] for a more comprehensive review on practical issues on processing RNA-seq data.

Realignment

Reads mapped to the same locus are highly correlated, but all read aligners map a read independent of others and thus cannot make use of the correlation between reads or the expected coverage at the same position. Especially in the presence of indels, not using this correlation may lead to wrong an alignment around the tail of a read. For indel calling, it is necessary to perform multi-alignment for reads mapped to the same locus. Realigner [61] is such a tool, but originally designed for capillary read alignment. GATK implements a different algorithm for new sequencing data. Sophisticated indel callers such as SAMtools [62] also implicitly realign reads around potential indels.

Alignment software

Over 20 short-read alignment software have been published in the past 2 years and dozens of more are still unpublished. The availability of these tools greatly boosts the development of alignment algorithms, yet only a few of them are being heavily used. Table 1 gives a list of free popular short-read alignment software packages, based on the 'tag cloud' of the SEQanswers forum (http://seqanswers.com). They all output alignments in the SAM format [62], the emerging standard alignment format which is widely supported by alignment viewers such as GBrowse [63], LookSeq [64], Tablet [65], BamView [66], Gambit (http://tinyurl.com/gambit-viewer), IGV (http://www.broadinstitute.org/igv/) and MagicViewer (http://bioinformatics.zj.cn/magicviewer/), as well as generic variant callers such as SAMtools, GATK (http://tinyurl.com/broad-gatk) VarScan [67] and BreakDancer [55].

On speed, Bowtie and BWA typically align ~7 Gbp against the human genome per CPU day. In comparison, the standard Illumina base caller, Bustard, processes ~3 Gbp per CPU day and the real-time image analysis requires similar amount of CPU time to base calling (Skelly and Bonfields, personal communication). Therefore, alignment is no longer the most time consuming step in the entire data processing pipeline. Speed improvement will not greatly reduce the time spent on data analyses.

For long reads, SSAHA2 and BLAT are still the most popular aligners. However, their alignment

speed is of the order of 0.5 Gbp per CPU day, much slower than short-read aligners. Recently developed algorithms such as Mosaik and BWA-SW are faster and may alleviate this computational bottleneck.

CONCLUSIONS AND FUTURE DEVELOPMENT

Short-read alignment is thought to be the computing bottleneck of the analysis of new sequencing data. Fortunately, the active development of alignment algorithms opens this bottleneck even with the rapidly increasing throughput of sequencing machines. In a couple of years, however, long reads will dominate again and programs developed for short reads will not be applicable; long-read alignment and *de novo* assembly will become crucial.

In addition, while major sequencing centers have sufficient localized computing resources to analyze data at present, such resources are not available to small research groups, which hamper the application of new sequencing technologies. Even between major centers, data sharing in a large collaborative project such as the 1000 genomes project (http://1000genomes.org) poses challenges. One possible solution to these problems might be cloud computing, with data uploaded and analyzed in a shared cloud. Several researchers [26, 68] have explored this approach, but establishing a cloud computing framework requires the efforts of the entire community. Furthermore, data transfer bottlenecks and leased storage have yet to be proved cost-effective for cloud computing.

Another trend of development is the simultaneous alignment against multiple genomes. Li *et al.* [69] have found the presence of extensive novel sequences absent from the human reference genome, which may lead to the loss of information when reads are aligned to a single genome. In the light of large-scale resequence projects such as the 1000 genomes project, the Drosophila population genomics project (http://dpgp.org) and the 1001 genomes project (http://1001genomes.org), alignment against multiple genomes will become increasingly important. Several groups [70, 71] have pioneered in this direction; the proposal of unifying multi-genome alignment and de novo assembly with an assembly graph (Birney and Durbin, personal communication) is attractive, but how to apply the methods given genome-wide human data is yet to be solved in practice.

Key Points

- The advent of new sequencing technologies paves the way for various biological studies, most of which involves sequence alignment in an unparalleled scale.
- The development of alignment algorithms has been successful and short-read alignment against a single reference is not the bottleneck in data analyses any more.
- With the increasing read lengths produced by the new sequencing technologies, we expect further development in multi-reference alignment, long-read alignment and de novo assembly.

Acknowledgements

We thank the three anonymous reviewers whose comments helped us to improve the manuscript.

FUNDING

H.L. is supported by the NIH grant 1U01HG005208-01 and N.H. by 1U01HG005210-01.

References

1. Dalca AV, Brudno M. Genome variation discovery with high-throughput sequencing data. *Brief Bioinform* 2010;**11**:3–14.
2. Pepke S, Wold B, Mortazavi A. Computation for ChIP-seq and RNA-seq studies. *Nat Methods* 2009;**6**:S22–32.
3. Cokus SJ, Feng S, Zhang X, *et al*. Shotgun bisulphite sequencing of the Arabidopsis genome reveals DNA methylation patterning. *Nature* 2008;**452**:215–9.
4. Flicek P, Birney E. Sense from sequence reads: methods for alignment and assembly. *Nat Methods* 2009;**6**:S6–12.
5. Simpson JT, Wong K, Jackman SD, *et al*. ABySS: a parallel assembler for short read sequence data. *Genome Res* 2009;**19**:1117–23.
6. Li R, Zhu H, Ruan J, *et al*. De novo assembly of human genomes with massively parallel short read sequencing. *Genome Res* 2010;**20**:265–72.
7. Metzker ML. Sequencing technologies – the next generation. *Nat Rev Genet* 2010;**11**:31–46.
8. Ning Z, Cox AJ, Mullikin JC. SSAHA: a fast search method for large DNA databases. *Genome Res* 2001;**11**:1725–9.
9. Kent WJ. BLAT–the BLAST-like alignment tool. *Genome Res* 2002;**12**:656–64.
10. Malhis N, Butterfield YSN, Ester M, *et al*. Slider–maximum use of probability information for alignment of short sequence reads and SNP detection. *Bioinformatics* 2009;**25**:6–13.
11. Malhis N, Jones SJ. High quality SNP calling using Illumina data at shallow coverage. *Bioinformatics* 2010;**26**:1029–35.
12. Altschul SF, Gish W, Miller W, *et al*. Basic local alignment search tool. *J Mol Biol* 1990;**215**:403–10.
13. Altschul SF, Madden TL, Schaffer AA, *et al*. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res* 1997;**25**:3389–402.
14. Smith TF, Waterman MS. Identification of common molecular subsequences. *J Mol Biol* 1981;**147**:195–7.

15. Gotoh O. An improved algorithm for matching biological sequences. *J Mol Biol* 1982;**162**:705–8.
16. Ma B, Tromp J, Li M. PatternHunter: faster and more sensitive homology search. *Bioinformatics* 2002;**18**:440–5.
17. Li R, Li Y, Kristiansen K, *et al*. SOAP: short oligonucleotide alignment program. *Bioinformatics* 2008;**24**:713–4.
18. Jiang H, Wong WH. SeqMap: mapping massive amount of oligonucleotides to the genome. *Bioinformatics* 2008;**24**: 2395–6.
19. Li H, Ruan J, Durbin R. Mapping short DNA sequencing reads and calling variants using mapping quality scores. *Genome Res* 2008;**18**:1851–8.
20. Smith AD, Xuan Z, Zhang MQ. Using quality scores and longer reads improves accuracy of Solexa read mapping. *BMC Bioinformatics* 2008;**9**:128.
21. Smith AD, Chung WY, Hodges E, *et al*. Updates to the RMAP short-read mapping software. *Bioinformatics* 2009;**25**: 2841–2.
22. Baeza-Yates RA, Perleberg CH. Fast and practical approximate string matching. In: Apostolico A, Crochemore M, Galil Z, Manber U, (eds). *CPM, Lecture Notes in Computer Science*, **Vol. 644**. Berlin: Springer, 1992:185–92.
23. Lin H, Zhang Z, Zhang MQ, *et al*. ZOOM! Zillions of oligos mapped. *Bioinformatics* 2008;**24**:2431–7.
24. Wu TD, Nacu S. Fast and SNP-tolerant detection of complex variants and splicing in short reads. *Bioinformatics* 2010; **26**:873–81.
25. Homer N, Merriman B, Nelson SF. BFAST: an alignment tool for large scale genome resequencing. *PLoS One* 2009;**4**: e7767.
26. Schatz M. CloudBurst: highly sensitive read mapping with MapReduce. *Bioinformatics* 2009;**25**:1363–9.
27. Chen Y, Souaiaia T, Chen T. PerM: efficient mapping of short sequencing reads with periodic full sensitive spaced seeds. *Bioinformatics* 2009;**25**:2514–21.
28. Clement NL, Snell Q, Clement MJ, *et al*. The GNUMAP algorithm: unbiased probabilistic mapping of oligonucleotides from next-generation sequencing. *Bioinformatics* 2010; **26**:38–45.
29. Rumble SM, Lacroute P, Dalca AV, *et al*. SHRiMP: accurate mapping of short color-space reads. *PLoS Comput Biol* 2009;**5**:e1000386.
30. Weese D, Emde AK, Rausch T, *et al*. RazerS–fast read mapping with sensitivity control. *Genome Res* 2009;**19**: 1646–54.
31. Jokinen P, Ukkonen E. Two algorithms for approximate string matching in static texts. In: *MFCS, Lecture Notes in Computer Science*, **Vol. 520**. Berlin: Springer, 1991:240–8.
32. Cao X, Li SC, Tung AKH. Indexing DNA sequences using q-Grams. In: Zhou L, Ooi BC, Meng X, (eds). *DASFAA, Lecture Notes in Computer Science*, **Vol. 3453**. Berlin: Springer, 2005:4–16.
33. Burkhardt S, Kärkkäinen J. Better filtering with gapped q-grams. In: Apostolico A, Takeda M, (eds). *CPM, Lecture Notes in Computer Science*, **Vol. 2089**. Berlin: Springer, 2001:73–85.
34. Farrar M. Striped Smith-Waterman speeds database searches six times over other SIMD implementations. *Bioinformatics* 2007;**23**:156–61.
35. Eppstein D, Galil Z, Giancarlo R, Italiano GF. Sparse dynamic programming. In: *SODA*. Philadelphia: Society for Industrial and Applied Mathematics, 1990;513–22.
36. Slater GSC, Birney E. Automated generation of heuristics for biological sequence comparison. *BMC Bioinformatics* 2005;**6**:31.
37. Myers EW. An O(ND) Difference algorithm and its variations. *Algorithmica* 1986;**1**(2):251–66.
38. Abouelhoda MI, Kurtz S, Ohlebusch E. Replacing suffix trees with enhanced suffix arrays. *J Discrete Algorithms* 2004;**2**: 53–86.
39. Ferragina P, Manzini G. Opportunistic data structures with applications. In: *Proceedings of the 41st Symposium on Foundations of Computer Science (FOCS 2000), Redondo Beach, CA, USA*. 2000;390–8.
40. Burrows M, Wheeler DJ. A block-sorting lossless data compression algorithm. Technical Report 124, Digital Equipment Corporation. CA: Palo Alto, 1994.
41. Munro JI, Raman V, Rao SS. Space efficient suffix trees. *J Algorithms* 2001;**39**(2):205–22.
42. Kurtz S, Phillippy A, Delcher AL, *et al*. Versatile and open software for comparing large genomes. *Genome Biol* 2004;**5**: R12.
43. Manber U, Myers EW. Suffix arrays: a new method for on-line string searches. *SIAM J Comput* 1993;**22**: 935–48.
44. Navarro G. A guided tour to approximate string matching. *ACM Comput Surv* 2001;**33**:31–88.
45. Meek C, Patel JM, Kasetty S. OASIS: an online and accurate technique for local-alignment searches on biological sequences. In: *Proceedings of 29th International Conference on Very Large Data Bases (VLDB 2003), Berlin*. 2003;910–21.
46. Hoffmann S, Otto C, Kurtz S, *et al*. Fast mapping of short sequences with mismatches, insertions and deletions using index structures. *PLoS Comput Biol* 2009;**5**:e1000502.
47. Langmead B, Trapnell C, Pop M, *et al*. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biol* 2009;**10**:R25.
48. Li H, Durbin R. Fast and accurate short read alignment with Burrows-Wheeler transform. *Bioinformatics* 2009;**25**: 1754–60.
49. Li R, Yu C, Li Y, *et al*. SOAP2: an improved ultrafast tool for short read alignment. *Bioinformatics* 2009; **25**:1966–7.
50. Lam TW, Sung WK, Tam SL, *et al*. Compressed indexing and local alignment of DNA. *Bioinformatics* 2008;**24**: 791–7.
51. Li H, Durbin R. Fast and accurate long-read alignment with Burrows-Wheeler transform. *Bioinformatics* 2010; **26**(5):589–95.
52. Blumer A, Blumer J, Haussler D, *et al*. The smallest automaton recognizing the subwords of a text. *Theoretical Computer Science* 1985;**40**:31–55.
53. Ossowski S, Schneeberger K, Clark RM, *et al*. Sequencing of natural strains of Arabidopsis thaliana with short reads. *Genome Res* 2008;**18**:2024–2033.
54. Krawitz P, Rödelsperger C, Jäger M, *et al*. Microindel detection in short-read sequence data. *Bioinformatics* 2010; **26**:722–9.

55. Chen K, Wallis JW, McLellan MD, *et al*. BreakDancer: an algorithm for high-resolution mapping of genomic structural variation. *Nat Methods* 2009;**6**:677–81.
56. Homer N, Merriman B, Nelson SF. Local alignment of two-base encoded DNA sequence. *BMC Bioinformatics* 2009;**10**:175.
57. Xi Y, Li W. BSMAP: whole genome bisulfite sequence MAPping program. *BMC Bioinformatics* 2009;**10**:232.
58. Mortazavi A, Williams BA, McCue K, *et al*. Mapping and quantifying mammalian transcriptomes by RNA-Seq. *Nat Methods* 2008;**5**:621–8.
59. De Bona F, Ossowski S, Schneeberger K, *et al*. Optimal spliced alignments of short sequence reads. *Bioinformatics* 2008;**24**:i174–80.
60. Trapnell C, Pachter L, Salzberg SL. TopHat: discovering splice junctions with RNA-Seq. *Bioinformatics* 2009;**25**: 1105–11.
61. Anson EL, Myers EW. ReAligner: a program for refining DNA sequence multi-alignments. *J Comput Biol* 1997;**4**(3): 369–83.
62. Li H, Handsaker B, Wysoker A, *et al*. The sequence alignment/map format and SAMtools. *Bioinformatics* 2009;**25**: 2078–9.
63. Stein LD, Mungall C, Shu S, *et al*. The generic genome browser: a building block for a model organism system database. *Genome Res* 2002;**12**(10):1599–610.
64. Manske HM, Kwiatkowski DP. LookSeq: a browser-based viewer for deep sequencing data. *Genome Res* 2009;**19**(11): 2125–32.
65. Milne I, Bayer M, Cardle L, *et al*. Tablet–next generation sequence assembly visualization. *Bioinformatics* 2010;**26**(3): 401–2.
66. Carver T, Bohme U, Otto T, *et al*. BamView: viewing mapped read alignment data in the context of the reference sequence. *Bioinformatics* 2010;**26**(5):676–7.
67. Koboldt DC, Chen K, Wylie T, *et al*. VarScan: variant detection in massively parallel sequencing of individual and pooled samples. *Bioinformatics* 2009;**25**:2283–5.
68. Langmead B, Schatz MC, Lin J, *et al*. Searching for SNPs with cloud computing. *Genome Biol* 2009;**10**(11):R134.
69. Li R, Li Y, Zheng H, *et al*. Building the sequence map of the human pan-genome. *Nat Biotechnol* 2010;**28**:57–63.
70. Schneeberger K, Hagmann J, Ossowski S, *et al*. Simultaneous alignment of short reads against multiple genomes. *Genome Biol* 2009;**10**:R98.
71. Mäkinen V, Navarro G, Sirén J, *et al*. Storage and retrieval of individual genomes. In: Batzoglou S (ed). *RECOMB, Lecture Notes in Computer Science, Vol. 5541*. Berlin: Springer, 2009;121–37.