RazerS—fast read mapping with sensitivity control

David Weese, Anne-Katrin Emde, Tobias Rausch, et al.

Genome Res. 2009 19: 1646-1654 originally published online July 10, 2009
Access the most recent version at doi:10.1101/gr.088823.108

Resource

RazerS—fast read mapping with sensitivity control

David Weese,[1,3] Anne-Katrin Emde,[1] Tobias Rausch,[2] Andreas Döring,[1] and Knut Reinert[1]

[1]*Department of Computer Science, Free University of Berlin, 14195 Berlin, Germany;* [2]*International Max Planck Research School for Computational Biology and Scientific Computing, 14195 Berlin, Germany*

Second-generation sequencing technologies deliver DNA sequence data at unprecedented high throughput. Common to most biological applications is a mapping of the reads to an almost identical or highly similar reference genome. Due to the large amounts of data, efficient algorithms and implementations are crucial for this task. We present an efficient read mapping tool called RazerS. It allows the user to align sequencing reads of arbitrary length using either the Hamming distance or the edit distance. Our tool can work either lossless or with a user-defined loss rate at higher speeds. Given the loss rate, we present an approach that guarantees not to lose more reads than specified. This enables the user to adapt to the problem at hand and provides a seamless tradeoff between sensitivity and running time.

[RazerS is freely available at http://www.seqan.de/projects/razers.html.]

Second-generation sequencing technologies are revolutionizing the field of DNA sequence analysis, as large amounts of sequencing data can be obtained at increasing rates and dramatically decreasing costs. Biological applications are manifold, including whole-genome resequencing for the detection of genomic variation, e.g., single nucleotide polymorphisms (SNPs) (Bentley et al. 2008; Hillier et al. 2008; Ley et al. 2008; Wang et al. 2008) or large structural variations (Chen et al. 2008), RNA sequencing for small noncoding RNA discovery or expression profiling (Morin et al. 2008), metagenomics applications (Huson et al. 2007), and sequencing of chromatin-immunoprecipitated DNA, e.g., for the identification of DNA binding sites and histone modification patterns (Barski et al. 2007).

Fundamental to all these applications is the problem of mapping all sequenced reads against a reference genome, denoted as the *read mapping problem*. It can be formalized as follows: given a set of read sequences $\mathcal{R}$, a reference sequence G, and a distance $k \in \mathbb{N}$, find all substrings g of G that are within distance k to a read $r \in \mathcal{R}$. The occurrences of g in G are called *matches*. Common distance measures are Hamming distance or edit distance; the former forbidding insertions and deletions (i.e., indels) in the alignment, the latter allowing for mismatches and indels alike.

As the new sequencing technologies are able to produce millions of reads per run, efficient algorithms for read mapping are necessary. Reads are typically quite short compared to the traditional Sanger reads and have specific error distributions depending on the technology used.

A variety of tools have been designed and developed specifically for the purpose of mapping short reads. A compilation of some popular tools is shown in Table 1 together with some key features of the algorithms.

Most of the existing read mapping approaches use a two-step strategy. First, a filtration algorithm is applied in order to identify candidate regions that possibly contain a match. This includes building an index data structure, either on the set of reads or on the reference sequence. Second, candidate regions are examined for true matches in a more time-consuming verification step. In current implementations one has to carefully distinguish whether both steps, the filtration step and the verification step, are adequate for the distance chosen (Hamming or edit distance). Some implementations, for instance, verify matches using base-call qualities, but filter the candidate regions using a fixed Hamming or edit distance (H Li et al. 2008). Filtration methods in use are based on single (Kent 2002; Ma et al. 2002) or multiple seeds (Li et al. 2003; Lin et al. 2008), the pigeonhole principle (Navarro and Raffinot 2002; H Li et al. 2008; R Li et al. 2008; AJ Cox, ELAND: Efficient local alignment of nucleotide data, unpubl.), or based on counting lemmas using (gapped) q-gram (Burkhardt et al. 1999; Rasmussen et al. 2006; Rumble and Brudno 2008). Verification methods encompass semiglobal alignment algorithms for Hamming or edit distance (Levenshtein 1966) or local-alignment algorithms (Smith and Waterman 1981).

BLAT (Kent 2002), as an example of a single seed filter, searches exact occurrences of short fixed sized substrings shared by two sequences. PatternHunter (Ma et al. 2002) was the first to generalize this strategy to gapped seeds (common discontiguous subsequences), thereby increasing sensitivity while maintaining specificity. Further sensitivity is achieved by using multiple gapped seeds; an approach implemented in the read mapping tool Zoom (Lin et al. 2008), which uses a restricted version of edit distance with at most one gap. After the initial submission of this paper a method using whole reads as seeds was published, which tolerates a small number of mismatches by backtracking all possible replacements of low-quality bases (Langmead et al. 2009). It uses Burrows-Wheeler transformed genomes and is an efficient approach to short read mapping.

Given two sequences within distance k, the pigeonhole lemma states that of any partition of the first sequence into $k + 1$ parts, at least one part must be found without errors in the other sequence. The shorter this seed, the more likely it is to encounter random matches, and therefore the lower the specificity of the filter. This strategy is thus rather limited and quickly grows impractical with an increasing number of errors. In an extension of this strategy the first sequence is divided into $k + 2$ parts. Now at least two of such parts will occur in the other sequence. These two parts retain their relative positions as long as no indels occur in between. ELAND (AJ Cox, unpubl.), MAQ (H Li et al. 2008), and

[3]Corresponding author.
E-mail weese@inf.fu-berlin.de; fax 49-30-83875218.
Article published online before print. Article and publication date are at http://www.genome.org/cgi/doi/10.1101/gr.088823.108.

19:1646–1654 © 2009 by Cold Spring Harbor Laboratory Press; ISSN 1088-9051/09; www.genome.org

Table 1. Short read mapping tools with their characteristics

	ELAND	MAQ	SOAP	SeqMap	Zoom	SHRiMP	RazerS
Filtering technique	Two-seed pigeonhole	Two-seed pigeonhole	Two-seed pigeonhole	Two-seed pigeonhole	Multiple gapped seeds	*q*-gram counting	*q*-gram counting
Distance measure in filtering step	Hamming	Hamming	Hamming	Both	Hamming	Both	Both
Distance measure in mapping step	Hamming	Hamming (Smith-Waterman for second mate)	Hamming (optionally with one gap)	Hamming or edit with at most five errors	Hamming or edit with at most one gap	Smith-Waterman	Either edit or Hamming
Supported read length	≤32	≤127	≤60	Arbitrary	≤63	Arbitrary	Arbitrary
Sensitivity	Full sensitivity only for up to two errors	Full sensitivity	Depends on setting, no switch to guarantee full sensitivity	Full sensitivity	Switch to guarantee full sensitivity	No help for parameter choice, default will be lossy for most settings	Arbitrarily adjustable
Can output all (suboptimal) hits	No	No	No	Yes	Yes	Yes	Yes

SOAP (R Li et al. 2008) make use of this observation but are therefore limited to Hamming distance. Furthermore, ELAND and SOAP always use a four-segment partition and MAQ at most a five-segment partition and can therefore not guarantee full sensitivity for $k > 2$ or $k > 3$, respectively. SeqMap (Jiang and Wong 2008) extends the two-seed pigeonhole strategy to edit distance and searches the two parts varying the gap length by $-k,\ldots, k$ nucleotides.

Another approach is the *q*-gram counting strategy which was first used in QUASAR (Burkhardt et al. 1999). It uses the *q*-gram lemma (Owolabi and McGregor 1988; Jokinen and Ukkonen 1991), which states that two sequences of length n with Hamming distance k share at least $t = n + 1 - (k + 1)q$ common substrings of length q, so-called *q*-grams. This lemma can also be applied to edit distance if n is the length of the larger sequence. A generalization for gapped *q*-grams was given by Burkhardt and Kärkkäinen (2002). SHRiMP (Rumble and Brudno 2008) employs a *q*-gram counting strategy with a default configuration that does not guarantee to be lossless.

In this work we present an implementation of a versatile read mapper RazerS based on *q*-gram counting, which distinguishes itself in several respects from existing algorithms. First, it can map reads using edit or Hamming distance in the filtering phase and in the verification phase without any restrictions. Second, given a user-defined loss rate (possibly 0 making the mapper exact), we developed an algorithm to select parameters such that the chosen loss rate is not exceeded in expectation. The algorithm depends on an error model derived from base-call quality values. Finally, our implementation can map reads of any length with an arbitrary number of errors and is currently the fastest in reporting all hits for typical read lengths and loss rates.

Methods

Definitions and notation

We consider strings over the finite ordered alphabet Σ. Σ^* is the set of all possible strings over Σ and ϵ denotes the empty string. A string s is a sequence of letters $s[1]\ldots s[n]$, where each $s[i] \in \Sigma$; st is the concatenation of two strings s and t; $|s|$ denotes the length of the string s; and $s[i..j]$ is a substring of s from position i to j. If $t \in \Sigma^*$ is a substring of s, we write $t \preceq s$, and $t \prec s$ if $t \neq s$ holds in addition.

A(n) *(edit) transcript* is a string over the alphabet $\Delta = \{M, R, D, I\}$ that describes a transformation from one string to another, see Figure 1. For two strings $r,g \in \Sigma^*$, a transcript from r to g is read and applied from left to right to single characters of r to produce g, whereby M, R, D, and I correspond to a match (no change), a replacement, a deletion, and an insertion of a single character in r, respectively. There is a one-to-one relation between alignments and transcripts. For any transcript T we define $||T||_E = |\{i|T[i] \in \{R, D, I\}\}|$, the number of errors in T. The edit distance of two strings is the minimum number of errors in all transcripts between these strings. A special case is the Hamming transcript with $\boldsymbol{\Delta} = \{M, R\}$. It is defined uniquely for two strings of equal length and the Hamming distance is the number of errors in it.

In the following we consider transcripts from reads $r \in \mathcal{R}$ to substrings g of a genome G. We say that the pair (r,g) is a *true match* if $d(r,g) \leq k$, for d being either edit or Hamming distance. A substring M^q of a transcript T is called a *q-match*. If a transcript from r to g contains t q-matches, then r and g have at least t common q-grams. A (q,t)-*filter* is an algorithm that detects any pair (r,g) for which a transcript T from r to g with $\geq t$ q-matches exists.

Sensitivity calculation

In this section we devise a method to determine the sensitivity of (q,t) filters. The sensitivity of a filter is the probability that a randomly chosen *true* match (r,g) is classified by the filter as *potential* match. Existing sensitivity estimation approaches are limited to multiple seed filters (Li et al. 2003) or assume transcripts to be generated by a Markov process (Herms and Rahmann 2008). Our method estimates the filtration sensitivity under any

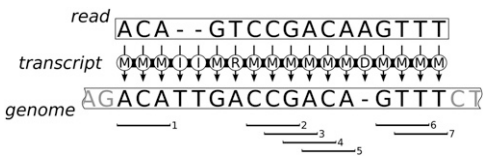

Figure 1. A transcript from a read to a part of the genome. The transcript contains seven three-matches, hence r and g share at least seven 3-gram. Actually, they also share an eighth 3-gram CAG that does not correspond to a three-match of the transcript.

position-dependent error distribution, e.g., as observed in Sanger or Illumina DNA sequencing technologies (Dohm et al. 2008).

Hamming distance sensitivity

We first consider the read mapping problem for Hamming distance and a given maximum distance k. In the following, all reads are considered to be of equal length n.

To determine a lower bound for the sensitivity of a (q,t)-filter we could enumerate all Hamming transcripts with up to k replacements and sum up the occurrence probabilities of those transcripts having at least t substrings M^q. As there are $\sum_{i=0}^{k}\binom{n}{i}$ many different transcripts, a full enumeration takes $\Omega[(n/k)^k]$ time and is not feasible for large reads or high error rates, e.g., of length $n = 200$ with $k = 20$ errors. We have developed a dynamic programming approach that is significantly faster by using a recurrence similar to the threshold calculation in Burkhardt and Kärkkäinen (2002).

Assume a given error distribution that associates each nucleotide position i in a read with an error probability P_i^{R}, e.g., the probability of a base miscall during sequencing or a SNP. Then, the occurrence probability of a Hamming transcript T is the product of the occurrence probabilities of the single transcript characters at each position $p(T) = \prod_{i=1,\dots,|T|} p_i^{T[i]}$, with $P_i^{\mathrm{M}} = 1 - p_i^{\mathrm{R}}$. We calculate the sensitivities for matches with e errors for each $e \le k$ separately. Let $S(n, e, t)$ be the sum of occurrence probabilities of transcripts of length n, having e errors, and at least t q-matches. The sensitivity of a (q,t)-filter to detect e-error matches is at least

$$P(T \text{ contains} \ge t\ q\text{-matches} \mid \|T\|_{\mathrm{E}} = e) = \frac{S(n,e,t)}{S(n,e,0)}.$$

We will see, how to calculate $S(n, e, t)$ using a DP algorithm. Let $p(T, j) = \prod_{i=1,\dots,|T|} p_{i+j}^{T[i]}$ be the occurrence probability of subtranscript T to occur after j letters of a read. We define $R(i, e, T_2)$ as the sum of occurrence probabilities of transcripts $T_i \in \Delta^I$, so that T_1 has e errors and the concatenation T_1T_2 contains at least t substrings M^q. By definition of S holds:

$$S(n,e,t) = \sum_{T \in \Delta^q, \|T\|_E \le e} R(n-q, e-\|T\|_{\mathrm{E}}, t, T)\, p(T, n-q). \qquad (1)$$

The sum goes over all transcript ends T of length q with at most e errors. The right factor is the probability of T occurring at the end of a random transcript of length n. The left factor is the occurrence probability sum over all transcript beginnings, so that the concatenation of beginning and end is a transcript of length n with e errors, and at least t q-matches. With the following lemma a DP algorithm can be devised to determine R and therefore the sensitivities $S(n, e, t)$ for all $e = 0, \dots, k$ and $t = 1, \dots, t_{\max}$ in $\mathcal{O}(nkt_{\max}2^q)$.

Lemma 1. *Let $i, q \in \mathbb{N}$; $e, t \in \mathbb{Z}$; $T \in \{\mathrm{M}, \mathrm{R}\}^q$. R can be calculated using the following recurrence:*

$$R(0,e,t,T) = \begin{cases} 1, & \text{if } e = 0, t \le \delta(T) \\ 0, & \text{else} \end{cases} \qquad (2)$$

$$R(i,e,t,T) = p_i^{\mathrm{M}} \cdot R(i-1, e, t-\delta(T), \mathrm{shift}(\mathrm{M}, T)) + p_i^{\mathrm{R}} \cdot R(i-1, e-1, t-\delta(T), \mathrm{shift}(\mathrm{R}, T)). \qquad (3)$$

with

$$\mathrm{shift}(x,T) = xT[1..|T|-1],\ \delta(T) = \begin{cases} 1, & \text{if } T = \mathrm{M}^q \\ 0, & \text{else} \end{cases}$$

Proof. See Appendix.

Extension to gapped shapes

A generalization of contiguous q-grams are gapped q-grams (Burkhardt and Kärkkäinen 2002), (noncontiguous) subsequences of length q. A *shape* Q is defined to be a set of non-negative integers corresponding to nongapped positions, where $0 \in Q$ is the first nongapped position. For example, the 3-gram of shape ##-# in the string GTTCA are GTC and TTA and the corresponding shape is $Q = \{0, 1, 3\}$. The *span* of Q is $\mathrm{span}(Q) = \max Q + 1$ and the *weight* of Q is the set cardinality $|Q|$. For any integer i we define $Q_i = \{i + j | j \in Q\}$. Let $Q_i = \{i_1, \dots, i_{|Q|}\}$, where $i = i_1 < i_2 < \dots < i_{|Q|}$ and s be a string. For $1 \le i \le |s| - \mathrm{span}(Q) + 1$, $s[Q_i]$ is defined to be the string $s[i_1]s[i_2]\dots s[i_{|Q|}]$ and called *Q-gram* at position i. Contiguous *q-grams* are Q-grams with the shape $Q = \{0, 1, \dots, q-1\}$.

A Q-gram $\mathrm{M}^{|Q|}$ of a Hamming transcript T is called a *Q-match*, and a *$(Q\ t)$-filter* is an algorithm that detects any pair (r,g) for which a transcript T from r to g with $\ge t$ Q-matches exists. To extend the sensitivity calculation to gapped shapes Q, the transcript T in Lemma 1 must be resized to cover the whole shape and the matching criterion in $\delta(T)$ must be adapted. This can be done by replacing q in Equation 1 by span (Q) and $\delta(T)$ in Equations 2 and 3 by

$$\delta(T) = \begin{cases} 1, & \text{if } T[Q_1] = M^{|Q|} \\ 0, & \text{else} \end{cases}.$$

For two strings of length span(Q) with the Hamming transcript $T \in \Delta^{\mathrm{span}(Q)}$, $\delta(T)$ returns 1 if and only if they share their sole Q-gram. A lemma similar to Lemma 1 can be proven analogously.

Edit distance sensitivity

We extended the sensitivity calculation to edit distance. The DP algorithm and the proof of correctness can be found in the Appendix.

The algorithm

In the following, we propose the read mapping algorithm implemented in RazerS. It consists of three parts: a parameter chooser, a filter, and a verifier.

Filtration

To find potential match regions of reads in the genome we use the SWIFT algorithm (Rasmussen et al. 2006). It incorporates the following two observations. Consider an edit transcript between two sequences with at least t q-matches. First, the dot plot of the sequences contains at least t contiguous diagonal lines of length q, so called q-hits. Second, if the transcript contains k errors then there are $k + 1$ consecutive diagonals covering at least t q-hits. In case of Hamming transcripts, there is a single diagonal completely covering all q-hits.

For any read $r \in \mathcal{R}$ each dot plot parallelogram of dimension $|r| \times (k + 1)$ with at least t q-hits contains a potential match. Instead of counting q-hits for each possible parallelogram separately, it suffices to count them in overlapping $|r| \times w$ parallelograms with $w > k + 1$ and an overlap of k, as every $|r| \times (k + 1)$ parallelogram is contained in one $|r| \times w$ parallelogram, see Figure 2.

If for $i,j \in \mathbb{N}$ holds $r[i..i + q - 1] = G[j..j + q - 1]$, the corresponding q-hit is covered by the diagonal $j - i$. For an overlap of k and $w = d + k$ the $|r| \times w$ parallelograms begin at diagonals 0, d, $2d, \dots$. If d is a power of 2, the parallelograms containing a q-hit can efficiently be determined by bit-shifting $j - i$.

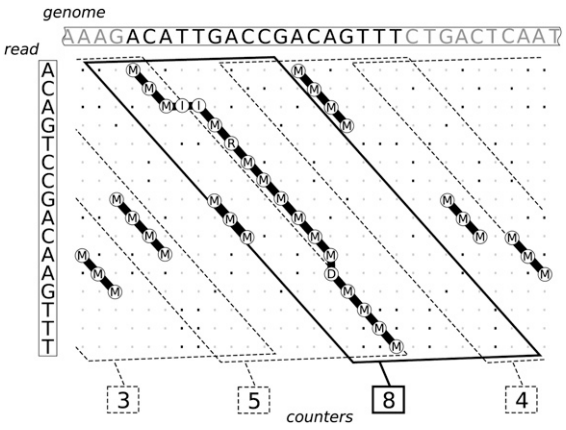

Figure 2. The dot plot between a read and a genome is covered by parallelograms that span $w = 12$ diagonals and overlap by $k = 4$ diagonals. Common 3-gram are counted for each parallelogram. The marked parallelogram contains the complete transcript, therefore it counts all 3-gram that correspond to the seven three-matches in the transcript.

q-Hits are determined by searching overlapping q-grams $G[j..j + q - 1]$, $j = 1,\ldots, |G| - q + 1$ in a q-gram index of all overlapping q-grams of sequences in $\mathcal{R}$. Only a small number of counters are needed per read, when sliding the q-gram over G. As every $|r| + w$ parallelogram spans at most $|r| + w - 1$ letters of G, a parallelogram counter can be re-used after $|r| - w - q$ sliding steps. Before re-using a counter, the associated parallelogram is verified if the counter has reached threshold t.

The SWIFT approach can also be used with gapped shapes Q using a Q-gram index and replacing each q in the formulas above by span(Q).

Verification

To verify a parallelogram possibly containing a true match with edit distance less or equal k, we use Myers (1999) bit-vector algorithm. It exploits hardware parallelism of bit-operations. A 64-bit CPU can calculate 64 cells of the edit matrix at once in a constant number of 14 arithmetic operations.

If $r \in \mathcal{R}$ and $g \prec G$ are the strings covered by a parallelogram of the previous step, it calculates for each $j = 1,\ldots, |g|$ the minimal edit distance between r and all substrings of g ending at position j. If for a position j the minimal edit distance is less or equal to k, a true match is found ending at j. With a slight modification of the algorithm, $g[1..j]$ can be searched for a beginning position i of the true match. Finally, $(r,g[i..j])$ is recorded as a true match.

If only Hamming matches are considered, a parallelogram is verified by scanning each diagonal character by character until more than k mismatches occur. A diagonal d corresponding to $r \in \mathcal{R}$ and $G[d + 1..d + |r|]$ with less or equal k mismatches covers the true match $(r,G[d + 1..d + |r|])$.

Choosing filtration parameters

We now want to automatically choose filtering parameters, a shape Q, and a threshold t, such that (1) we achieve a certain sensitivity level and (2) the running time of the mapping procedure is minimized.

For all read lengths from 24 to 100 and error rates up to 10%, we have therefore precomputed the sensitivities using a number of different shapes and thresholds. As an error distribution we assume a typical Illumina error profile (Dohm et al. 2008). Additionally, all parameter combinations were used to run RazerS on simulated data, yielding a rough estimate for the corresponding filtration efficiency. Parameters for reads longer than 100 base pair (bp) are extrapolated from parameters of precomputed shorter reads with the same error rate. Given a user-defined minimum sensitivity, suitable filtration parameters are chosen from the precomputed tables, such that the anticipated running time is minimized.

If preferred, the parameter tables can be precomputed based on a machine-specific error distribution and user-defined shapes. This error distribution can be obtained in two ways. (1) *Quality-based probabilities:* Transform the average base-call quality value for each position into a probability value. (2) *A posteriori probabilities:* Map a small subset of reads and determine the position-dependent error frequency. Given an error distribution the parameters for reads of length 50 can, for instance, be calculated within 10 min.

Results and Discussion

In this section, we empirically verify the above-described calculation of the expected mapping sensitivity and compare the performance of our tool to that of other read mappers. We use simulated as well as real data. The real data sets were all downloaded from the NCBI Short Read Archive (http://www.ncbi.nlm.nih.gov/Traces/sra/):

- The first set (run accession SRR001815) comprises 10,760,364 *Drosophila melanogaster* reads of length 36 bp.
- The second set (run accession SRR006387) contains $2\times$ 7,894,743 76 bp paired-end reads obtained from whole genome shotgun sequencing of a human HapMap individual, which we trimmed to the first 63 bp (the maximal read length of Zoom at the time of writing).

In both sets, the Illumina technology was used. As references we used the RepeatMasker (http://www.repeatmasker.org) versions of the *D. melanogaster* genome from FlyBase (http://www.flybase.org/), Release 5.9, and the human genome from NCBI (http://www.ncbi.nlm.nih.gov/), Build 36.3.

Verification of expected sensitivity

We verify the correctness of our sensitivity estimation by assessing the discrepancy between estimated and empirical sensitivity for the following two scenarios, where the first one serves as a sanity check:

(1) *Simulated data*. We simulate DNA sequence reads using position-dependent error probabilities and group them according to the number of implanted errors. After mapping the reads to the reference sequence, we define the empirical sensitivity for each group as the proportion of reads that could be mapped back to their genomic origin. Using the same error distribution as for simulation, we compute the estimated sensitivity as described above.

(2) *Real data*. We map the set of reads once with 100% sensitivity and keep as reference matches only those reads that map uniquely. Again, we group them according to the number of errors and determine the empirical sensitivity as for simulated data. The expected sensitivity is computed using the a posteriori probabilities (see previous section).

Using these two protocols, we examined mapping sensitivity for simulated 36 bp reads and for a subset of the 36 bp reads in SRR001815. We inspected both Hamming as well as edit distance sensitivity and did the mapping for all ungapped shapes of weight q where $8 \le q \le 14$ and all thresholds t where $1 \le t \le 20$.

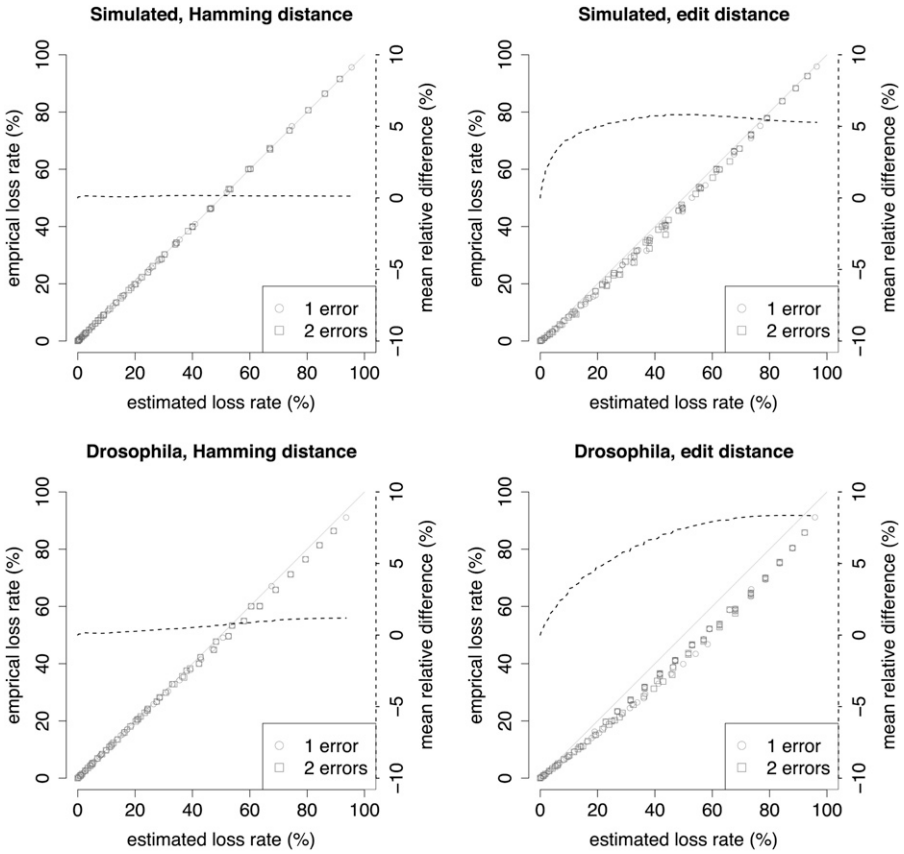

Figure 3. Comparison of estimated and empirical loss rates (loss rate = 1 − sensitivity) varying weight $q = 8,\ldots, 14$ and threshold $t = 1,\ldots, 20$. The dashed line reflects the mean of relative differences, 1 − (empirical loss rate/estimated loss rate), of all estimated loss rates below a varying level.

As a measure of accuracy we use the relative difference between empirical and estimated loss rate. We observe a very good agreement for Hamming distance (see Fig. 3). For expected loss rates between 0 and 10% the mean relative difference for simulated, as well as real data, is below 0.1%. Considering edit distance, the expected loss rates overestimate the empirical loss. Between 0 and 10% of expected loss the mean relative difference is below 4% for simulated and below 2.8% for real data. For edit distance the SWIFT parallelograms are broader and produce more random q-gram hits compared to Hamming distance mapping, where the parallelograms are single diagonals. This leads to more matches than expected. The discrepancy for simulated data is slightly more pronounced. This can be explained by the observation that simulated matches are not necessarily optimal, i.e., reads can be mapped with less error than originally implanted (e.g., an insertion next to a deletion will be aligned as one replacement). Most notably, in all cases the empirical sensitivities are higher than expected, thereby yielding better mapping results than estimated.

Furthermore we investigated the distribution of the lost reads. We therefore compared the output of the 99% with the 100% sensitivity mapping of all *Drosophila* reads onto the *Drosophila* genome and observed a uniform distribution of lost reads over the whole genome. For example, of all 224,229 nonoverlapping 100 bp bins of chromosome X, 199,029 contained at least one mapped read. A percentage (1.14%) of these bins lost one or more reads when mapping with 99% sensitivity. Of all affected bins, only 2.38% lost more than one and 0.22% more than two reads. More than three reads were never lost. We observed similar values for all the other chromosomes.

Runtime comparisons

We demonstrate the performance of RazerS in comparison to five other read mapping tools: MAQ (H Li et al. 2008), SeqMap (Jiang and Wong 2008), SHRiMP (Rumble and Brudno 2008), SOAP (R Li et al. 2008), and the closed-source, commercial Zoom (Lin et al. 2008). We do not consider ELAND (AJ Cox, unpubl.), as we want to map reads longer than 32 bp and only have access to a version limited to 32 bp reads.

All tools differ in some aspects making the comparison rather difficult. MAQ, for example, has a different objective, minimizing the sum of mismatching quality values instead of minimizing the number of mismatches. Also the treatment of N (unknown base) or how many hits per read can be reported is different from tool to tool. We tried to be as fair as possible by using comparable program configurations. In addition, the comparisons should be understood as a means to give the user an impression of the capabilities of RazerS and not as a comprehesive comparison of read mapping tools. All experiments were conducted on an AMD Opteron 2.8 GHz machine with 64 GB of RAM.

Short reads

(1) Drosophila, *Hamming distance 2*. In the first experiment we mapped the *Drosophila* read set SRR001815 onto the *Drosophila* genome with up to two mismatches and measured runtime and space consumption of each read mapping tool. RazerS was run with full sensitivity and with 99% sensitivity. All tools were configured to tolerate two mismatches and output, at most, one best match per read. Since SHRiMP uses a Smith-Waterman verifier, we had to adapt the scoring parameters (mismatch = 0, match = 1, gap penalties = −1000, score threshold = 32).

(2) Drosophila, *edit distance 2*. In this experiment the same set of reads was mapped with up to two errors, allowing mismatches and also indels. Since SHRiMP computes local alignments, there is no scoring scheme for semiglobal edit distance alignments. Nevertheless, we tried to emulate edit distance (mismatch = −1, match = 1, gap penalties = −1, score threshold = 32).

(3) *Human, Hamming distance 5*. For the last experiment on real data we used the first mate set SRR006387 containing 76 bp reads, which we trimmed to 63 bp to incorporate Zoom in our tests. All tools were configured to tolerate five mismatches and output at most one best match per read. Although according to the manual the longest acceptable read length of SOAP is 60 bp, it was capable to match reads of length 63 bp without simply trimming the reads. We tried to run Zoom with 100% sensitivity using the "-sv r5" option, which resulted in an error message, so we reduced Zoom to be fully sensitive for four error

Table 2. Short read results

Experiment	RazerS100	RazerS99	Zoom	SHRiMP	SeqMap	SOAP[a]	MAQ[a]
(1) *Drosophila*, Hamming distance 2							
1M							
Time (min)	2.13	1.63	**1.47**	15.3	6.70	9.27	4.10
Space (GB)	1.31	1.30	0.72	0.68	6.56	0.67	0.60
Mapped reads	505,506	503, 595	505,506	505,084	505,059	506,476	503,999$^{(605K)}$
All							
Time (min)	10.6	**5.55**	7.80	145	12.8	116	9.68
Space (GB)	4.10	3.92	3.77	5.80	11.1	0.67	5.36
Mapped reads	5,353,287	5,335,554	5,353,287	5,349,007	5,348,776	5,414,337	5,338,676$^{(6.5M)}$
(2) *Drosophila*, edit distance 2							
1M							
Time (min)	12.3	**5.92**	32.7	13.6	15.5	—	—
Space (GB)	0.48	0.53	0.72	0.68	8.38	—	—
Mapped reads	512,477	511,695	512,139	515,080	512,477	—	—
All							
Time (min)	163	**68.45**	267	146	Abort	—	—
Space (GB)	4.58	4.59	3.77	5.90	—	—	—
Mapped reads	5,431,142	5,424,088	5,427,589	5,486,467	—	—	—
(3) Human, Hamming distance 5							
1M							
Time (h)	3.14	**0.40**	26.1	10.7	48.8	3.88	2.43
Space (GB)	1.14	1.86	1.27	6.10	8.10	6.20	0.70
Mapped reads	352,725	351,767	352,617	352,742	349,721	354,020	323,893$^{(362K)}$
All							
Time (h)	25.4	**1.95**	45.3	>3 d	Abort	33.8	5.74
Space (GB)	5.60	6.13	2.89	—	—	6.2	4.38
Mapped reads	3,102,320	3,095,435	3,091,063	—	—	3,133,920	2,817,561$^{(3.0M)}$

Results for mapping 1,000,000 and 10,760,364 (all) reads of length 36 bp onto the *Drosophila* genome allowing for two errors considering Hamming (1) or edit distance (2) and for mapping 1,000,000 and 7,894,743 reads of length 63 bp onto the human genome allowing for five replacement errors (3). The fastest method is marked in bold.
[a]SOAP and MAQ do not support edit distance and MAQ also reports matches with more errors (quality based matching).

matches (r4). For SHRiMP we again adapted the scoring parameters as in Experiment 1 with scoring threshold = 58.

The results are summarized in Table 2. Clearly, reducing the sensitivity by 1% has a notable influence on runtime. In Experiment 2 the runtime could be reduced by a factor of 2 and for larger reads in Experiment 3 even by a factor of 13. RazerS99 outperforms the other read mapping tools in almost all experiments and has comparable memory consumption. The runtimes for mapping with Hamming distance are up to 16 times less compared to edit distance. This can be explained by the more efficient Hamming verification algorithm and the observation that gapped shapes, which are chosen for Hamming distance, improve the filtration specificity and runtime. For edit distance, gapped shapes are often less sensitive than their corresponding ungapped shape of equal weight with a smaller span, as indels harm a gapped shape along the whole span.

The numbers of mapped reads are similar across the different read mapping tools. In all experiments RazerS99 recognizes 99.62%–99.87% of the matches found by RazerS100. Zoom finds less matches than RazerS100 in experiments (2) and (3) because it recognizes matches with at most one contiguous gap and is not fully sensitive for five errors. SeqMap loses matches partially covering reference regions consisting of N's. SHRiMP's *q*-gram filter is lossless only for Experiment 3 and reports additional matches because it matches N with N, which are discarded by RazerS. SOAP maps all masked or unknown bases N to A and thus reports more matches. As MAQ uses a quality-based match verification (minimizing the sum of mismatch qualities) it finds matches with more errors, shown in brackets. To make the results comparable, we counted only matches with at most two or five mismatches. The matches not found by MAQ are those having more than three mismatches in the first 28 bp prefix and those that are discarded for a match with more errors and a lower sum of mismatch quality values.

Paired–end reads

RazerS is also capable of mapping paired-end reads.We therefore scan the reference genome from left to right in parallel with two SWIFT filters having the distance of the library size minus a tolerated library size deviation. Each filter searches for potential matches of one of the two ends of all pairs. Additionally, we record in a queue all matches of the left filter within a distance of the doubled tolerated library size deviation. Only if the right filter finds a potential match whose mate potential match is stored in the queue both potential matches are verified. This guarantees that verifications are only done if both potential matches are within the correct distance.

To compare the paired-end alignment performance of RazerS with other paired-end aligners and also to see the behavior of the tools on *unmasked* reference sequences, we mapped the trimmed SRR006387 set of 2 × 63 bp paired-end reads onto the unmasked human chromosome 21 with up to five mismatches per mate, a library size of 200 bp, and a tolerated library size error of 50 bp. Paired-end alignment is supported by MAQ, SOAP, and Zoom, however, SOAP is limited to two mismatches and therefore left out of this comparison. All tools were run with the same options as in Experiment 3 adding the corresponding paired-end options.

Table 3. Results for mapping 2 × 1,000,000 and 2 × 7,894,743 paired-end reads of length 63 bp onto the unmasked human chromosome 21, allowing for five replacement errors per read

Paired-end	RazerS100	RazerS99	Zoom	MAQ
1M				
Time (min)	36.1	6.45	**3.38**	11.5
Space (GB)	1.28	3.13	2.59	0.85
Mapped pairs	26,923	26,828	14,018	19,025$^{(28.5K)}$
All				
Time (min)	71.4	47.5	**22.2**	72.9
Space (GB)	10.8	12.5	20.5	4.72
Mapped pairs	241,308	240,385	129,704	167,015$^{(238K)}$

The fastest method is marked in bold.

Table 3 shows the results. We observe a runtime decrease by a factor of five between full and 99% sensitivity. Compared to Experiment 3 this factor is lower due to the library length criteria, which prevents costly verifications of matches outside the tolerated library size independent of the chosen sensitivity. Zoom is the fastest but also the least sensitive tool. We suspect that the sensitivity option ("-sv r4") is not applicable to paired-end reads. Of all pairs mapped by MAQ within the tolerated library size (values shown in parentheses) we counted those having up to five errors. The smaller number of mapped pairs can be explained as above.

Large reads

To demonstrate that our approach is also applicable to larger reads, we simulated 500,000 reads of lengths 125 and 250 bp of the unmasked human chromosome 21 and implanted up to 10 and 20 errors under a widened Illumina error profile (Dohm et al. 2008). Only RazerS and SHRiMP were able to map both and MAQ was able to map the first read set back onto the chromosome. Other read mapping tools are limited to either shorter reads or less errors. The results are shown in Table 4.

Table 4. Results for mapping 500,000 simulated 125 and 250 bp reads with up to 10 and 20 errors onto the unmasked human chromosome 21 with full and 99% sensitivity

	RazerS100	RazerS99	SHRiMP100	SHRiMP99	MAQ
Hamming					
125 bp					
Time (min)	8.53	**4.15**	9.61 h	60.9	4.54
Space (GB)	0.80	1.54	1.93	4.48	0.38
Mapped	500,000	499,991	500,000	499,991	405,377
250 bp					
Time (min)	14.7	**6.65**	32.9 h	160	—
Space (GB)	1.26	2.00	1.46	2.61	—
Mapped	500,000	500,000	500,000	500,000	—
Edit					
125 bp					
Time (min)	65.0	**23.7**	9.44 h	61.6	—
Space (GB)	0.74	1.71	0.84	4.50	—
Mapped	500,000	500,000	500,000	500,000	—
250 bp					
Time (min)	55.8	**39.6**	14.7 h	28.5 h	—
Space (GB)	1.21	2.18	1.38	5.45	—
Mapped	500,000	499,940	500,000	499,940	—

In each experiment, SHRiMP was manually adjusted to run with the same shape and threshold as automatically chosen by RazerS. The fastest method is marked in bold.

To have a closer look at the performance of SHRiMP compared to RazerS, we adjusted SHRiMP to run with exactly the same filtration parameters making it more efficient than its default parameters. As a result, the numbers of mapped reads of SHRiMP and RazerS are equal in all experiments. RazerS is always six to 40 times faster than SHRiMP due to the more efficient verification and parallelogram filtration. In this setting the number of mapped reads of RazerS99 and RazerS100 are nearly identical. This leads to the conclusion that extrapolating from precomputed parameters of shorter reads underestimates the actual sensitivity, however, the performance increase is comparable to the experiments with short reads. MAQ is only applicable to Hamming distance and reads shorter than 128 bp. In our simulation we set each quality value to 25 and MAQ's quality threshold to 250 in order to prevent MAQ from preferring matches with more than 10 errors. The running time is comparable to RazerS99, however, it is questionable whether an approach using only the first 28 bp as a seed is appropriate for large reads with possible inserts or more than three errors.

Discussion

We presented an efficient read mapping tool that guarantees to find all reads within a user-defined Hamming or edit distance. In addition, a fixed error model and a user-defined loss rate can be used to find the reads at higher speed with controlled sensitivity. Hence, RazerS allows a perfect sensitivity–time tradeoff. Our tool can also handle paired-end reads, as well as arbitrary number of errors and arbitrary read lengths, which make it usable for the new or improved technologies that will provide longer reads. Both latter features are unique among the current implementations. In addition, for typical settings, RazerS outperforms current read mapper algorithms. We are currently incorporating quality values similar to MAQ (H Li et al. 2008) as an optional match verifier to increase the tolerance to sequencing errors. We plan to parallelize the filtration and verification processes using multicore libraries, e.g., Intel TBB (http://www.threadingbuildingblocks.org/) or OpenCL (http://www.khronos.org/opencl). With slight modifications our approach is also applicable to the dinucleotide based ABI SOLiD sequencing technology. RazerS is part of SeqAn (Döring et al. 2008) and is publicly available at http://www.seqan.de/projects/razers.html.

Appendix

Edit distance sensitivity calculation

We now propose a DP algorithm to calculate the sensitivity of a q-gram filter to detect a randomly chosen true match (r,g) with edit distance $d(r,g) \leq k$ as a potential match.

Again, we consider all reads $r \in \mathcal{R}$ to be of equal length n and reduce randomly chosen true matches (r,g) to randomly chosen edit transcripts from r to g with $\Delta = \{M, R, D, I\}$. We therefore assume a given error distribution that associates each nucleotide position i in a read with error probabilities p_i^R, p_i^D, p_i^I, p_i^R and p_i^D are the probabilities that the nucleotide $r[i]$ is replaced or deleted in g. p_i^I is the probability that a single nucleotide

is inserted after the nucleotide $r[i]$ in g. For any transcript T from read to genome we define $||T||_R = |\{i|T[i] \in \{M, R, D\}\}|$, the number of read characters affected by T. Finally, we assume the following occurrence probability of an edit transcript T:

$$p(T) = \prod_{i=1,\dots,|T|} p_{\|T[1..i]\|_R}^{T[i]},$$

with $p_i^M = 1 - p_i^R - p_i^D - p_i^I$. We define the set $\Delta(i) = \{T | T \in \Delta^*, ||T||_R = i\}$ of transcripts from reads of length i. In the following we omit to enumerate transcripts beginning or ending with I, as these transcripts can always be shortened resulting in a match with less errors. Similar to Equation 1 the occurrence probability sum $S(n, e, t)$ of edit transcripts from reads of length n, with e errors, and at least t substrings M^q can be written as

$$S(n,e,t) = \sum_{\substack{T\in\Delta(q),\|T\|_E\le e,\\ T[|T|]\neq I}} R(n-q, e-\|T\|_E, t, T)p(T, n-q), \qquad (4)$$

where $R(i, e, t, T_2)$ is the occurrence probability sum of transcripts $T_i \in \Delta(i)$ with e errors, so that $T_1[1] \neq I$ and T_1T_2 contains at least t substrings M^q. $p(T, j) = \prod_{i=1,\dots,|T|} p_{\|T[1..i]\|_R+j}^{T[i]}$ is the occurrence probability of subtranscript T to occur after j letters of a read.

Lemma 2. *Let e, i, $q \in \mathbb{N}$; $t \in \mathbb{Z}$; $T \in \Delta(q)$. R can be calculated using the following recurrence:*

$$R(0,e,t,T) = \begin{cases} 1, & \text{if } e=0,\ t \le \delta(T),\ T[1] \neq I \\ 0, & \text{else} \end{cases} \qquad (5)$$

$$R(i,-1,t,T) = 0$$

$$\begin{aligned} R(i,e,t,T) = {} & p_i^M R(i-1, e\ \ , t-\delta(T), \text{shift}(M,T)) \\ & + p_i^R R(i-1, e-1, t-\delta(T), \text{shift}(R,T)) \\ & + p_i^D R(i-1, e-1, t-\delta(T), \text{shift}(D,T)) \\ & + p_i^I R(i\ \ , e-1, t\ \ , IT), \end{aligned} \qquad (6)$$

with

$$\text{shift}(x,T) = xT[1..\max\{i | 1 \le i < |T| \wedge T[i] \neq I\}]$$

and

$$\delta(T) = \begin{cases} 1, & \text{if } T \text{ contains } M^q \\ 0, & \text{else} \end{cases}.$$

Proof. See below.

Accordingly, the sensitivities $S(n, e, t)$ for all $e = 0, \dots, k$ and $t = 1, \dots, t_{max}$ can be determined in $\mathcal{O}(nkt_{max}4^{q+k})$.

Extension to gapped shapes

Although "don't care" positions of gapped shapes are not immune to indels, we extend the edit distance sensitivity calculation to gapped shapes for the sake of completeness. To calculate R and S for a gapped shape Q, every q in equation 4 and Lemma 2 must be replaced by span(Q).

Algorithm 1 can be used to detect whether a common Q-gram is retained or destroyed by a transcript affecting *span*(Q) read characters.

Algorithm 1: $\delta(T)$ for gapped shapes

```
Input  : transcript T, shape Q
Output : 0, if T destroys Q-gram; 1, else
r ← 0, g ← 0
j ← min{i | T[i] ≠ I}
for i ← j to |T| do
    if T[i] ≠ D then
        g ← g + 1
    if T[i] ≠ I then
        if r ∈ Q and (T[i] ≠ M or r ≠ g) then
            return 0
        r ← r + 1
return 1
```

Proofs

Proof of Lemma 1. Let $\mathcal{T}(i, e, t, T_2) \subseteq \Delta^i$, $\Delta = \{M,R\}$ be the set of Hamming transcripts with e errors, so that for every $T_1 \in \mathcal{T}(i, e, t, T_2)$ T_1T_2 contains at least t substrings M^q. For $i < 0$, $e < 0$, or $t < 0$, we define $\mathcal{T}(i, e, t, T_2) = \emptyset$.

Randomly choose $i, e, t \in \mathbb{N}$, $i > 0$, $T_2 \in \Delta^q$, and $T_1 \in \mathcal{T}(i, e, t, T_2)$ and let $T_1 = T_1'x$, $T_2 = T_2'y$ for appropriate $T_1' \in \Delta^{i-1}, T_2' \in \Delta^{q-1}$, and $x, y \in \Delta$. As $T_1T_2 = T_1'xT_2'y$ contains at least t substrings M^q it follows that $T_1'xT_2'$ contains at least $t - \delta(T_2'y)$. Additionally, it holds that $e = \|T_1\|_E = \|T_1'x\|_E = \|T_1'\|_E + \|x\|_E$ and thus, $\|T_1'\|_E = e$, if $x = M$, and $\|T_1'\|_E = e - 1$, if $x = R$. Because shift(x, T_2) = xT_2' it follows $T_1' \in \mathcal{T}(i-1, e, t-\delta(T_2),$ shift(M,T_2)) or $T_1' \in \mathcal{T}(i-1, e-1, t-\delta(T_2),$ shift(R,T_2)) and thus:

$$\begin{aligned} \mathcal{T}(i,e,t,T_2) \subseteq {} & \mathcal{T}(i-1, e\ \ , t-\delta(T_2), \textit{shift}(M,T_2))M \\ & \cup\ \mathcal{T}(i-1, e-1, t-\delta(T_2), \textit{shift}(R,T_2))R. \end{aligned} \qquad (7)$$

Now, randomly choose $i', e', t' \in \mathbb{N}$, $x \in \Delta$, $T_2 \in \Delta^q$, and $T_1' \in \mathcal{T}(i', e', t',$ shift(x, T_2)). It holds $|T_1'x| = i' + 1$, $\|T_1'x\|_E = e' + \|x\|_E$, and if T_1'shift(x,T_2) = $T_1'xT_2[1..|T_2| - 1]$ contains at least t' substrings M^q, then $T_1'xT_2$ contains at least $t' + \delta(T_2)$. Therefore, it follows that $T_1'x \in \mathcal{T}(i' + 1, e' + \|x\|_E, t' + \delta(T_2), T_2)$ and thus:

$$\begin{aligned} \mathcal{T}(i,e,t,T_2) \supseteq {} & \mathcal{T}(i-1, e\ \ , t-\delta(T_2), \textit{shift}(M,T_2))M \\ & \cup\ \mathcal{T}(i-1, e-1, t-\delta(T_2), \textit{shift}(R,T_2))R. \end{aligned} \qquad (8)$$

By the definition of R it holds that $R(i, e, t, T_2) = \sum_{T_1 \in \mathcal{T}(i,e,t,T_2)} p(T_1)$. Applied to Equations 7 and 8, Equation 3 follows.

T_2 contains exactly $\delta(T_2)$ substrings M^q, therefore $\mathcal{T}(0, e, t, T_2) = \{\epsilon\}$ if $e = 0$ and $0 \le t \le \delta(T_2)$, otherwise $\mathcal{T}(0, e, t, T_2) = \emptyset$. With $p(\epsilon) = 1$, Equation 2 follows.

Proof of Lemma 2. This lemma can be proven analogously to Lemma 1. Let $\Delta = \{M, R, D, I\}$ and $\mathcal{T}(i, e, t, T_2) \subseteq \Delta(i)$ be the set of transcripts with e errors, so that for every $T_1 \in \mathcal{T}(i, e, t, T_2)$, T_1T_2 contains at least t substrings M^q.

Randomly choose $i, e, t \in \mathbb{N}$, $i > 0$, $T_2 \in \Delta(q)$, T_2 $[|T_2|] \neq I$, and $T_1 \in \mathcal{T}(i, e, t, T_2)$. Let $T_1 = T_1'x$, $T_2 = T_2'y$ for appropriate $T_1', T_2' \in \Delta^*$ and $x, y \in \Delta$. It holds that $\|T_1'\|_R = i - \|x\|_R$ and $\|T_1'\|_E = e - \|x\|_E$. Additionally, $T_1'xT_2'$ and thus also T_1' shift $(x, T_2'y)$ contain at least $t - \delta(T_2'y)$ substrings M^q. This proves the "$\subseteq$" part of Equation 6. We omit the analog rest of the proof due to space limitations.

References

Barski A, Cuddapah S, Cui K, Roh T, Schones D, Wang Z, Wei G, Chepelev I, Zhao K. 2007. High-resolution profiling of histone methylations in the human genome. *Cell* **129:** 823–837.

Bentley DR, Balasubramanian S, Swerdlow HP, Smith GP, Milton J, Brown CG, Hall KP, Evers DJ, Barnes CL, Bignell HR, et al. 2008. Accurate whole human genome sequencing using reversible terminator chemistry. *Nature* **456:** 53–59.
Burkhardt S, Kärkkäinen J. 2002. Better filtering with gapped *q*-grams. *Fundam Inf* **56:** 51–70.
Burkhardt S, Crauser A, Ferragina P, Lenhof H-P, Rivals E, Vingron M. 1999. *q*-Gram based database searching using a suffix array (Quasar). In *RECOMB'99*, Lyon, France, pp. 77–83. ACM, New York.
Chen W, Kalscheuer V, Tzschach A, Menzel C, Ullmann R, Schulz MH, Erdogan F, Li N, Kijas Z, Arkesteijn G, et al. 2008. Mapping translocation breakpoints by next-generation sequencing. *Genome Res* **18:** 1143–1149.
Dohm JC, Lottaz C, Borodina T, Himmelbauer H. 2008. Substantial biases in ultra-short read data sets from high-throughput DNA sequencing. *Nucleic Acids Res* **36:** e105. doi: 10.1093/nar/gkn425.
Döring A, Weese D, Rausch T, Reinert K. 2008. SeqAn an efficient, generic C++ library for sequence analysis. *BMC Bioinformatics* **9:** 11. doi: 10.1186/1471-2105-9-11.
Herms I, Rahmann S. 2008. Computing alignment seed sensitivity with probabilistic arithmetic automata. *Lect Notes Comput Sci* **5251:** 318–329.
Hillier LW, Marth GT, Quinlan AR, Dooling D, Fewell G, Barnett D, Fox P, Glasscock JI, Hickenbotham M, Huang W, et al. 2008. Whole-genome sequencing and variant discovery in *C. elegans*. *Nat Methods* **5:** 183–188.
Huson DH, Auch AF, Qi J, Schuster SC. 2007. Megan analysis of metagenomic data. *Genome Res* **17:** 377–386.
Jiang H, Wong WH. 2008. SeqMap: Mapping massive amount of oligonucleotides to the genome. *Bioinformatics* **24:** 2395–2396.
Jokinen P, Ukkonen E. 1991. Two algorithms for approximate string matching in static texts. *Lect Notes Comput Sci* **520:** 240–248.
Kent W. 2002. BLAT—the BLAST-like alignment tool. *Genome Res* **12:** 656–664.
Langmead B, Trapnell C, Pop M, Salzberg S. 2009. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biol* **10:** R25.
Levenshtein VI. 1966. Binary codes capable of correcting deletions, insertions and reversals. *Sov Phys Dokl* **10:** 707.
Ley T, Mardis E, Ding L, Fulton B, McLellan M, Chen K, Dooling D, Dunford-Shore B, McGrath S, Hickenbotham M, et al. 2008. DNA sequencing of a cytogenetically normal acute myeloid leukaemia genome. *Nature* **456:** 66–72.
Li M, Ma B, Kisman D, Tromp J. 2003. PatternHunter II: Highly sensitive and fast homology search. *Genome Inform* **14:** 164–175.
Li H, Ruan J, Durbin R. 2008. Mapping short DNA sequencing reads and calling variants using mapping quality scores. *Genome Res* **18:** 1851–1858.
Li R, Li Y, Kristiansen K, Wang J. 2008. SOAP: Short oligonucleotide alignment program. *Bioinformatics* **24:** 713–714.
Lin H, Zhang Z, Zhang MQ, Ma B, Li M. 2008. Zoom! Zillions of oligos mapped. *Bioinformatics* **24:** 2431–2437.
Ma B, Tromp J, Li M. 2002. PatternHunter: Faster and more sensitive homology search. *Bioinformatics* **18:** 440–445.
Morin RD, O'Connor MD, Griffith M, Kuchenbauer F, Delaney A, Prabhu A-L, Zhao Y, McDonald H, Zeng T, Hirst M, et al. 2008. Application of massively parallel sequencing to microrna profiling and discovery in human embryonic stem cells. *Genome Res* **18:** 610–621.
Myers EW. 1999. A fast bit-vector algorithm for approximate string matching based on dynamic programming. *JACM* **46:** 395–415.
Navarro G, Raffinot M. 2002. *Flexible pattern matching in strings*, Ch. 6.5, pp. 162–166. Cambridge University Press, Cambridge, MA.
Owolabi O, McGregor DR. 1988. Fast approximate string matching. *Softw Pract Exper* **18:** 387–393.
Rasmussen K, Stoye J, Myers G. 2006. Efficient *q*-gram filters for finding all ϵ-matches over a given length. *J Comput Biol* **13:** 296–308.
Rumble S, Brudno M. 2008. SHRiMP—SHort Read Mapping Package. http://compbio.cs.toronto.edu/shrimp.
Smith TF, Waterman MS. 1981. Identification of common molecular subsequences. *J Mol Biol* **147:** 195–197.
Wang J, Wang W, Li R, Li Y, Tian G, Goodman L, Fan W, Zhang J, Li J, Zhang J, et al. 2008. The iploid genome sequence of an Asian individual. *Nature* **456:** 60–65.

Received November 2, 2008; accepted in revised form April 29, 2009.