Title

Mapping single molecule sequencing reads using basic local alignment with successive refinement (BLASR): application and theory

Journal

BMC Bioinformatics, 13(1)

ISSN

1471-2105

Authors

Chaisson, Mark J
Tesler, Glenn

Publication Date

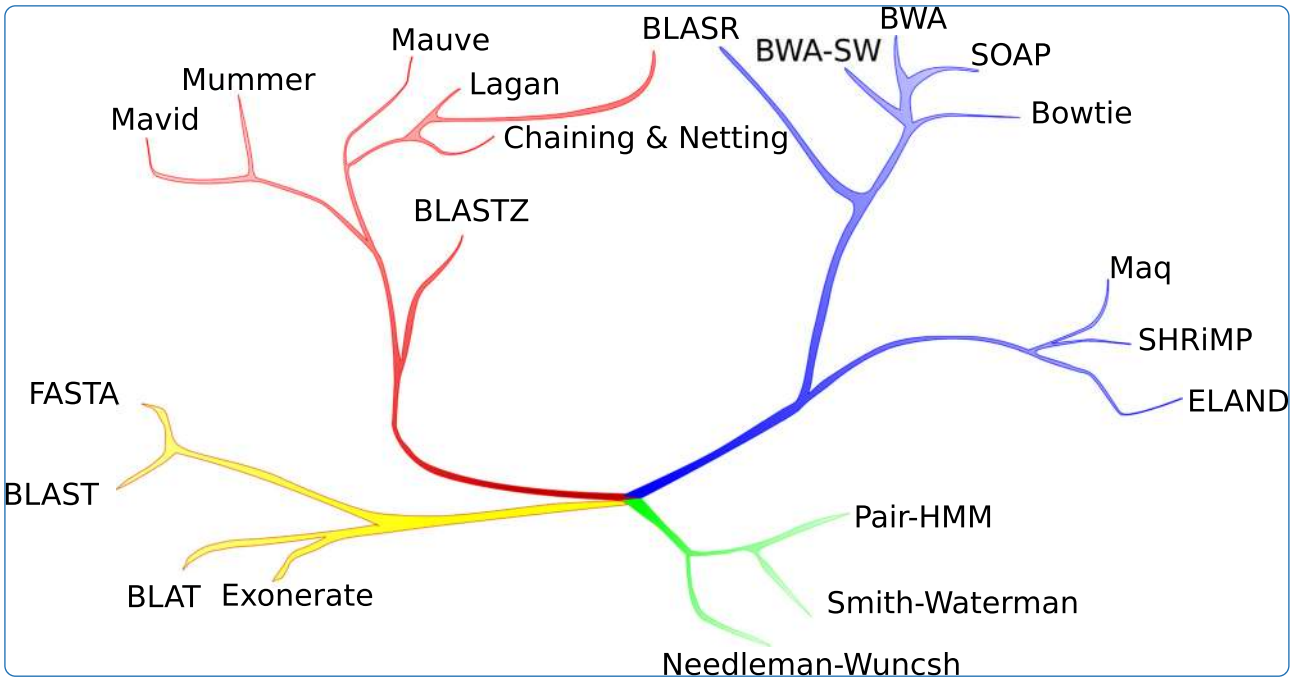
2012-09-19

DOI

http://dx.doi.org/10.1186/1471-2105-13-238

Peer reviewed

BMC
Bioinformatics



Taxon ordering in phylogenetic trees by means of evolutionary algorithms

Chaisson and Tesler

BioMed Central

Chaisson and Tesler *BMC Bioinformatics* 2012, **13**:238
http://www.biomedcentral.com/1471-2105/13/238

Chaisson and Tesler *BMC Bioinformatics* 2012, **13**:238
http://www.biomedcentral.com/1471-2105/13/238

BMC Bioinformatics

METHODOLOGY ARTICLE Open Access

Mapping single molecule sequencing reads using basic local alignment with successive refinement (BLASR): application and theory

Mark J Chaisson[1] and Glenn Tesler[2*]

Abstract

Background: Recent methods have been developed to perform high-throughput sequencing of DNA by Single Molecule Sequencing (SMS). While Next-Generation sequencing methods may produce reads up to several hundred bases long, SMS sequencing produces reads up to tens of kilobases long. Existing alignment methods are either too inefficient for high-throughput datasets, or not sensitive enough to align SMS reads, which have a higher error rate than Next-Generation sequencing.

Results: We describe the method BLASR (Basic Local Alignment with Successive Refinement) for mapping Single Molecule Sequencing (SMS) reads that are thousands of bases long, with divergence between the read and genome dominated by insertion and deletion error. The method is benchmarked using both simulated reads and reads from a bacterial sequencing project. We also present a combinatorial model of sequencing error that motivates why our approach is effective.

Conclusions: The results indicate that it is possible to map SMS reads with high accuracy and speed. Furthermore, the inferences made on the mapability of SMS reads using our combinatorial model of sequencing error are in agreement with the mapping accuracy demonstrated on simulated reads.

Background

The first step in a resequencing study is to map reads from a sample genome onto a reference, accounting for sample variance and sequencing error. An accurate and sensitive approach is to use Smith-Waterman [1] alignment; however, this is computationally infeasible for mapping to nearly any genome. Instead, methods have been created using heuristics and data structures that are appropriate for rapid mapping of the type of read considered. For example, reads produced by Sanger sequencing that are highly accurate and nearly 1000 bases long are successfully mapped using hash-based methods such as MEGABLAST [2], cross_match (Green P., www.phrap.org, *unpublished*), and BLAT [3]. These methods are too inefficient to map read sets from next generation sequencing (NGS) instruments by Illumina (San Diego, CA, USA) and Life Technologies (Carlsbad, CA, USA), since they contain hundreds of millions of short reads. Instead, methods such as Bowtie, Bwa, and Soap2 are used [4-6]. These are based on querying the Burrows-Wheeler Transform Full-text Minute-space index (BWT-FM) [7] of a genome. They are able to rapidly align reads when there is little variation between the read and the genome.

Sequencing methods based on single molecule sequencing (SMS) also produce large datasets that have high computational demands for mapping. SMS datasets do not have the length limitations of NGS or Sanger sequencing, but have a higher number of errors, and the errors are primarily insertions and deletions rather than substitutions. Thus, mapping methods created for NGS sequencing do not extend well to SMS reads. A recent study using the PacBio*RS* platform [8] included a large number of reads over 10 kilobases long. As reads become longer, the computational problem begins to resemble the whole genome alignment (WGA) problems that were examined when multiple mammalian genomes were sequenced [9-11].

*Correspondence: gptesler@math.ucsd.edu
[2]Department of Mathematics, University of California, San Diego, 9500 Gilman Dr, La Jolla, CA, USA
Full list of author information is available at the end of the article

The problem arises of how to align long (many kilobase) reads with moderate divergence from the genome (up to 20% divergence, concentrated in insertions and deletions) at the speed and sensitivity that NGS alignment methods operate.

Many alignment methods in similar application areas share related algorithmic approaches or data structures that are tailored to optimize the particular targeted application. The relationship between many existing alignment methods [1,3-5,10-23] is qualitatively illustrated in Figure 1. We present an approach, Basic Local Alignment via Successive Refinement (BLASR), which maps reads using coarse alignment methods developed during WGA studies, while speeding up these methods by using the advanced data structures employed in many NGS mapping studies.

Advances in isolation and detection of single molecules and reactions have enabled SMS methods [24-26]. These SMS methods monitor processes in real time. The PacBio*RS* instrument produces reads by detecting which fluorescently labeled nucleotides are incorporated into a DNA chain as a template sequence is replicated by DNA polymerase. Other SMS methods have been proposed using detection of cleaved bases that pass through a protein nanopore [25], and identifying bases that have translocated through a nanopore fabricated in a graphene membrane [27]. In the case of the PacBio*RS* sequencing, a missing or weak signal of nucleotide incorporation results in a deleted base, and nucleotides that give fluorescence signal without being incorporated lead to insertions.

We propose aligning SMS reads with high indel rates to genomes as follows. First, find clusters of short exact matches between the read and the genome using either a suffix array or BWT-FM index [7]. Then, perform a more detailed alignment of the regions where reads are matched to assign the alignment. To investigate the feasibility of doing this in the human genome, we need to determine two metrics: (1) the number of matches of minimal length expected to exist between a read and the genome at a given sequencing accuracy and read length, and (2) the number of false positive clusters the read is expected to have elsewhere in the genome. If the chances of finding a match between the read and the genome are low, or if there are many regions a read may map to incorrectly with high identity, our proposed approach would not be feasible. For a particular read length and accuracy, we present a method to determine the probability that the read contains a sufficient number of anchors to map; this method is based on counting integer compositions. We next examine the repeat structure of the human genome to determine how difficult it is to map to due to the repetitive nature of the genome. Rather than defining repeat content as the amount of sequence sharing high percent identity, we measure a different similarity metric on the human genome, the *anchor similarity*, where sequence similarity is measured as the number of shared anchors between the two sequences from the genome. We find that there are both a high number of expected matches between the read and the genome, and few false positive clusters of matches of the same size elsewhere in the genome, indicating that the proposed approach is feasible for mapping reads to the human genome.

We implemented our method in a program called BLASR (Basic Local Alignment with Successive Refinement), which combines the data structures used in short read mapping with alignment methods used in whole genome alignment. A BWT-FM index or suffix array of a

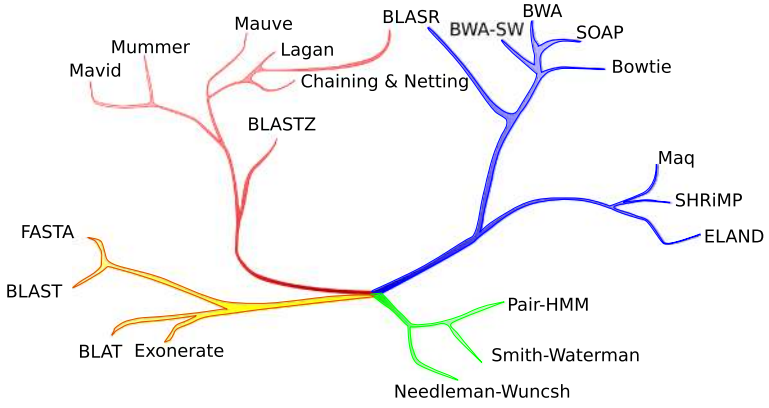


Figure 1 An illustration of relationships between alignment methods. The applications / corresponding computational restrictions shown are (green) short pairwise alignment / detailed edit model; (yellow) database search / divergent homology detection; (red) whole genome alignment / alignment of long sequences with structural rearrangements; and (blue) short read mapping / rapid alignment of massive numbers of short sequences. Although solely illustrative, methods with more similar data structures or algorithmic approaches are on closer branches. The BLASR method combines data structures from short read alignment with optimization methods from whole genome alignment.

genome is queried to generate short exact matches that are clustered and give approximate coordinates in the genome for where a read should align. A rough alignment is generated using sparse dynamic programming on a set of short exact matches in the read to the region it maps to, and a final detailed alignment is generated using dynamic programming within an area guided by the sparse dynamic programming alignment.

Results and discussion

Our results are broken down into two sections; in the first, we examine characteristics of PacBio*RS* reads, and present theory on how these sequences contain matches that may be used to anchor alignments to the genome. In the next, we present a practical comparison of alignment methods on PacBio*RS* sequences.

Mapping feasibility

Our strategy to map SMS reads is to locate a relatively small number of candidate intervals where the read may map and then use detailed pairwise alignments to determine the best candidate. The candidate intervals may be found by locating all exact matches between the read and the genome, and then finding dense clusters of exact matches (anchors) in spans of similar length and the same (or reverse complement) order and orientation in both the genome and read, as described in detail in Methods. The feasibility of the method depends on the balance of having enough anchors to detect the correct interval to align a read to, vs. having so many anchors that clustering takes a prohibitive amount of time.

One approach to limiting the number of anchors is to limit to a set of anchors of low multiplicity in the genome; this is commonly done by using longer anchors. When the sequencing error rate is ρ per position, without positional bias, the average length of an exact match is $\frac{1}{\rho} - 1$ bases. For $\rho = 0.15$, the average length is $\frac{1}{\rho} - 1 \approx 5.67$. Every word of length 5 occurs on average over 3 million times in the human genome, far too many times to be suitable as an anchor for rapid alignment. Fortunately, the condition that a sequencing error occurs precisely every $\lfloor 1/\rho \rfloor$ bases is worst-case, and is exceedingly rare: for a sequence of length L with roughly $E = \rho L$ sequencing errors, this only happens with probability on the order of $1/\binom{L}{E}$. Rather than focusing on the average case, it is more informative to consider the distribution of runs of error-free sequences; for a uniform distribution of errors across a read, this is a geometric distribution. To look at the empirical distribution of error-free sequences, a sample of reads from *Escherichia coli* sequenced by a PacBio*RS* was aligned back to the reference. The resulting distribution of spans of error-free sequences is shown in Figure 2, and closely follows the geometric distribution for over 95% of the data.

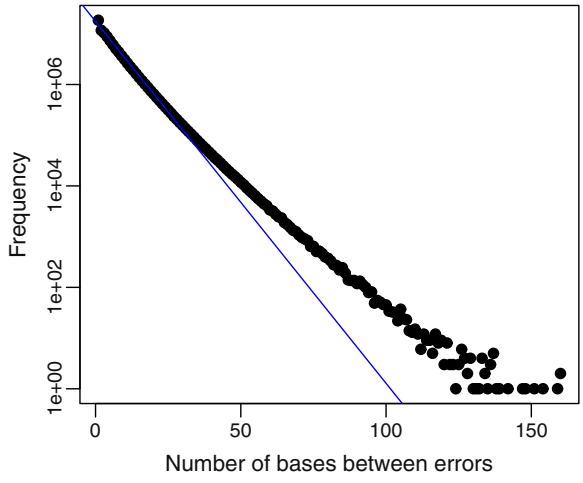


Figure 2 The distribution of lengths of error-free segments of reads. The line fitted to the points weighted by frequency has slope −0.071, corresponding to a geometric distribution with parameter 0.848, in close agreement with the 84.5% accuracy of the dataset used. Over 95% of segments are of length 20 less.

We may model SMS sequencing as a process that generates a series of error-free words with a geometric length distribution, each separated by a single erroneous base. With this model, it is possible to determine how many words must be sequenced until there is a high probability that a word of length K or greater (suitable for use in anchoring an alignment) has been sequenced. Denoting the length of a word as W, $\Pr\{W = K\} = (1-\rho)^K\rho$, and $\Pr\{W \geq K\} = (1-\rho)^K$, where $K \geq 0$. In order to have a probability of $1 - \epsilon$ that a word of length K or greater is sequenced without error, t words must be sequenced, where $t = \frac{\log(\epsilon)}{\log(1-(1-\rho)^K)}$. The **waiting length** is the corresponding number of bases for t words, each followed by one incorrect base. The waiting length is

$$t\left(1+\sum_{i=1}^{K-1} i\Pr(W=i|W<K)\right) = t\left(1+\sum_{i=1}^{K-1} i\frac{(1-\rho)^i\rho}{(1-(1-\rho)^K)}\right) = t\left(\frac{1}{\rho} - \frac{K(1-\rho)^K}{1-(1-\rho)^K}\right).$$

The waiting lengths for words of size 15, 20, and 25 are shown for $\epsilon = 0.05$ and varying ρ in Figure 3. We refer to error-free sequences of length K or greater as **anchors**.

Other alignment methods such as Gapped BLAST [28] and BLAT [3] have shown that it is useful to initiate alignments at pairs of anchors. The waiting lengths may be used to compute the length of read required to be certain of having at least N anchors. Instead of using waiting lengths, it is possible to directly compute the probability of sequencing a certain number of anchors when the error rate is known. We do this with a model that approximates all errors as point mutations on a scan across a template. Given a fixed template length L, a minimal anchor

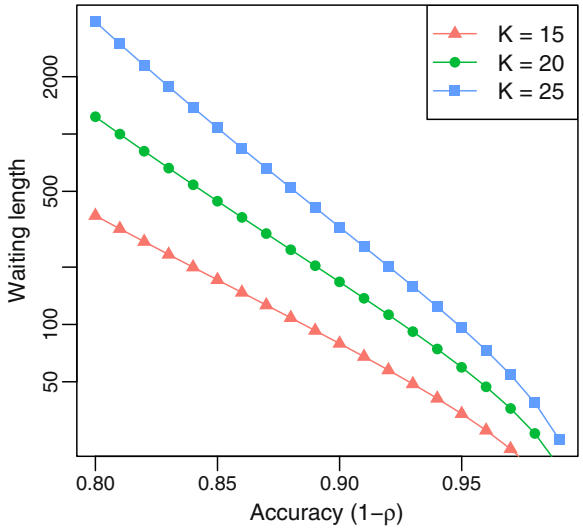


Figure 3 Waiting length to sequence a word of length $\geq k$ at $\epsilon = 0.05$. The waiting lengths to sequence a word of length $\geq k$ at $\epsilon = 0.05$ at varrying accuracy. This gives an estimate of the number of bases required to sequence before having an error free stretch that may serve as an alignment anchor.

length K, a number of errors M, and a number of anchors N, define **NumConfigurations*(M,N,K,L)*** as the number ways to distribute the positions of M errors when reading from the template such that there are at least N maximal substrings of length $\geq K$ not interrupted by error. In Appendix 1, we compute this using generating functions, allowing us to apply the result across the read lengths and error profiles found in SMS sequencing. Weese et al. [29] considered a similar problem for short reads and low error rates, and set bounds for filtering alignment hits in a q-gram based mapping method by using a dynamic programming approach.

Assuming all permutations of errors are equally likely, $\text{NumConfigurations}(M,N,K,L)/\binom{L}{M}$ gives the probability of sequencing at least N anchors. We computed this probability for the parameters $L = 1000$, and $K = 15$, 20, and 25, to study the number of anchors to use for mapping. The results are shown in Figure 4 for $M = 200$, 150, 100, and 50, corresponding to read accuracies of 80%, 85%, 90%, and 95%. At an accuracy of 85%, nearly all configurations have at least 10 anchors of length at least 15. This indicates that with minimum anchor size $K = 15$, one would expect to find at least 10 anchors at the correctly mapped interval in the genome.

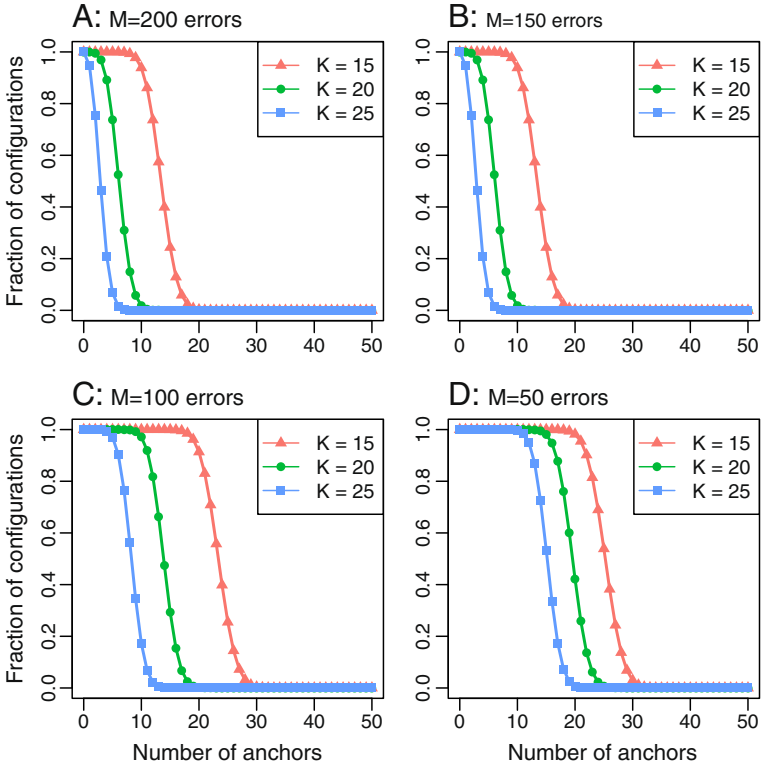


Figure 4 Values for $\text{NumConfigurations}(M,N,K,L)/\binom{L}{M}$ for parameters similar to SMS sequencing. The fraction of configurations allowing at least N anchors of length 15, 20, and 25 for N between 0 and 50 are shown for a 1000 base read when placing (**A**) 200, (**B**) 150, (**C**) 100, and (**D**) 50 errors.

When a read is sampled from a repeat in the genome, there are likely to be many dense clusters of anchors mapping the read across the genome. Assuming the repeat is divergent, it is necessary to perform a detailed alignment (Smith-Waterman) to all intervals containing dense clusters of anchors in order to distinguish the correct mapping location from other repeats. For copies of a repeat such as Alu or LINE in the human genome, the computational demands are too prohibitive to align the read against all instances of the repeat. On the other hand, if only a limited number of mapped locations are aligned in detail, the chance of finding the correct location is small. The similarity of repeats in a genome is typically defined by percent identity from a pairwise alignment of the two sequences [30]. However, sequences that have a high percent similarity may not share many long stretches of exact matches, which is how they are compared when using anchor-based mapping. To characterize repeats with respect to anchor-based mapping, we introduce an alternative metric: the **anchor similarity** of two sequences is the maximum number of fixed-length, non-overlapping, ordered anchors, shared between two sequences, with certain constraints on anchor spacing. If the anchor similarity is S, we also say the two sequences are ***S*-similar**, and **$\geq S$-similar** when two sequences have anchor similarity that is at least S. Using fixed-length anchors simplifies the presentation, although the BLASR method uses variable length anchors. Anchor similarity requires two parameters: K, the minimum anchor size; and δ, the indel rate, which may change the spacing between anchors. The constraints reflect the spacing one would expect between anchors of a read with indel errors and a genome. For example, consider a sequence that contains anchors at coordinates a and b, matching anchors at coordinates a' and b' in another sequence. If the ratio of the gaps between anchors is bounded by $1-\delta \leq \frac{b-a}{b'-a'} \leq 1+\delta$ (consistent with the indel rate), then a and b may be included in the count for the anchor similarity of the two sequences. Further details on computing anchor similarity are given in the Additional file 1: Text S1, Section 1.1.

To characterize the repetitiveness by anchor similarity of sequences in the human genome, we took a sample of 1 million random intervals of length $L = 1$ kb in the genome, and computed anchor similarity of each interval with all other intervals up to length $(1+\delta)L = 1150$ (assuming an indel rate $\delta = 0.15$) in the rest of genome. We used anchors of lengths 15, 20, and 25. For each interval and anchor length, a histogram is generated for the number of times $\geq S$-similar intervals are found in the genome. A hypothetical sample sequence with $K=15$ may have 50 thousand $\geq$1-similar intervals in the genome; one thousand $\geq$2-similar intervals; one hundred $\geq$3-similar sequences; ten $\geq$4-similar sequences; and one $\geq$5-similar sequence. This results in one million histograms (for each anchor length). To summarize these, we examined the cumulative distribution of values of all histograms for $\geq$1, $\geq$5, $\geq$10, and $\geq$20-similar sequences, as shown in Figure 5.

We compared the distribution of values of anchor similarity from the human genome with values of $\text{NumConfigurations}(M,N,K,L)/\binom{L}{M}$ to see how the mapability of sequences compares to the expected distributions of matching anchors. Reads from intervals of a genome that have low anchor-similarity to the rest of the genome are likely to have few spurious matching clusters and are thus likely to be uniquely mapped. Conversely, a read sampled from an interval that has high anchor-similarity with many other intervals likely has many clusters of matches to the genome. Figure 5 shows an estimate of the number of intervals that must be searched when using anchor-based seeding to gain a certain degree of sensitivity of finding the true match. For example, when requiring only one or more matches of length 15 to find an interval, 22% of the sequences have up to 100 matching intervals in the genome (Figure 5A, point P). If instead 20 or more matches were required in order to find an interval, 97% of the regions sampled have up to 100 matching intervals in the genome (Figure 5D, point Q). The combination of the values of $\text{NumConfigurations}(M,N,K,L)/\binom{L}{M}$ and $\geq S$-similarity search give intuition for the feasibility of mapping sequences at various error rates in the human genome. From Figure 4, for reads sequenced at 85% accuracy, it is very likely there are least 8 anchors of length 20 or greater in any read. The green points in C show the number of matching intervals when using a similar set of parameters: at least 10 anchors of length 20. Importantly, 95% of the samples match uniquely in the genome.

To gauge the mapability of sequences to various genomes, we simulated reads from *Escherichia coli*, *Arabidopsis thaliana*, and human, for read lengths that vary from 100 to 10000 bases, and error rates from 20% down to 0%. We mapped them back to their reference genomes with BLASR (see Methods). The results are shown in Figure 6. We note for mapping to the human genome, while it is difficult to have precise predictions on the mapability of sequences, the results are in agreement with the inferences drawn from the distributions of number of anchors and anchor-similarity measures. For example, 95% of 1000-base reads from the human genome simulated with a 15% error rate map to the correct location in the genome.

As shown in Figure 4B, a read with a 15% error rate has a 97% chance of having 10 anchors of length 15 or more. The anchor similarity corresponding to these reads uses parameters $\delta = 0.15, L = 1000$, and $k = 15$, and is shown by the red curve in Figure 5A. Over 90% of the sampled intervals only have one location with at least 10 anchors of

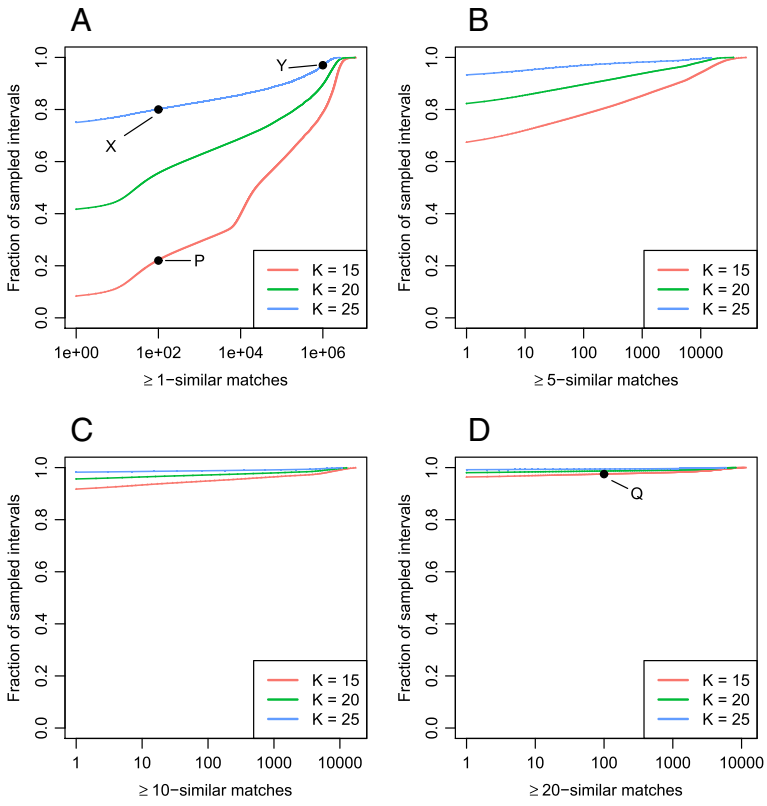


Figure 5 ≥*S*-similar sequences measured in the human genome. 1 million query intervals, each 1000 bases long, were randomly sampled from the genome. Each query interval was searched against the human genome to determine the number of non-overlapping 1000 base intervals in the genome that are ≥*S*-similar to the query. The cumulative distribution for the number of target intervals that are (**A**) ≥1-similar, (**B**) ≥5-similar, (**C**) ≥10-similar, and (**D**) ≥20-similar to these 1 million query intervals, is shown. Each panel uses minimum anchor lengths $k = 15$, 20, and 25 and indel rate $\delta = 0.15$. From this, one may interpret the number of intervals that must be searched when mapping a read using anchors. For example, when mapping with a minimum of a single 25 base match, 80% of the queries match to 100 other intervals in the genome with at least one 25 base match (point X). On the other extreme, the top 3% of queries map to over 1 million other intervals with at least one matchpoint Y), due to the high repeat content of the genome. This indicates that 80% of sequences may be correctly mapped to the human genome using a single 25 base match by only searching 100 candidates, however for full sensitivity many more candidates must be searched. Points P and Q show a contrast of the fraction of intervals that have 100 or fewer matches in the genome when matching using 1 or more anchors versus 20 or more anchors, for an anchor length of 15. Only 20% of the samples are limited to 100 or fewer additional matching intervals with at least 1 anchor (point P), and 97.5% of the samples have 100 or fewer matches when requiring at least 20 anchors in a match (point Q).

length 15, indicating they map uniquely under this repeat metric. The other two genomes, *E. coli*, and *A. thaliana*, are shown for guidance.

Mapping benchmarks

We generated three datasets for evaluating mapping speed and accuracy of different aligners on SMS reads (see Table 1). For all *E. coli* datasets, reads were aligned to the genome of an isolate of the O104:H4 strain (doi: 10.5524/100001). The source reads are available at http://bix.ucsd.edu/projects/blasr. Performance was measured additionally with both BLAT and the BWA-SW aligners [18]. BWA-SW was the first mapping method written that used both the BWT-FM index used in short read mapping and methods that allow mapping long reads with indel error. This method is very compact (under 5 GB of memory for human genome alignments), and very sensitive to mapping reads with indel error, as compared to other existing methods. Other methods that were tested either did not run or produced insufficient results. This may be expected, as these methods are highly optimized for other types of data that is either short read or whole genome sequences. Of the programs that did not run, Soap2 and Lagan crashed, while Bowtie did not accept the read input due to read length, and the mapping sensitivity was low on reads truncated to the maximum allowed length. The Mosaik (Strömberg M., http://bioinformatics.bc.edu/marthlab/Mosaik, *unpublished*), Mummer, and RazerS methods did execute, however the first two could only align to one chromosome of the human genome at a time due to space limitations, and were orders of magnitude slower than either BLASR

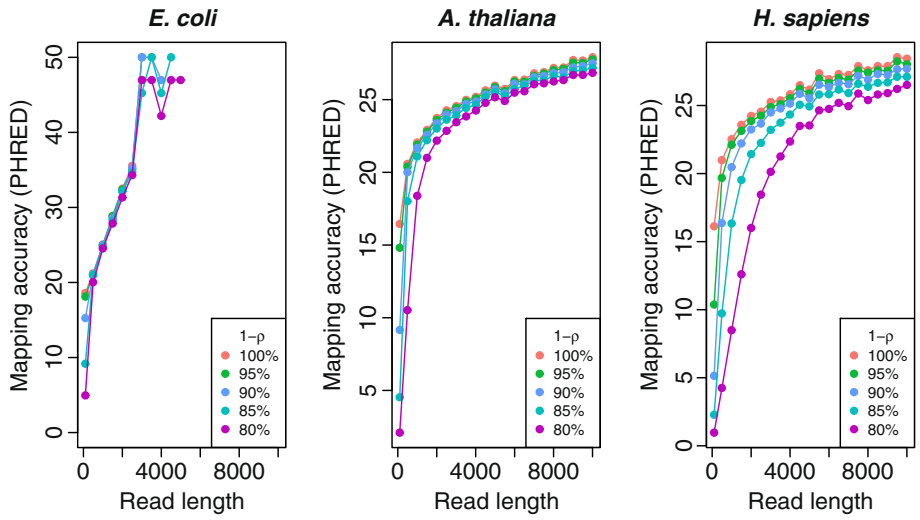


Figure 6 The mapability of simulated sequences from the *E. coli*, *A. thaliana*, and human genomes. Mapping accuracy is shown on a Phred scale ($-10 \log \frac{\text{missing+mismapped}}{\text{total}}$) for all three plots. Reads were simulated with base accuracies $1 - \rho = 80\%, 85\%, \ldots, 100\%$. In the fraction ρ of positions that are erroneous, we simulated 10% substitutions, 62% insertions, and 28% deletions. Missing values have no mismapped reads.

or BWA-SW while finding very few hits. Finally, the RazerS method was only tested on *E. coli* reads, and found few hits across all tested parameters. Because of the low mapping sensitivity, these methods were excluded from benchmarking results. The BLAT method is included as a reference for comparison to methods optimized for mapping Sanger sequences, though it is slower and less sensitive than both BLASR and BWA-SW.

The *E. coli*-PacBio*RS* dataset contains 123,246 reads comprising 261.7 M bases after filtering, with lengths and error rate shown in Figure 7 (Short Read Archive accession numbers SRR305922, SRR305923, SRR305924, and SRR305925). The reads contain 10.7% insertion, 4.3% deletion, and 0.9% substitution error, though the details are sensitive to alignment penalty parameters. A summary of the mapping statistics from each of the three programs is shown in Table 2. All programs were executed on a single core of a 2.9 GHz Xeon processor. The parameters used for each program are given in Additional file 2: Table S1.

To test the sensitivity and specificity of mapping, reads were simulated using an empirical model (described in Additional file 1: Text S1, Section 1.2) based on the measurement of error rates from reads aligned to *E. coli*. The results are shown in Table 3. The methods are largely in agreement on the reads that are correctly mapped, as well as in the number of bases from every read, and BLASR is marginally faster. The slight differences in mapping statistics between BLASR-SA and BLASR-BWT are due to implementation differences in the order anchors are generated: using a suffix array, sequences are searched left to right, but for a BWT-FM index, sequences are searched right to left. One difference between BLASR

Table 1 Datasets used in benchmarking

Dataset	Description
E. coli-PacBio*RS*	*E. coli* O104:H4 sequenced at 48× coverage by the Pacific Biosciences-*RS* sequencer.
E. coli-simulated	50× coverage of reads simulated from *E. coli* O104:H4.
H. sapiens	100 MB of reads simulated from the human genome.

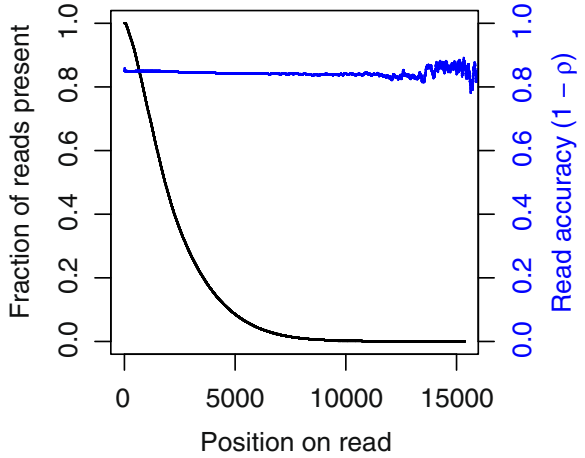


Figure 7 Statistics of reads from *E. coli* O104:H4 produced by the PacBio*RS* sequencing platform. (*Black*) The fraction of reads with length at least x. This is roughly the survival curve of an exponential distribution. (*Blue*) The fraction of reads (of length at least x) that are correct at position x. Accuracy is nearly position independent, so the blue curve is roughly the constant $1 - \rho$, where ρ is the error rate per position.

Table 2 A comparison of the BLASR, BWA-SW, and BLAT methods on *E. coli* reads

Method	Number of aligned reads	Number of aligned bases	Run time
BLASR-SA	94057	230.8 M	20m 54s
BLASR-BWT	94527	230.1 M	33m 57s
BWA-SW	97729	132.4 M	434m 5s
BLAT	99530	181.7 M	4724m 40s

Each method was used to align 48× coverage of reads from *E. coli* O104:H4. BLASR-SA uses a suffix array index of the genome, while BLASR-BWT uses a BWT-FM index of the genome.

and BWA-SW is that BWA-SW often produces several short alignments of possibly overlapping substrings of a read rather than one contiguous alignment. We consider the number of bases mapped by BWA-SW as the sum of uniquely mapped bases from each read. Usually this does not affect mapping and consensus quality, but occasionally there are subsequences from reads that are incorrectly mapped while the rest of the read is mapped correctly.

In addition to the information encoding the alignment, BLASR produces a mapping quality value for every alignment. This value represents the PHRED scale probability that the coordinates the read is aligned to in the genome are incorrect, similar to the mapping quality values produced by Maq [20]. To test mapping quality values, we created three datasets of 10M simulated reads sampled from the genome with fixed read lengths of one, two, and three kilobases each. Errors were added to the reads using the empirical read simulator (Additional file 1: Text S1, Section 1.2). For each mapped read, we classified it as correctly and incorrectly mapped, allowing a measurement of accuracy of mapping quality value. The frequency of computed mapping quality values are shown in Figure 8A. The mapping quality values are largely binary, owing to the fact most reads contain sequences that align uniquely to the genome. The empirical mapping quality values are shown in Figure 8B.

Conclusion

Methods to produce reads through single molecule sequencing were mostly theoretical a decade ago and are now produced in high throughput on an industrial platform. The different characteristics of the sequences produced by SMS relative to Next Generation sequencing (sequences several orders of magnitude longer than previous technologies, at the expense of a higher error rate concentrated in insertions and deletions), require new computational techniques to be efficiently analyzed. Here, we addressed the problem of mapping SMS reads to a reference genome by first examining the feasibility of mapping SMS reads, and then by benchmarking our new alignment method on reads produced by the PacBio*RS* instrument. The source code is available under the BSD license at https://github.com/PacificBiosciences/blasr and is the default alignment method available to all running the PacBio*RS*.

There are many emerging problems for processing SMS sequences. As the length of the reads produced by SMS increases, the computational problem resembles whole genome alignment more than the read mapping problem. This increases the need to have methods that accurately detect structural rearrangements covered by single reads. Furthermore, with the inevitable exponential increase in sequencing throughput, the current methods will not be sufficient to align SMS reads without a large amount of time or computational resources, and further algorithmic improvements will be necessary. We did not address the issue of using multiple sequence alignment to produce a consensus sequence or variant calls. It has been shown that the additional information one may gain by observing the signal from single-molecule events in real time may indicate DNA modifications such as methylation [25,31].

Table 3 A comparison of the BLASR, and BWA-SW methods on simulated reads

Method	Correctly mapped		Incorrectly mapped		Skipped	Runtime	Memory
	reads	bases	reads	bases	reads		footprint
E. coli							
BLASR-SA	108789	266.5M	229	0.38M	3766	48m 18s	202 MB
BLASR-BWT	108795	265.3M	259	0.45M	3604	59m 39s	46 MB
BWA-SW	111192	261.9M	1835	0.91M	3005	223m 57s	190 MB
H. sapiens							
BLASR-SA	41726	102.3M	1074	1.89M	413	92m 26s	14.7 GB
BLASR-BWT	41582	101.7M	1159	1.75M	472	53m 26s	8.1 GB
BWA-SW	40381	96.3M	292	1.16M	1554	105m 24s	4.2 GB

Reads are simulated from *E. coli* and *H. sapiens* with length and accuracy parameters modeled from real reads from *E. coli*. Skipped reads are either marked as filtered in the SAM output, or missing from the output.

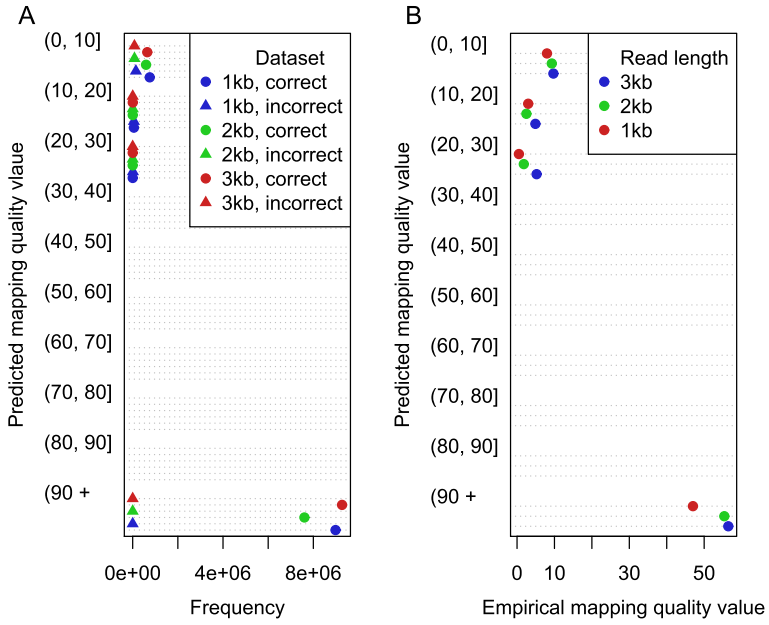


Figure 8 Mapping quality values of reads simulated from the human genome. (**A**) The frequency of quality values for alignments of 10^6 simulated 1000, 2000, and 3000 base sequences from the human genome. (**B**) The empirical mapping quality values of the alignments.

Thus, methods that produce consensus calls from SMS sequences may reveal more information about the sample sequence if this extra information is used. We aim to address many of these problem in subsequent iterations of the BLASR method.

Methods

We use a *successive refinement* approach to map SMS reads. This approach operates in three phases: (1) detecting candidate intervals by clustering short, exact matches; (2) approximate alignment of reads to candidate intervals using sparse dynamic programming; and (3) detailed banded alignment using the sparse dynamic programming alignment as a guide, as shown in Figure 9. It is not until the third step that read base positions are assigned to reference positions.

Detecting candidate intervals

The input to the BLASR method is a read r with nucleotides $r_1, \ldots, r_R$; a genome g with nucleotides $g_1, \ldots, g_G$; and a minimum match length, K. Other parameters that modify small details of mapping are introduced in their context later. We find all exact matches of substrings (of length at least K) from the read and the genome. An exact match of anchor a to the genome may be described by a triplet $(\text{Read}(a), \text{Genome}(a), l(a))$, where $\text{Read}(a)$ is the start of the match in the read; $\text{Genome}(a)$ is the start of the match in the genome; and $l(a)$ is the length of the match. The set of all matches is $\mathcal{A}$.

We use either a suffix array (SA) or BWT-FM index on the genome to query for exact matches, depending on time and space requirements. While some NGS alignment methods such as mrFAST and RazerS match using hash

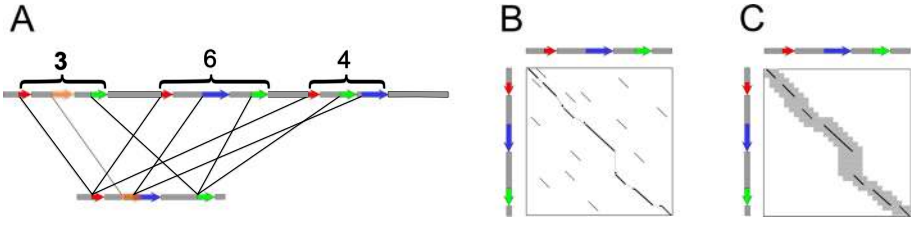


Figure 9 Overview of the BLASR method. (**A**) Candidate intervals are found by mapping short, exact matches as shown by colored arrows. Either a suffix array or BWT-FM index of the genome are used to find the exact matches. Intervals are defined over clusters of matches and are ranked; intervals with score 3, 6, and 4 are shown. (**B**) Matches scoring above a threshold are aligned using sparse dynamic programming on shorter exact matches. (**C**) Alignments that have a high-scoring sparse-dynamic programming score are realigned by dynamic programming over a subset of cells defined using the sparse dynamic programming alignment as a guide.

tables on fixed width words (q-grams) [29,32], the SA and BWT-FM index allow matching long exact matches if they exist, and also encode positions of shorter matches if a more sensitive search is required. The two data structures support the same queries: $c = \text{Count}(q, t)$, the number of times a query sequence r occurs exactly in a text g; and $\mathcal{P} = \{p_1, \ldots, p_c\} = \text{Locate}(q, t)$, the starting positions of all instances of r in g. Without changing the computational complexity of these queries, they may be modified to answer equivalent queries for counts and locations of the longest common prefix (LCP) between a query and a genome. Let $(c, l) = \text{CountLcp}(q, t)$ be the operation that finds the count c and length l of the LCP between q and t. We locate anchors by greedily finding matches slightly *shorter* than the LCP (specified by a parameter defaulting to 1 base shorter than the LCP) to increase sensitivity and avoid using an LCP that erroneously ends in a sequencing error. The minimum length anchor that is allowed is of length K, where $K = 12$ in most applications. To build $\mathcal{A}$, we scan across all positions in a read $i \in \{1, \ldots, R-k\}$; we compute $(c, l_i) = \text{CountLcp}(r_{i,\ldots,R}, g)$ and $\mathcal{P}_i = \text{Locate}(r_{i,\ldots,i+l_i-e}, g)$; and then for all positions $p^i_j \in \mathcal{P}^i$, we include in $\mathcal{A}$ a match a with $\text{Read}(a) = i$, $\text{Genome}(a) = p^i_j$, and $l(a) = l_i$. We choose a parameter MaxCount, which specifies the maximum number of times we allow a match to appear to generate an anchor. We exclude positions mapped when $|\mathcal{P}^i| >$ MaxCount, or short matches when $l_i < k$.

Descriptions of the implementation and methods for the Count and Locate queries using suffix arrays are given in [33]. Similar descriptions for the BWT-FM index are in [7] and [4]. The CountLcp operation is about 1.5× faster using a suffix array than a BWT-FM index, in our tests searching the human genome and limiting the number of times an LCP occurs to 10,000; however, the space usage for the index on a human genome is 12.8 GB with a suffix array, vs. 4.8 GB in our implementation of a BWT-FM index. Our implementation of the Locate operation is faster for larger genomes using the BWT-FM index than the suffix array when using SIMD hardware optimization. Because either index is shared across many threads, the amortized space usage is modest for both data structures.

Once the set of anchors $\mathcal{A}$ is generated, we cluster anchors using global chaining [34]. To do so, we first sort $\mathcal{A}$ by position in the genome and then by position in the read. Next, clusters of anchors are found in intervals roughly the length of the read. For every anchor $a_i \in \mathcal{A}$, a set $\mathcal{A}_i$ is created with $\mathcal{A}_i = \{a_j \in \mathcal{A} : 0 \le \text{Genome}(a_j) + l(a_j) - \text{Genome}(a_i) \le R\}$. For every set $\mathcal{A}_i$, we find a maximal subset (using global chaining) of anchors, $\mathcal{C}_i \subset \mathcal{A}_i$, that are not overlapping and are increasing in both $\text{Read}(a)$ and $\text{Genome}(a)$. For later use in evaluating the mapping quality value of a read, for each cluster, we record the sum of all $l(a)$ values for all anchors in $\mathcal{C}_i$.

The clusters are assigned a frequency weighted score that is the sum $\sum_{a_j \in \mathcal{C}_i} \log(1/\text{Freq}(a_j))$, where $\text{Freq}(a_j)$ is the frequency of the sequence of a_j in the genome, and are ranked by this score. Only the top MaxCandidates clusters are retained (typically 10). The original indexing of clusters by anchor position is replaced by indexing by rank of the frequency-weighted score. The subscript notation is dropped and rank of a cluster is indicated by the superscript. The remaining clusters are denoted $\mathcal{C}^1, \mathcal{C}^2, \ldots, \mathcal{C}^n$, where $\text{rank}(\mathcal{C}^1) \le \text{rank}(\mathcal{C}^2) \le \ldots \le \text{rank}(\mathcal{C}^n)$, and $n \le$ MaxCandidates.

While limiting the number of clusters retained may miss alignments to repetitive regions, filtering clusters on this frequency-weighted score was shown to be highly discriminative in our tests.

Refining alignments

Each cluster $\mathcal{C}^i$ is used to define an interval to which the read is realigned and rescored using sparse dynamic programming (SDP) [35]. To help describe how the interval is defined, let a^{FIRST} (a^{LAST}) be the anchors with least (greatest) $\text{Genome}(a)$ and $\text{Read}(a)$ coordinates in $\mathcal{C}^i$, ordered by position in genome and then by read. The anchors in $\mathcal{C}^i$ frequently do not contain the first and last bases in the read, and the actual starting and ending positions of the read are unknown due to insertion and deletion error in the read. Considering δ to be the maximum insertion rate of the instrument, the starting position of the interval aligned from the genome is $s = \text{Genome}(a^{\text{FIRST}}) - (1 + \delta)\text{Read}(a^{\text{FIRST}})$, and ending position $f = \text{Genome}(a^{\text{LAST}}) + (1 + \delta)(R - (\text{Read}(a^{\text{LAST}}) + l(a^{\text{LAST}})))$, of length $l_{\mathcal{C}} = f - s$.

The read must be quickly aligned to a candidate interval, even if it is many tens of kilobases long. Similar to the method of anchoring the interval to the genome but on a smaller scale, a set of matches are found between the read and the candidate interval. The matches used in SDP are of a short fixed length, K^{SDP} (typically 8–11 bases). Let $\mathcal{A}^{\text{SDP}}$ be the set of anchors of length K^{SDP} that are exact matches between the read and the genome interval $g_s, \ldots, g_f$. Sparse dynamic programming finds the largest subset of anchors $\mathcal{C}^{\text{SDP}} \subseteq \mathcal{A}^{\text{SDP}}$ that are of increasing $\text{Read}(a)$ and $\text{Genome}(a)$ values.

The SDP alignment does not align all bases in a read, and so it is necessary to realign a final time using banded dynamic programming. For long reads with indels, the size of the band used to contain the entire alignment becomes prohibitively large. The set of anchors $\mathcal{C}^{\text{SDP}}$ forms a guide for performing a banded dynamic programming alignment where the band follows the layout of the anchors in $\mathcal{C}^{\text{SDP}}$, shown in Figure 9C. The subset of cells included from full $R \times l_{\mathcal{C}}$ dynamic programming grid include a band of length b^{SDP} centered about the diagonal where there are anchors, as well as a banded alignment of size b^{drift} between anchors where b^{drift} is

the off-diagonal distance between adjacent anchors + b^{SDP}.

In addition to the base sequences produced by the PacBio*RS*, each base in the read is also given three quality values (insertion, deletion, and substitution) and two alternative base calls (substituted base and deleted base). Let $\mathcal{I}$, $\mathcal{S}$, $\mathcal{D}$ be the insertion, substitution, and deletion quality value arrays for a read, and $\hat{\mathcal{S}}$ and $\hat{\mathcal{D}}$ be the deletion and substitution nucleotide arrays. We use these quality values to compute the score of each cell $s_{i,j}$ in the dynamic programming matrix according to:

$$s_{i,j} = \min \begin{cases} s_{i-1,j-1} + \begin{cases} 0 & \text{if } r_i = g_j \\ \mathcal{S}_i & \text{if } r_i \neq g_j, \hat{\mathcal{S}}_i = g_j \\ \text{MismatchPrior} & \text{otherwise} \end{cases} \\ s_{i-1,j} + \mathcal{I}_i \\ s_{i,j-1} + \begin{cases} \mathcal{D}_i & \text{if } \hat{\mathcal{D}}_i = g_{j-1} \\ \text{DeletionPrior} & \text{otherwise.} \end{cases} \end{cases}$$

The MismatchPrior and DeletionPrior are PHRED scaled penalties that reflect the global mismatch and deletion rates. In practice, MismatchPrior is 20 and DeletionPrior is 15.

Mapping quality values

Due to the repetitive nature of genomes, a read often maps with a high alignment score to many locations. It is informative to calculate the probability that the interval a read is mapped to by an alignment is the correct location in the genome. This probability may be interpreted as a *mapping quality* value $\mathcal{Q}$ for an alignment, allowing downstream analysis such as variant calling to filter alignment by quality.

A Bayesian probability technique was presented in [20] to compute the mapping quality for short reads with base calling quality values. We present the formulation in [20] using the notation in this paper: we are given read r and a position m that it is mapped to in a sequence g. The *posterior mapping probability* that a read r is sampled from m is computed as

$$\Pr_s(m|r,g) = \frac{\Pr(r|g_{m,\ldots,m+R-1}) \Pr(m)}{\sum_i \Pr(r|g_{i,\ldots,i+R-1}) \Pr(i)}, \quad (1)$$

where i runs over all positions in the genome. The probability that position i is sampled by the sequencer is denoted $\Pr(i)$, and is considered to be uniform both here and in [20]. The quantity $\Pr(r|g_{i,\ldots,i+R-1})$ is the probability of observing the read r if the sequence at positions $i, \ldots, i+R-1$ in the genome is read by the sequencer. For reads that include base quality values q, let q_i denote the probability that a base in a read is incorrect. Then $\Pr(r|g)$ may be replaced by $\Pr(r|g,q)$. In [20], $\Pr(r|g,q)$ is rapidly approximated by summing the quality values of bases that mismatch in the ungapped alignment between r and $g_{i,\ldots,i+R-1}$. When there are insertions and deletions in the sequence, the value $\Pr(r|g_{i,\ldots,i+R-1})$ may be computed as $\Pr_f(r|g_{i,\ldots,i+R-1}, \mathcal{H})$; this denotes the forward algorithm probability using a pairwise hidden Markov model (Pair-HMM) $\mathcal{H}$ that encodes probabilities for substitution, insertion, and deletion at every position.

The denominator of Equation 1 gives the marginal probability that the read is observed from anywhere in the genome. Evaluating this full sum is computationally infeasible even for short reads and ungapped alignments. Since the probability of observing a read given a template sequence drops geometrically with divergence, most positions in the genome do not contribute significantly to the sum. For short reads, the sum is approximated in [20] as the sum of the probability of the top scoring alignment and all second best alignments.

In BLASR, the mapping quality value is calculated in a similar manner. The sum in Equation 1 is limited to the top MaxCandidates alignments, and is then scaled by a factor that reflects the limited sample size by aligning only at most MaxCandidates clusters. When the read is sampled from a unique region of the genome, there will be few clusters of high score, and the highest scoring cluster will likely contain the true match to the genome. However, when the read is sampled entirely from a repetitive sequence, there will be many high scoripng clusters. In this case, it is possible the cluster from the correct interval on the genome will not have high enough score to be retained in MaxCandidates clusters. To account for this, we assume that the correct interval in the genome may correspond to any significantly highly scoring cluster, and multiply the sum in Equation 1 by the ratio of the number of significant clusters found in the genome to MaxCandidates, as long as the number of significantly highly scoring clusters is greater than MaxCandidates. The significance of a cluster may be measured by comparing the number of anchors in a cluster to the number of anchors expected at the correctly mapped location. The distributions of numbers of anchors expected to correctly map were found using simulations of error processes for different error rates and read lengths; however, it is possible to model this theoretically (see the next section). The expected number of anchors a read has when mapped to the correct location is genome-independent: it depends only on the error rate, length of the read, and minimum anchor length. We use a slightly different metric, the number of anchor-bases (the total number of bases in all anchors) to measure cluster significance, and this is similarly genome-independent. For efficiency, in BLASR we precompute the expectation and variance for the number of anchor-bases for a range of feasible accuracies, read lengths, and minimum match lengths, and minimum match size. The accuracy of the highest scoring alignment is used as a proxy for the true accuracy of the read. Given

the accuracy, the length of the aligned sequence, and the minimum match length, we look up the mean μ and variance σ^2 for number of anchor bases, and count all clusters with more than $\mu - 2\sigma$ anchor bases as significant.

Appendix 1

Enumeration of configurations with specified numbers of errors and anchors

In this section, we will show how to explicitly compute NumConfigurations(M, N, K, L).

Consider a read of length L with exactly M errors, at positions $1 \le x_1 < x_2 < \ldots < x_M \le L$. Also set $x_0 = 0$ and $x_{M+1} = L + 1$.

For the sake of simplicity, we assume all sequencing errors are of length 1, but this can be generalized to insertions and deletions that change the length of the read.

The error positions split the read into **parts** of sizes $\lambda_i = x_i - x_{i-1} \ge 1$ for $i = 1, \ldots, M + 1$. Each part λ_i $(i = 1, \ldots, M)$ consists of $\lambda_i - 1$ matches followed by one mismatch. The last part consists of $\lambda_{M+1} - 1$ matches. Note that if there are two consecutive mismatches, there will be a part $\lambda_i = 1$ corresponding to 0 matches followed by one mismatch.

Part sizes λ_i are related to the notation W of the Results section by $\lambda_i = W + 1$. Note that W counted only the correct positions, and we did not have a subscript (W_i) to specify the word number. In this section, λ_i counts the correct bases and also counts one incorrect base at the end, based on our simplification that all sequencing errors are of length one.

Set $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_{M+1})$. These are positive integers that add up to $L + 1$. In Combinatorics, this is called a **strict composition** of $L + 1$ into $M + 1$ parts. Let K be the minimum anchor length (a parameter).

Consecutive errors greater than K apart $(\lambda_i > K)$ give segments that are **anchors** while consecutive errors shorter than this $(\lambda_i \le K)$ give segments called **short matches**.

In Figure 10, we illustrate a read of length $L = 7$ with $M = 2$ error positions. For a minimum anchor length $K = 3$, there are 6 compositions where the first part is an anchor:

$$(4, 3, 1),\ (4, 2, 2),\ (4, 1, 3),\ (5, 2, 1),\ (5, 1, 2),\ (6, 1, 1).$$

For reads of length 7 with 2 errors, and minimum anchor length 3, the number of compositions with exactly one anchor (allowing it to be any of the parts, via permutations of these compositions) is $6 \cdot 3 = 18$.

For arbitrary values of the parameters, we first compute the number of configurations where all N anchors come first and all $M + 1 - N$ short parts come last. Then we multiply the count of these by $\binom{M+1}{N}$ to allow any of the N parts to be the anchors.

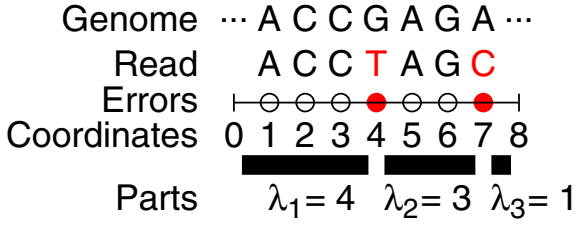


Figure 10 Toy example for counting components. A read of length $L = 7$ with $M = 2$ errors is shown, with errors in red. In general, M errors splits the read into $M + 1$ parts, some of which may be null; in this case, the third part is null. For anchor length threshold $K = 3$ (meaning parts of size > 3 are anchors, parts of size ≤ 3 are not), we have $N = 1$ anchor (the first part).

Let $N \ge 0$ be an integer. For given parameters M, N, K, L, we will enumerate the number of arrangements of error positions that result in exactly N anchors. This is equivalent to the combinatorial problem of counting **integer compositions** of $L+1$ with certain restrictions on the sizes of the parts. We will use generating function techniques from combinatorics to count arrangements of M error positions that give exactly N anchors (so the other $N+1-M$ parts are short fragments). Let $c_{M,N,K}(L)$ denote the number of arrangements of M error positions that result in exactly N anchors, where the read length is L and anchors are defined as parts $\lambda_i > K$. Let $c'_{M,N,K}(L)$ be the number of arrangements where all the anchors precede all the short parts $(\lambda_1, \ldots, \lambda_N > K)$ and $\lambda_{N+1}, \ldots, \lambda_{M+1} \le K)$. These are related by $c_{M,N,K}(L) = \binom{M+1}{N} c'_{M,N,K}(L)$ since we can select any N of the $M + 1$ parts to be the anchors.

Note that

$$\text{NumConfigurations}(M, N, K, L) = \sum_{N' \ge N} c_{M,N',K}(L) \tag{A1}$$

The compositions of $L + 1$ into $M + 1$ parts, where the first N parts are anchors and the remaining $M + 1 - N$ parts are short, have the following constraints:

- $\lambda_1, \ldots, \lambda_N \in 1, 2, \ldots, K$ (short parts).
- $\lambda_{N+1}, \lambda_{N+2}, \ldots, \lambda_M \in K + 1, K + 2, \ldots$ (anchors).
- $\lambda_1 + \cdots + \lambda_M = L + 1$.

The generating functions for short parts, $S(t)$, and anchors, $A(t)$, are

$$S(t) = t^1 + t^2 + \cdots + t^K = \frac{t(1 - t^K)}{1 - t} \tag{A2}$$

$$A(t) = t^{K+1} + t^{K+2} + \cdots = \frac{t^{K+1}}{1 - t} \tag{A3}$$

Standard methods for enumerating compositions with generating functions give that

$$A(t)^N S(t)^{M+1-N} = \sum_L c'_{M,N,K}(L) t^{L+1} \tag{A4}$$

where we expand the left side in a MacLaurin series (Taylor series centered at $t = 0$) to obtain the right side. The counts $c'_{M,N,K}(L)$ are the coefficients in this series. Multiplying this by $\binom{M+1}{N}$ gives

$$\begin{aligned} G_{M,N,K}(t) &= \binom{M+1}{N} A(t)^N S(t)^{M+1-N} \\ &= \sum_L c_{M,N,K}(L) t^{L+1} \end{aligned} \tag{A5}$$

To compute $c_{M,N,K}(L)$, we use Taylor series methods to compute the coefficient of t^{L+1} in (A5). We present two methods to do this.

First Taylor series method: The coefficient of t^{L+1} in (A5) may be determined by polynomial multiplication. We truncate the middle expression in (A3) to terms of degree $\le L+1$, which turns it into a polynomial; the middle expression of (A2) is already a polynomial. We take powers and products of the polynomials, truncating terms of degree $> L+1$ at intermediate steps. The coefficient of t^{L+1} in the result is $c_{M,N,K}(L)$. All intermediate products and sums involve only nonnegative integers.

Second Taylor series method: We present an exact closed-form solution. Mathematically, closed-form solutions are usually preferred. However, the first method above may be preferable for computation because intermediate steps of this second method require much higher precision, as discussed in Appendix 2.

Theorem A1. *For $K = 0$: if $N = M+1$ then $c_{M,N,K}(L) = \binom{L}{M}$; otherwise, $c_{M,N,K}(L) = 0$.*

For $K \ge 1$, set $D = L - NK - M$ and $i_{\max} = \min(\lfloor D/K \rfloor, M+1-N)$.

If $D < 0$ or $i_{\max} < 0$ then $c_{M,N,K}(L) = 0$. Otherwise,

$$c_{M,N,K}(L) = \binom{M+1}{N} \sum_{i=0}^{i_{\max}} (-1)^i \binom{M+1-N}{i} \binom{M+D-iK}{M}. \tag{A6}$$

Proof. For $K = 0$, there are no short parts; all parts are anchors. This is equivalent to counting the number of strict compositions of $L+1$ into $M+1$ parts, which is well-known to be $\binom{L}{M}$.

For $K \ge 1$, note that we may write

$$\begin{aligned} G_{M,N,K}(t) &= \binom{M+1}{N} A(t)^N S(t)^{M+1-N} \\ &= \binom{M+1}{N} \frac{t^{KN+M+1}(1-t^K)^{M+1-N}}{(1-t)^{M+1}} \end{aligned} \tag{A7}$$

The binomial theorem and the negative binomial series give

$$(1-t^K)^{M+1-N} = \sum_{i=0}^{M+1-N} (-1)^i \binom{M+1-N}{i} t^{iK}$$

$$\frac{1}{(1-t)^{M+1}} = \sum_{j=0}^{\infty} \binom{M+j}{M} t^j$$

Plugging these into (A7), we obtain

$$\begin{aligned} G_{M,N,K}(t) = \binom{M+1}{N} \sum_{i=0}^{M+1-N} \sum_{j=0}^{\infty} \binom{M+j}{M} (-1)^i \\ \times \binom{M+1-N}{i} t^{KN+(M+1)+iK+j} \end{aligned} \tag{A8}$$

In (A5), the coefficient of t^{L+1} is $c_{M,N,K}(L)$. Collecting together the terms in (A8) where the exponent of t is $L+1$ gives (A6). We omit the detailed but straightforward derivation. □

Appendix 2

Numerical precision of the closed form solution for the number of anchors

Theorem A1 (also called the "Second Taylor series method") gives a closed form expression (A6) to compute $c_{M,N,K}(L)$. This closed form solution has only a small number of terms. However, for practical parameter values, it may require more bits of precision than are available in a finite precision computation, even if the final answer does not overflow the variable size. This is because the expression has an alternating sum with terms of much higher absolute value than the final answer. Consider this part of the summation in Theorem A1, omitting the coefficient $\binom{M+1}{N}$:

$$\sum_{i=0}^{i_{\max}} (-1)^i \binom{M+1-N}{i} \binom{M+D-iK}{M}.$$

For $M = 75$, $N = 1$, $K = 15$, $L = 1000$, this has 61 alternating terms of magnitude between 2^{93} and 2^{401}, while the value of the sum is much smaller, with magnitude 2^{294}. Using high precision floating point, we need at least 110 bits for the mantissa to get the first decimal digit correct. This is significantly more bits than is currently standard: the current standard for floating point, IEEE 754, provides for a 53 bit mantissa in double precision. Alternatively, using high precision integers, we would need 294 bits of integer precision, plus a sign bit. However, software for arbitrary precision integers, such as Maple or Mathematica, will handle this example correctly.

By contrast, the "First Taylor series method" only involves sums and products of positive integers, each

bounded above by the value of $c_{M,N,K}(L)$. Thus, if the integer precision is adequate to store the value of $c_{M,N,K}(L)$, it is also adequate to perform all intermediate calculations.

Appendix 3

Statistics of number of anchors

We may estimate the number of anchors using the following theorem.

Theorem A2. *Fix M, K, L. Under the uniform distribution on compositions of $L+1$ into $M+1$ parts, the mean number of anchors and its variance are given by*

$$\mu = E[N] = \frac{(M+1)\binom{L-K}{M}}{\binom{L}{M}} \tag{A9}$$

$$\sigma^2 = \mathrm{Var}[N] = \frac{M(M+1)\binom{L-2K}{M}}{\binom{L}{M}} + \frac{(M+1)\binom{L-K}{M}}{\binom{L}{M}} - \left(\frac{(M+1)\binom{L-K}{M}}{\binom{L}{M}}\right)^2. \tag{A10}$$

For fixed M and K, consider the two-variable generating function

$$\begin{aligned} H_{M,K}(t,u) &= \sum_{N=0}^{\infty}\sum_{L} c_{M,N,K}(L)\, t^{L+1} u^N \\ &= \sum_{N=0}^{\infty} G_{M,N,K}(t)\, u^N \qquad \text{(A11)} \\ &= \sum_{N=0}^{\infty} \binom{M+1}{N} \left(\frac{t^{K+1}}{1-t}\right)^N u^N \\ &\quad \times \left(\frac{t(1-t^K)}{1-t}\right)^{M+1-N} \\ &= \left(\frac{u\,t^{K+1}}{1-t} + \frac{t(1-t^K)}{1-t}\right)^{M+1} \\ &= \left(\frac{(u-1)t^{K+1}+t}{1-t}\right)^{M+1} \qquad \text{(A12)} \end{aligned}$$

For fixed M, K, L, the probability of exactly N anchors is $c_{M,N,K}(L)/T$, where

$$T = \sum_{N'=0}^{\infty} c_{M,N',K}(L)$$

Note that T counts the total number of compositions of $L+1$ into $M+1$ parts, with 0 or more anchors. Thus, it actually counts the total number of compositions of $L+1$ into $M+1$ parts, without regard to sizes of parts. So we have:

$$T = \sum_{N'=0}^{\infty} c_{M,N',K}(L) = \binom{L}{M} \tag{A13}$$

and thus the probability of exactly N anchors is $c_{M,N,K}(L)/\binom{L}{M}$.

Next, for fixed M, K, L, we evaluate $E[N]$, the mean number of anchors under the uniform distribution of compositions of $L+1$ into $M+1$ parts.

$$E[N] = \sum_{N} \frac{c_{M,N,K}(L)}{\binom{L}{M}} \cdot N$$

Using standard generating function properties, the numerator $\sum_N c_{M,N,K}(L) \cdot N$ is the coefficient of t^{L+1} in the following expression:

$$\frac{\partial}{\partial u} H_{M,K}(t,u)\bigg|_{u=1} \tag{A14}$$

First we evaluate the derivative; second, we plug in $u = 1$; third, we extract the coefficient of t^{L+1}; and fourth, we use this to compute $E[N]$:

1. The derivative in Eq. (A14) is

$$\begin{aligned} \frac{\partial}{\partial u} H_{M,K}(t,u) &= \frac{\partial}{\partial u}\left(\frac{(u-1)t^{K+1}+t}{1-t}\right)^{M+1} \\ &= (M+1)\left(\frac{(u-1)t^{K+1}+t}{1-t}\right)^{M} \frac{t^{K+1}}{1-t} \end{aligned}$$

2. Plug in $u = 1$:

$$\begin{aligned} \frac{\partial}{\partial u} H_{M,K}(t,u)\bigg|_{u=1} &= (M+1)\left(\frac{t}{1-t}\right)^{M} \frac{t^{K+1}}{1-t} \\ &= (M+1)\frac{t^{M+K+1}}{(1-t)^{M+1}} \end{aligned}$$

3. Expand the Taylor series and extract the coefficient of t^{L+1}:

$$(M+1)\frac{t^{M+K+1}}{(1-t)^{M+1}} = (M+1)\sum_{j=0}^{\infty}\binom{M+j}{M} t^{M+K+1+j}$$

The term t^{L+1} occurs when $j = L-M-K$.
If $j < 0$, this coefficient is 0. If $j \geq 0$, this coefficient is $(M+1)\binom{L-K}{M}$.

4. Evaluate $E[N]$ to obtain Equation (A9):

$$E[N] = \frac{(M+1)\binom{L-K}{M}}{\binom{L}{M}}.$$

Note that if $L - K < M$, then $E[N] = 0$.

Next we compute the variance of N, using a similar generating function technique. The generating function will enable us to compute $E[N(N-1)]$, so we will compute the variance in the form

$$\sigma^2 = E[N(N-1)] + E[N] - E[N]^2$$

which is equivalent to the more common formula $\sigma^2 = E[N^2] - E[N]^2$. We have:

$$E[N(N-1)] = \sum_{N} \frac{c_{M,N,K}(L)}{\binom{L}{M}} \cdot N(N-1)$$

The numerator $\sum_N c_{M,N,K}(L) \cdot N(N-1)$ is the coefficient of t^{L+1} in the following expression:

$$\frac{\partial^2}{\partial u^2} H_{M,K}(t,u)\Big|_{u=1} \tag{A15}$$

We evaluate this in a fashion similar to $E[N]$:

1. The derivative in Eq. (A15) is

$$\begin{aligned}\frac{\partial^2}{\partial u^2} H_{M,K}(t,u) &= \frac{\partial^2}{\partial u^2}\left(\frac{(u-1)t^{K+1}+t}{1-t}\right)^{M+1} \\ &= M(M+1)\left(\frac{(u-1)t^{K+1}+t}{1-t}\right)^{M-1} \\ &\quad \times \left(\frac{t^{K+1}}{1-t}\right)^2\end{aligned}$$

2. Plug in $u = 1$:

$$\begin{aligned}\frac{\partial^2}{\partial u^2} H_{M,K}(t,u)\Big|_{u=1} &= M(M+1)\left(\frac{t}{1-t}\right)^{M-1}\left(\frac{t^{K+1}}{-t}\right)^2 \\ &= M(M+1)\frac{t^{M+1+2K}}{(1-t)^{M+1}}\end{aligned}$$

3. Expand the Taylor series and extract the coefficient of t^{L+1}:

$$\begin{aligned}&M(M+1)\frac{t^{M+1+2K}}{(1-t)^{M+1}} \\ &= M(M+1)t^{M+1+2K}\sum_{j=0}^{\infty}\binom{M+j}{M}t^j \\ &= M(M+1)\sum_{j=0}^{\infty}\binom{M+j}{M}t^{M+1+2K+j}\end{aligned}$$

The term t^{L+1} occurs when $j = L-(M+2K)$. This coefficient is $M(M+1)\binom{L-2K}{M}$ (which is 0 when $L-2K < M$).

4. Evaluate $E[N(N-1)]$:

$$E[N(N-1)] = \frac{M(M+1)\binom{L-2K}{M}}{\binom{L}{M}}$$

5. Evaluate $\sigma^2 = \mathrm{Var}[N]$ to prove Equation (A10):

$$\begin{aligned}\sigma^2 = \mathrm{Var}[N] &= E[N(N-1)] + E[N] - E[N]^2 \\ &= \frac{M(M+1)\binom{L-2K}{M}}{\binom{L}{M}} + \frac{(M+1)\binom{L-K}{M}}{\binom{L}{M}} \\ &\quad - \left(\frac{(M+1)\binom{L-K}{M}}{\binom{L}{M}}\right)^2\end{aligned}$$

Appendix 4

Asymptotic number of anchors

Theorem A3. *Let μ, σ^2 be given by Theorem A2. For sufficiently large M,*

$$c_{M,N,K}(L) \approx \binom{L}{M}\phi\left(\frac{N-\mu}{\sigma}\right) \tag{A16}$$

$$\mathrm{NumConfigurations}(M,N,K,L) \approx \binom{L}{M} \times \left(1-\Phi\left(\frac{N-\frac{1}{2}-\mu}{\sigma}\right)\right) \tag{A17}$$

where $\phi(z) = \frac{1}{\sqrt{2\pi}}e^{-z^2/2}$ and $\Phi(z)$ are the probability density function and cumulative distribution function of the standard normal distribution.

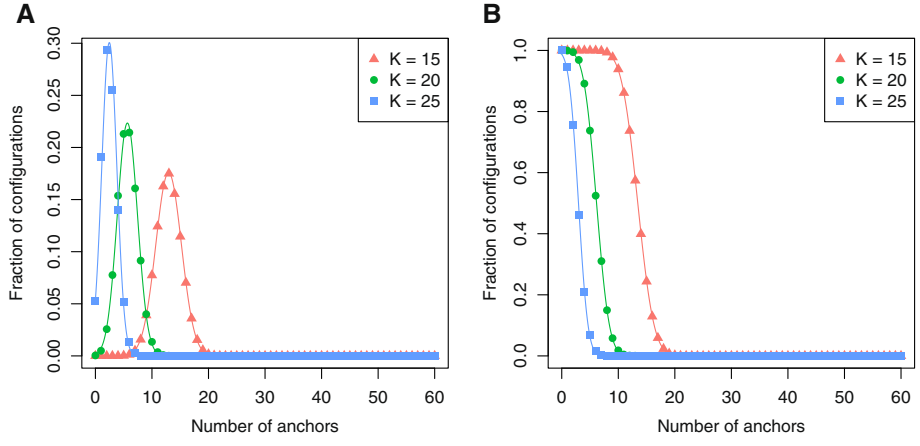


Figure 11 The fraction of configurations with exactly and at least *N* anchors. (**A**) Plot of the fraction of configurations with exactly N anchors, $c_{M,N,K}(L)/\binom{L}{M}$, as N varies. An anchor is a run of at least K correct bases (shown for $K = 15$, 20, and 25). We assume the read length is $L = 1000$ and the error rate per base is $\rho = 15\%$ (and that there are exactly $M = 150$ error positions). The solid markers are computed by finding exact coefficients $c_{M,N,K}(L)$ in the generating functions. The curve is a normal distribution approximating the exact values (illustrating Theorem A3), where parameters μ and σ^2 are computed by Theorem A2. (**B**) The solid markers are a plot of $\mathrm{NumConfigurations}(M,N,K,L)/\binom{L}{M}$, the fraction of configurations with at least N anchors, as N varies. The parameters are the same as for (A). The curve is the survival function of the normal distribution in (A).

Proof. For fixed M, K, Eq. (A12) gives the generating function $H_{M,K}(t,u)$ as a rational function in t, u, raised to the power $M+1$. When M is sufficiently large, the Central Limit Theorem gives that the coefficients $c_{M,N,K}(L)$ in its Taylor series, Eq. (A11), are well-approximated by a bivariate normal distribution. Restricting to the coefficients of t^{L+1} for fixed L gives that the coefficients of $f(u) = \sum_{N=0}^{\infty} c_{M,N,K}(L) u^N$ are approximated by a univariate normal distribution. This represents the distribution of N for fixed M, K, L. We computed the parameters μ, σ^2 of this distribution in Theorem A2. By Eq. (A13), the total of the coefficients in $f(u)$ is $f(1) = T = \binom{L}{M}$. Note that NumConfigurations(M,N,K,L) is the survival function of $c_{M,N,K}(L)$:

$$\text{NumConfigurations}(M,N,K,L) = \sum_{N' \geq N} c_{M,N',K}(L)\,.$$

Thus, we obtain Eqs. (A16) and (A17) as approximations for the coefficients $c_{M,N,K}(L)$ and the survival function NumConfigurations(M,N,K,L). In Eq. (A17), note that $-\frac{1}{2}$ is a continuity correction. □

In Figure 11, we plot $c_{M,N,K}(L)$ and NumConfigurations(M,N,K,L). The solid markers are the true values computed from the generating function. The curve is the estimate computed by the preceding theorem, and does indeed approximate the true values well.

Additional files

Additional file 1: Supplementary Text S1. The supplementary text contains additional implementation details for the anchor similarity method, and description of the empirical model based read simulator.

Additional file 2: Supplementary Table S1. Supplementary Table S1 gives the command line parameters used to run the benchmarks.

Competing interests
MJC is a full-time employee at Pacific Biosciences, a company commercializing single-molecule, real-time nucleic acid sequencing technologies. GT was partially supported by a grant from the National Institutes of Health, USA (NIH grant 3P41RR024851-02S1). NIH had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Author's contributions
MJC proposed and implemented the mapping method, performed the analysis, and wrote the manuscript. GT solved and implemented the combinatorial analysis, and wrote the manuscript. Both authors read and approved the final manuscript.

Acknowledgements
We thank Jon Sorenson, James Bullard, Eric Schadt, and Jonas Korlach for useful comments in writing this manuscript.

Author details
[1]Department Secondary Analysis, Pacific Biosciences, 1005 Hamilton Rd, Menlo Park, CA, USA. [2]Department of Mathematics, University of California, San Diego, 9500 Gilman Dr, La Jolla, CA, USA.

Received: 12 March 2012 Accepted: 17 September 2012
Published: 19 September 2012

References

1. Smith T, Waterman M: **Identification of Common Molecular Subsequences.** *J Mol Biol* 1981, **147:**194–197.
2. Zhang Z, Schwartz S, Wagner L, Miller W: **A greedy algorithm for aligning DNA sequences.** *J Comput Biol* 2000, **7:**203–214.
3. Kent W: **BLAT–the BLAST-like alignment tool.** *Genome Res* 2002, **12:**656–664.
4. Langmead B, Trapnell C, Pop M, Salzberg S: **Ultrafast and memory-efficient alignment of short DNA sequences to the human genome.** *Genome Biol* 2009, **10:**R25.
5. Li H, Durbin R: **Fast and accurate short read alignment with Burrows-Wheeler transform.** *Bioinformatics* 2009, **25:**1754–1760.
6. Li R, Yu C, Li Y, Lam T, Yiu S, Kristiansen K, Wang J: **SOAP2: an improved ultrafast tool for short read alignment.** *Bioinformatics* 2009, **25:**1966–1967.
7. Ferragina P, Manzini G: **Opportunistic data structures with applications.** In *Proc. of the 41st IEEE Symp on Found of Comput Sci*; 2000:390–398.
8. Rasko D, Webster D, Sahl J, Bashir A, Boisen N, Scheutz F, Paxinos E, Sebra R, Chin C, Iliopoulos D, Klammer A, Peluso P, Lee L, Kislyuk A, Bullard J, Kasarskis A, Wang S, Eid J, Rank D, Redman J, Steyert S, Frimodt-Mller J, Struve C, Petersen A, Krogfelt K, Nataro J, Schadt E, Waldor M: **Origins of the E. coli strain causing an outbreak of hemolytic-uremic syndrome in Germany.** *New England J Med* 2011, **365:**709–717.
9. Brudno M, Poliakov A, Salamov A, Cooper G, Sidow A, Rubin E, Solovyev V, Batzoglou S, Dubchak I: **Automated Whole-Genome Multiple Alignment of Rat, Mouse, and Human.** *Genome Res* 2004, **14:**685–692.
10. Schwartz S, Kent W, Smit A, Zhang Z, Baertsch R, Hardison R, Haussler D, Miller: **Human-Mouse Alignments with BLASTZ.** *Genome Res* 2003, **13:**103–107.
11. Kurtz S, Phillippy A, Delcher A, Smoot M, Shumway M, Antonescu C, Salzberg S: **Versatile and open software for comparing large genomes.** *Genome Biol* 2004, **5:**R122–R129.
12. Altschul S, Gish W, Miller W, Myers E, Lipman D: **Basic local alignment search tool.** *J Mol Biol* 1990, **215:**403–410.
13. Lipman D, Pearson W: **Rapid and sensitive protein similarity searches.** *Science* 1985, **4693:**1435–1441.
14. Bray N, Dubchak I, Pachter L: **AVID: A global alignment program.** *Genome Res* 2003, **13:**97–102.
15. Darling A, Mau B, Blatter F, Perna N: **Mauve: multiple alignment of conserved genomic sequence with rearrangements.** *Genome Res* 2004, **14:**1394–1403.
16. Brudno M, Do C, Cooper G, Kim M, Davydov E, Green E, Sidow A, Batzoglou S: **LAGAN and Multi-LAGAN: efficient tools for large-scale multiple alignment of genomic DNA.** *Genome Res* 2003, **13:**721–731.
17. Kent WJ, Baertsch R, Hinrichs A, Miller W, Haussler D: **Evolution's cauldron: Duplication, deletion, and rearrangement in the mouse and human genomes.** *Proc Nat Acad Sci* 2003, **100:**11484–11489.
18. Li H, Durbin R: **Fast and accurate long-read alignment with Burrows Wheeler transform.** *Bioinformatics* 2010, **26:**589–595.
19. Li R, Li Y, Kristiansen K, Wang J: **SOAP: short oligonucleotide alignment program.** *Bioinformatics* 2008, **24:**713–714.
20. Li H, Ruan J, R D: **Mapping short DNA sequencing reads and calling variants using mapping quality scores.** *Genome Res* 2008, **18:**1851–1858.
21. Rumble S, Lacroute P, Dalca A, Fiume M, Sidow A, et al.: **SHRiMP: Accurate Mapping of Short Color-space Reads.** *PLoS Comput Biol* 2009, **5:**e1000386.
22. Durbin R, Eddy S, Krobh A, Mitchison G: *Biol Sequence Anal: Probabilistic Models of Proteins and Nucleic Acids. The Edinburgh Building, Cambridge, CB2 2RU*. United Kingdom: Cambridge University Press; 1998.
23. Needleman S, Wunsch C: **A general method applicable to the search for similarities in the amino acid sequence of two proteins.** *J Mol Biol* 1970, **48:**443–453.
24. Eid J, Fehr A, Korlach J, Turner S, et al.: **Real-time DNA sequencing from single polymerase molecules.** *Science* 2009, **323:**133–138.
25. Clarke J, Wu H, Jayasinghe L, Patel A, Reid S, Bayley H: **Continuous base identification for single-molecule nanopore DNA sequencing.** *Nat Nanotechnol* 2009, **4:**265–270.

26. Cherf G, Lieberman K, Rashid H, Lam C, Karplus K, Akeson M: **Automated forward and reverse ratcheting of DNA in a nanopore at 5-Å precision.** *Nat Nanotechnol* 2012, **14:**344–348.
27. Garaj S, Hubbard W, Reina A, Kong J, Branton D, Golovchenko J: **Graphene as a subnanometre trans-electrode membrane.** *Nature* 2010, **467:**190–193.
28. Altschul S, Madden T, Schäffer A, Zhang J, Zhang Z, Miller W, Lipman D: **Gapped BLAST and PSI-BLAST: a new generation of protein database search programs.** *Nucleic Acids Res* 1997, **17:**3389–3402.
29. Weese D, Emde A, Rausch T, Döring A, Reinert K: **RazerS-fast read mapping with sensitivity control.** *Genome Res* 2009, **19:**1646–1654.
30. Bao Z, Eddy S: **Automated de novo identification of repeat sequence families in sequenced genomes.** *Genome Res* 2002, **12:**1269–1276.
31. Flusberg B, Webster D, Lee J, Travers K, Olivares E, Clark T, Korlach J, SW T: **Direct detection of DNA methylation during single-molecule, real-time sequencing.** *Nat Methods* 2010, **7:**461–465.
32. Alkan C, Kidd J, Marques-Bonet T, Aksay G, Antonacci F, Hormozdiari F, Kitzman J, Baker C, Malig M, Mutlu O, Sahinalp S, Gibbs R, Eichler E: **Personalized Copy-Number and Segmental Duplication Maps using Next-Generation Sequencing.** *Nat Genet* 2009, **41:**1061–1067.
33. Myers E, Manber U: **Suffix arrays: A new method for on-line string searches.** *SIAM J Comput* 1993, **22:**935–948.
34. Abouelhoda M, Ohlebusch E: **A Local Chaining Algorithm and its Applications in Comparitive Genomics.** *Lecture Notes in Comput Sci* 2812, **2003:**1–16.
35. Eppstein D, Galil Z, Giancarlo R, Italiano G: **Sparse Dynamic Programming I: Linear cost functions.** *J Assoc Comput Machinery* 1992, **39:**519–545.

doi:10.1186/1471-2105-13-238
Cite this article as: Chaisson and Tesler: Mapping single molecule sequencing reads using basic local alignment with successive refinement (BLASR): application and theory. *BMC Bioinformatics* 2012 **13**:238.