SEME: A Fast Mapper of Illumina Sequencing Reads with Statistical Evaluation

Shijian Chen[1,⋆], Anqi Wang[1,⋆], and Lei M. Li[1,2,⋆⋆]

[1] NCMIS, Academy of Mathematics and Systems Science,
Chinese Academy of Sciences, Beijing, 100190
[2] Molecular and Computational Biology Program, Department of Biological Sciences,
University of Southern California, Los Angeles, CA 90089
lilei@amss.ac.cn

Abstract. Mapping reads to a reference genome is a routine yet computationally intensive task in research based on high-throughput sequencing. In recent years, the sequencing reads of the Illumina platform get longer and their quality scores get higher. According to our calculation, this allows perfect k-mer seed match for almost all reads when a close reference genome is available subject to reasonable specificity. Our another observation is that the majority reads contain at most one short INDEL polymorphism. Based on these observations, we propose a fast mapping approach, referred to as "SEME", which has two core steps: first it scans a read sequentially in a specific order for a k-mer exact match seed; next it extends the alignment on both sides allowing at most one short-INDEL each, using a novel method "auto-match function". We decompose the evaluation of the sensitivity and specificity into two parts corresponding to the seed and extension step, and the composite result provides an approximate overall reliability estimate of each mapping. We compare SEME with some existing mapping methods on several data sets, and SEME shows better performance in terms of both running time and mapping rates.

Keywords: high-throughput sequencing, mapping, perfect match, INDEL, auto-match function.

1 Introduction

The Next Generation Sequencing (NGS) technologies are generating unprecedented large amounts of short reads in routine genome research. The high-throughput and read length of NGS make it especially suitable for re-sequencing individuals with known references and thus for detecting variations. In whole genome re-sequencing projects for mammals, NGS usually generates billions of short reads, and mapping these reads back to the reference genome is computationally intensive. Hence the design of efficient mapping algorithms is a key and challenging problem in current computational biology.

Many short-read mapping methods have been developed along the evolution of the sequencing technologies[1]. The specific read length, error rates and patterns of each

⋆ These authors contribute equally.
⋆⋆ corresponding author.

M. Deng et al. (Eds.): RECOMB 2013, LNBI 7821, pp. 14–29, 2013.

technology at the time are the primary constraints in the design of mapping algorithms. In the early days of NGS, the short reads were only 35bp long and error rates were fairly high for the Illumina/Solexa platform. Besides, 5-6 years ago the 32-bit architecture was the main model for PCs or cluster nodes, and their memory size is limited to 4Gb. Bowtie[2] applied the Burrows-Wheeler transform and FM index to the representation of the reference, and could reduce the memory footprint to as low as 1.3Gb for the human genome. This advantage makes Bowtie very popular among high-throughput sequencing users. Although the Burrows-Wheeler transform is effective in searching perfect matches of a k-mer in a reference, we have to allow mismatches to maintain sensitivity. For instance, MAQ[3] and SeqMap[4] use spaced seeds which allow up to k mismatches. Bowtie conducts a backtracking search to allow mismatches, and mitigates excessive backtracking by "double indexing", which doubles the memory foot print. No matter what method is used for handling mismatches, complexity is substantially increased.

As chemistry and instruments of NGS are under constant improvement, the reads are getting longer with higher quality. Now the Illumina platform can generate reads longer than 100bp with fairly high quality. MiSeq[5] can even sequence reads up to 250bp. Some short read mapping programmes, like Bowtie2, have been developed for these longer reads. Bowtie2 maps multiple evenly distributed seeds of a read and uses dynamic programming to extend seed alignments into a full alignment that allows INDELs. We observed that INDEL errors are extremely rare compared to substitution errors for Illumina systems. Thus if an INDEL occurs in the alignment or mapping, most likely it is a result from a polymorphism between the read and the reference.

Most high-throughput sequencing applications are for conserved genomes such as human, which is the focus of this article. In [6], it is found that the size of INDEL obeys a power law distribution in Human and Rodent pseudo genes: 78 human pseudo genes have been analyzed and it shows that the average length of small INDEL is less than three; furthermore, among those INDELs with length no larger than 20bp, 95% of them are no larger than 11bp. In[7], it is found that INDELs locate throughout the genome at a frequency of one per 7.2kb on average. If we approximate the occurrences of short INDELs by a Poisson point process that matches the frequency [12], the probability of finding at most one INDEL in a 100bp window is greater than 0.9999. Most existing methods apply dynamic programming to allow general INDELs. This is unnecessary most of the time for mapping short reads when a close reference genome is available.

Partially motivated by the above considerations, in this article we propose a new short read mapping method, referred to as SEME (Sequential Exact seed-Match and Extend) hereafter, which focuses on mapping Illumina short reads generated from conserved genomes. Different from most existing Seed-and-Extend methods which map multiple seeds simultaneously, SEME scans the read according to a specific strategy and maps the seeds sequentially. Once a seed is perfectly matched to the reference we extend it on both sides to get the full alignment result or reject it. This approach avoids mapping a fixed number of seeds for each short read. The higher the sequencing quality is, the less number of seeds are needed in SEME on average. This feature is particularly favorable as sequencing technology improves. In the extension step, we introduce the AMF (Auto-Match-Function) method to detect up to two INDELs. Compared with

alignment algorithms based on dynamic programming, the average complexity of the AMF method is linear. For the remaining complicated occasions, which are rare, we can incorporate the Smith-Waterman[8] algorithm for full alignments.

As important as the computational complexity of an algorithm, its mapping rate and accuracy, which is usually measured by sensitivity and specificity, needs to be statistically evaluated fairly. For example, BLAST[9] is now widely used in the search of sequence databases. Its success comes from both its efficient algorithmic implementation and the associated statistical evaluation of the alignment significance[10,11,12]. In the situation of mapping short reads, the read length, say 100bp, is so small compared to the the genome size, that the classical asymptotics of alignment cannot be applied directly. In this report we make some efforts to evaluate the accuracy of the SEME procedure. In concert with the algorithm, we start off by comparing two sequences of the same read length. If one sequence is different from the other by only substitution and small INDEL polymorphisms plus sequencing errors, then the chance of detecting matching is essentially sensitivity. On the other hand, if one sequence is sampled randomly, say according to an *i.i.d.* – independent and identically distributed – model for the sake of simplicity, then accepting a match leads to a false positive error and its chance needs to be calculated. To evaluate the overall specificity, we decompose the entire genome into many reads of the same read lengths, either overlapping or non-overlapping, and apply the above result to provide bounds to the probability of accepting at least one match by chance across the genome. We could complement the analysis based on the simple model by simulation as well. With such a probabilistic framework that takes into account of read length, read error pattern, and polymorphism rate, we can optimize the seed length by trading off sensitivity and specificity.

To enhance sensitivity, we propose a soft counting criterion for accepting or rejecting a mapping result if appropriate sequencing quality scores are available. That is, we impute "possible polymorphism" fractions from mismatches based on polymorphism rates and quality values, and use the sum of these fractions for decision.

2 Method

SEME follows a "seed-and-extend" paradigm. In the first stage, it extracts k-mers sequentially from a short read, and for each k-mer SEME searches through the reference for perfect-match locations, where the read can anchor. We will discuss the selection of k later. In the second stage, SEME extends the seed on both sides separately. If the read is indeed from a reference location, then their true alignment falls into three categories: no INDEL; one INDEL; other more complicated INDEL patterns. As we explained earlier, most short reads from a conserved genome contain no more than one INDEL and possibly some mismatches. Thus the principal task of extension can be simplified as follows: on each side of the seed, detect the possible "one-INDEL" including its type, position and length (no larger than a given upper bound). We introduce the auto-match vector and auto-match function to efficiently solve this problem.

The search of k-mer exact-match across a reference genome is a common theme in most mapping tools. Several options are available for implementation. If memory size

is limited, then the Burrows-Wheeler Transform is a good choice for compressing the genome and index information. If memory is sufficiently large, then hashing can help speed up the search, [19].

2.1 Index Table of Sorted 32-Mers and Binary Search

In our scheme, we may use more than one kind of seed sizes depending on the data. Therefore we propose to use the index table of 32-mers for the human genome. That is, we encode each 32-mer subsequence of the reference genome by an integer $s_i, i = 1, \cdots, L$, where L is the genome size, and sort them by the heap sorting algorithm. Denoted the sorted 32-mer-integers by $s_{(1)} \leq s_{(2)} \leq s_{(i)} \leq s_{(L)}$, and their corresponding addresses on the genome by $a(s_{(i)}), i = 1, \cdots, L$. We keep the addresses of these sorted 32-mers in an array $u[i] = a(s_{(i)}), i = 1, \cdots, L$, referred to as "index table" hereafter. We also put the reference genome in RAM so that we can quickly find the i-th sorted k-mer by linking the i-th address in the index table with the genome, see Fig 1. Note we do not save the sorted 32-mer-integers in a vector directly because an 32-mer takes 8 bytes while the address of the 32-mer takes only 4 bytes. With such an index structure, we apply binary search, whose time complexity is $O(log_2 L)$. Take the human genome for example, as the size of the index table is about three billion, approximately 30 steps are needed to insert a k-mer into the index table.

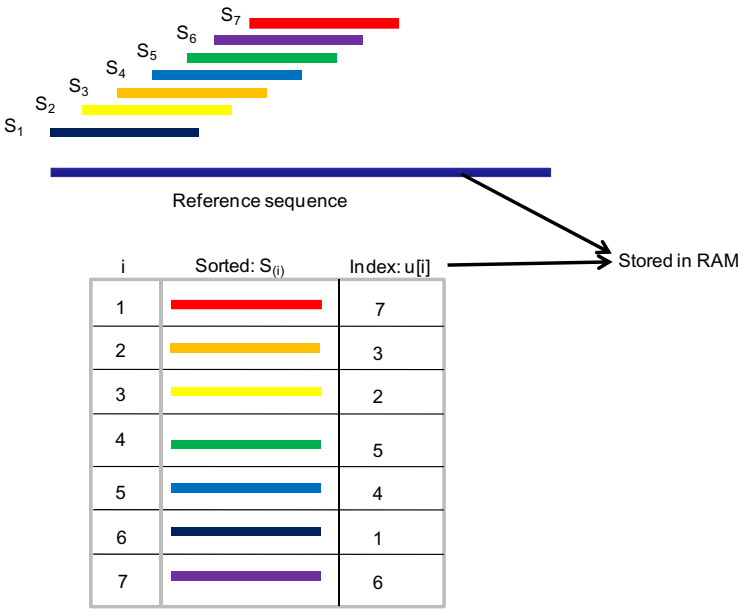

Fig. 1. Illustration of the index table. The blue bar is the reference sequence. The short bars on top of the reference represent 32-mers $s_{(i)}$ extracted from the reference. The sorted 32-mers $\{s_{(i)}\}$ and their corresponding indices $u[i]$ are listed in the table below. Only the reference genome and $\{u[i]\}$ are kept in RAM.

We make a note here. Regardless the value of k, we can carry out a search of the 32-mer starting at the same position as the k-mer. Along the binary search, the lower bound either stays or moves upwards while the upper bound either stays or moves downwards. We could have two outcomes: at some point, the 32-mer hits a match with either the lower or the upper bound; otherwise, the 32-mer matches neither of the two bounds when they meet. In the former case, the 32-mer finds a perfect match. In the latter, we check the maximum number of matching nucleotides between the target and the lower bound starting from the beginning position. Similarly we check the number for the upper bound. If this number is no smaller than k, then the k-mer has its perfect match on the genome. In comparison, search based on hashing does not have this flexibility.

Each item in the index table is a 32-bit integer which needs 4 bytes and the reference genome takes no more than 1Gb. They add up to no more than 13Gb. As the 64-bit architecture is taking over in the computer business, this memory requirement is not a serious problem. However, if we select every other 32-mer in the genome, say those at the odd addresses, then the resulted index table would be around 6Gb, and the total memory requirement is less than 7Gb. Of course, to be consistent with this configuration, we need to search two consecutive k-mers on a short read before we jump to the next seed. According to our simulation, this reduction of memory sacrifices very little in terms of performance.

To reduce the steps of binary search, we could introduce "block address" or "zip code" for each 32-mer, which is encoded into an integer in the range $[0, 4^{32} - 1]$. For a number $r < 30$, we pick up the 2^r integers $d_i = i * 2^{64-r}, i = 0, \cdots, 2^r - 1$, that divide the range uniformly, and insert each of them to the index table of the sorted 32-mers of the human genome. Denote the two indices that are just next to d_i are $(u[j_i], u[j_i + 1])$, namely, $s_{(j_i)} \leq d_i \leq s_{(j_i+1)}$ — it is possible that $s_{(j_i)} = s_{(j_i+1)}$. Now we keep the pointers $[j_i]$ in an array $q[i], i = 0, \cdots, 2^r - 1$, referred to as "block address vector" hereafter. In the practice of mapping reads, we load $q[i]$ together with $u[i]$ and the genome into computer memory. For an 32-mer-integer s, we divide it by 2^{64-r}, and the resulting integer after rounding off gives its block index denoted by i_1. Suppose $q[i_1]$ and $q[i_1 + 1]$ respectively point to $u[j_1]$ and $u[j_2]$, then the two indices $u[j_1]$, $u[j_2 + 1]$ can serve as a more delicate starting point of the lower and upper bound respectively for the binary search of s. Since the distribution of $s_i, i = 1, \cdots, L$ can roughly be approximately by a uniform distribution, we could reduce r steps of binary search on averge using this strategy. Of course, the larger the r is, the more memory is needed. If we take $r = 15$, at the cost of 128K more memory, we could reduce the the average complexity of the binary search by half.

2.2 Seed Stage

In this stage, we use the strategy GSM (Grouped Scan and Map) to scan the short read sequentially to find a perfectly matched seed, in other words, to anchor the short read to a candidate position in the reference genome. Since we only index a single strand of the reference to save memory, we scan both short reads and their reverse complements. For the sake of simplicity, we just describe the scan scheme on one strand.

The scan function GSM puts all seeds of a short read into several groups. It scans the first seed of each group in the first round, and the second seed in the next round.

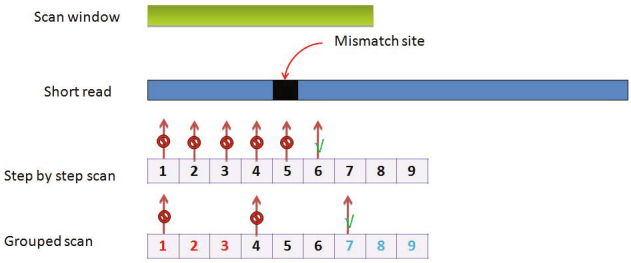

Fig. 2. Illustration of the grouped scan method. The green bar shows the scope of a scan window from which a seed is extracted. The blue bar represents a short read, on which a mismatch is marked by a black square. If we scan the read nucleotide by nucleotide, we would go though five failed mapping marked by red before the successful mapping marked by green occurs. If we scan the read with jump 3, only two failed mapping occur before a success.

This process goes on till a seed is mapped or the number of trial seeds exceeds a certain threshold. Fig. 2 is an illustration of the method. It can be seen that five seeds have to be scanned before the perfect match seed is detected by the step-by-step scan method. In contrast, we only need to map two seeds before the detection of a perfect match seed using a proper grouping strategy.

The grouped scan strategy reduces the number of trial seeds in most occasions. In our experience, if a short read is mappable(can be mapped back if all seeds are scanned) then the number of trial seeds does not exceed a certain threshold in most cases. We could experiment with a small portion of the read data to set this threshold. The principle will be discussed in section 2.4.

In addition to the scan order, seed length is another important factor we should consider. Later we will estimate the length interval which meets both sensitivity and specificity requirement on the basis of a probabilistic model. Seed lengths near the upper bound of the interval give the best specificity while seed lengths near the lower bound give the best sensitivity. If we put specificity prior to sensitivity, at each scan position we can first map a seed at the upper bound and then map a seed at the lower bound.

2.3 Extension Stage

In this stage, we detect the pattern, length and position of a possible INDEL. The core of the method are the notions of auto-match vector and auto-match function which we will define as follows.

Given two DNA segments denoted by S_1 and S_2, not necessarily of the same length, we define $V(S_1, S_2)$ to be a vector whose i-th element is 0 if the i-th elements of S_1 and S_2 are the same, and is 1 otherwise. The length of $V(S_1, S_2)$ is the shorter one of S_1 and S_2. For any string S, denote the substring of S with the first i elements removed as $S\{i\}$.

We define match vectors as: $M(0) = V(S_1, S_2)$; $M(i) = V(S_1\{i\}, S_2)$, for $i > 0$; $M(-i) = V(S_1, S_2\{i\})$, for $i > 0$, see Fig. 3. The auto-match vector $w(i)$ of S_1 and S_2 is defined as: the i-th element of $w(i)$ is the minimum of the i-th element of $M(0)$ and $M(i)$. Fig. 5 illustrates how $w(1)$ is obtained from $M(0)$ and $M(1)$. Finally, we define

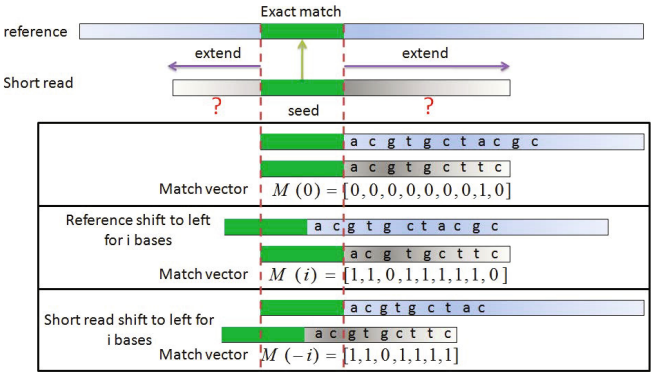

Fig. 3. Illustration of the auto-match vectors during extension

the auto-match function $AMF(i)$ to be the number of 1's in the auto-match vector $w(i)$. $AMF(0)$ is simply the number of 1's in the match vector $M(0)$.

With the help of AMF, we can detect the pattern and length of an INDEL. Fig. 4 shows a case of a two-nucleotide deletion, in which $AMF(i)$ is zero only for $i = 2$ while all other values are larger than five. We use this property to detect the pattern and length of an INDEL. Once the type and length of an INDEL is determined, we further use auto match vectors to detect its position. The idea is illustrated in Fig. 5, where we

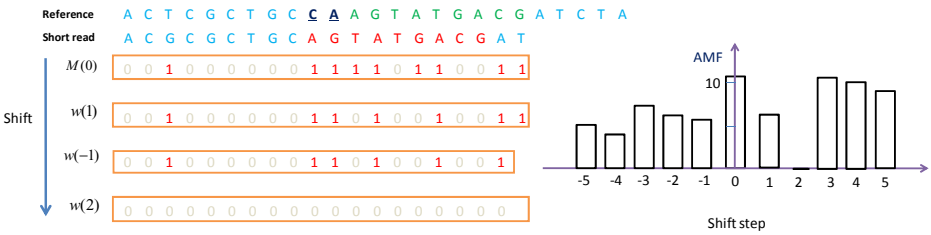

Fig. 4. The pattern of AMF corresponding to a deletion of size 2

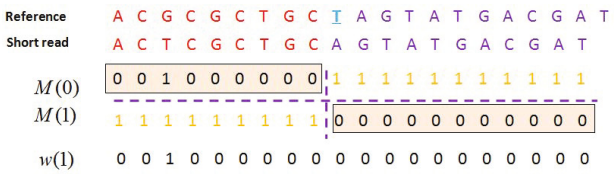

Fig. 5. Detection of the position of a DELETION. A nucleotide 'T' in green color on the reference genome is deleted. In $M(0)$ almost all elements before this nucleotide are 0 while almost all elements after it are 1. In $M(1)$ the pattern is just the opposite.

only consider $M(1)$ because the AMF calculation indicates that an 1-nucleotide-deletion exists somewhere, and the purple boundary indicates the position of the deletion.

Now we summarize the general AMF method as below.

Algorithm 1 AMF Algorithm

1. *Examine* $AMF(0)$, $AMF(1)$, $AMF(-1)$, ..., $AMF(d)$, $AMF(-d)$ *sequentially (d is the maximum length of INDEL allowed) till* $AMF(\mu) < \xi$ *for a certain* μ*, where* ξ *is a predetermined value. A positive* μ *means a DELETION, and a negative* μ *means an INSERTION. The absolute value of* μ *estimates the length of the INDEL.*
2. *If such* μ *does not exist, we skip this extension (either a false mapping or a more complicated INDEL pattern exists); Else if* $\mu = 0$*, it means no INDEL; otherwise we use the pair* $[M(0), M(\mu)]$ *to detect the position of the INDEL in the next step.*
3. *Take the subsequence to the right of the mapped seed for example, and denote its length by l.*
   - *Initialization: let* $D_0 = \sum_j w(\mu)_j$, $TMP = D_0$, $POSITION = 0$.
   - *Recursion: For* $j = 1 : l$, $D_i = D_{i-1} + [M(0)_i - M(\mu)_i]$; *If* $D_i < TMP$, $TMP = D_i$, *and* $POSITION = j$.
   - *Output POSITION.*

For 100bp re-sequencing reads of the human genome, only a tiny fraction could be anchored by a fairly large seed, say 32bp, but could not be extended by the AMF method, and they are examined by the Smith-Waterman algorithm.

2.4 Computational Complexity

Some notations and definitions that are necessary for the complexity evaluation are listed in Table 1. We first consider those reads that can be mapped to the reference. Mapping such a short read is accomplished through: 1) finding a perfect match seed, 2) detecting the INDEL length, 3) detecting the start position of the INDEL if its length is nonzero. Next we decompose the time spent on each part in details according to the algorithm.

The time spent on exact match is $n_s T_{mp\_seed}$, where T_{mp_seed} varies depending on the algorithmic implementation and hardware. If we take the searching scheme described in

Table 1. Symbols and notations used in the complexity evaluation

Symbol	Definition
n_s	number of scanned seeds in a read, $n_s \leq 2(n-k+1)$
n_w	number of seeds which are mapped to the reference but cannot be extended
T_{cmp_nt}	time of comparing a pair of nucleotides
T_{cmp_int}	time of comparing two integers
T_{add_int}	time of adding two integers
T_{mp_seed}	time of mapping a single seed
l	length of the read's subsequence involved in extension, $l \leq n-k+1$
μ	length of an INDEL
Q	maximum length of an INDEL to be detected

Subsection 2.1, mapping a single seed has three steps: 1) obtaining a starting lower and upper bound in the index table for the seed using its block address; 2) binary searching for the two adjacent 32-mers between which the seed can insert; 3) finding the maximum length of perfect match up to 32 nucleotide bases. In the first step, the integer corresponding to a 32-mer seed needs to be divided by 2^{64-r}, where r is the number of binary search we would like to reduce. This can be achieved by $64 - r$ shift operations on the integer. In addition, two data access operations are required to get the two starting index bounds. The second step contains about $(30 - r)$ data access operations and $30 - r$ integer comparison. The third step can easily be implemented by shifting and comparing integers.

After finding a perfect match seed, we need to compute the values of AMF function to detect the possible "1-INDEL" length μ. First, the calculation of $M(i)$ takes l, l and $l-|i|$ comparisons of nucleotides pairs between the read and reference respectively for $i = 0$, $i > 0$ and $i < 0$. Second, calculation of $w(i)$ takes l and $l - i$ comparisons of Boolean elements in $M(0)$ and $M(i)$ respectively for $i > 0$ and $i < 0$. Third, the calculation of $AMF(i)$ takes roughly l integer additions. To detect an INDEL's start position, we can implement the third step of Algorithm 1 with the following time respectively for deletion and insertion:

$$(l-1)T_{add\_int} + l(T_{cmp\_int} + T_{add\_int}), \quad \text{and} \quad (l-\mu-1)T_{add\_int} + (l-\mu)(T_{cmp\_int} + T_{add\_int}).$$

Putting together and assuming we calculate $AMF(i)$ in the order of $i = 0, 1, -1, 2, -2, \cdots$, we have the following total time, ignoring the constant terms with respect to l.

$$T \approx n_s T_{mp\_seed} + n_w\, l[(2Q+1)T_{cmp\_nt} + 2QT_{cmp\_int} + (2Q+1)T_{add\_int}] \tag{1}$$

$$+ \begin{cases} l[T_{cmp\_nt} + T_{add\_int}] & \mu = 0 \\ l[2\mu T_{cmp\_nt} + 2\mu T_{cmp\_int} + (2\mu+2)T_{add\_int}] & \mu > 0 \\ l[(2|\mu|+1)T_{cmp\_nt} + (2|\mu|+1)T_{cmp\_int} + (2|\mu|+3)T_{add\_int}] & \mu < 0 \end{cases} \tag{2}$$

The first term is the complexity of mapping seeds; the second term is the complexity of the unsuccessful extension of those anchored seeds. The third term is the complexity of the successful extension of the final seed. Possibly n_s includes the number of seeds that cannot be mapped anywhere, thus $n_s \geq n_w$. Later we will show that the specificity goes up as the seed length goes up. When the specificity is sufficiently large, the chance of $n_w > 0$ is small. For those reads that cannot be mapped to the reference, the third term is zero. So the time is

$$T \approx n_s T_{mp\_seed} + n_w\, l[(2Q+1)T_{cmp\_nt} + 2QT_{cmp\_int} + (2Q+1)T_{add\_int}].$$

In our experience, for most of the mappable reads, the number of trial seeds is much smaller than the total number of seeds. If we set a threshold for the number of trial seeds then we avoid fruitlessly scanning. To set this threshold, we need to know the distribution of n_s for the mappable reads. Let $A(i) = \#\{n_s(\text{among mappable reads}) = i\}$, namely, the number of reads which need i trial seeds till a successful mapping, $1 \leq i \leq (n - k + 1)$, Fig. 6 shows the frequencies of $A(i)$ for an 100K-short-read data set. It is obvious that most of the mappable short reads are scanned only a few times. In fact, the 99% quantile of $A(i)$ in this example is 11, and the average number of trial seeds, for all

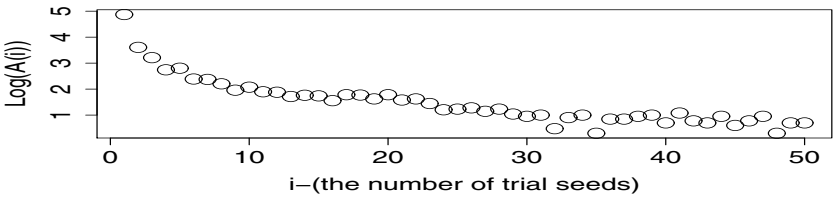

Fig. 6. Frequencies of n_s for mappable reads. The y-axis shows $log_{10}(A(i))$. The results are obtained from 100K 76bp short reads downloaded from NCBI data base, archive SRR003196. Among them 83K reads are mappable.

the short reads (including the unmapped reads) is only 2.9 for this data set. This gives an estimate of $E(n_s)$ and it explains, at least from one angle, why the sequential seeding strategy is efficient compared with that of fixed-number-seeding. The higher the quality of a read data set is, the less the average number of trial seeds are needed.

We also calculate the average length of INDELs in the example explained in the introduction section. It turns out that the average of μ is around 2.9. This means that on average, we only need to shift a short read rightwards and leftwards with respect to the reference 3 times.

3 Statistical Evaluation

In this section we evaluate the mapping accuracy of SEME based on probabilistic models. Several important statistical approaches have been developed for specific sequence alignment problems. For example, the statistic D_2[13] concentrates on the number of k-mer perfect match between two sequences of lengths m and n, and evaluate its asymptotics when m and n go to infinity. The concept of excursion in random walks and some other advanced techniques in probability were used in evaluating the significance of BLAST[14] results. In the current mapping problem, the read, say 100 bp, is much shorter than the reference genome. The asymptotics that requires both m and n go to large do not apply. In order to evaluate the sensitivity and specificity of the SEME mapping result, we propose another approach, which essentially compares the n-length read with every n-length subsequence of the reference.

Suppose that the read length is n, and we define sensitivity to be the probability that a read is mapped to where it is from, and specificity as the probability that the read does not map to any other positions – excluding repeats and possibly highly conserved homologs – on the reference. We approximate this event by any positions on a random reference of the same size. Let ν be the chance that the read is mapped to a random n-mer subsequence. According to subadditivity of probability

$$1 - specificity \leq \min\{N\nu, 1\},$$

Corresponding to the two stages of SEME, we make the following decomposition:

$$\text{sensitivity} = \tau\,\theta, \qquad \nu = \eta\,\theta^*, \tag{3}$$

where τ and r are respectively the probabilities of finding a k-mer perfect match between two n-length sequences under a correct location and a random location. θ and θ^* are respectively the conditional probabilities of accepting extension under a correct location and a random situation. Assuming that the n positions are independent and the match rate is constant, we calculate τ and η precisely, and the result is accurate no matter what the read length is. θ is obtained by a soft counting method, which calculates the probability of the extension based on the distribution of the imputed "possible polymorphism" numbers that aim to adjust the effect of base-calling errors.

Lemma. For two n-length sequences, assume that bases at different positions are independent and the match rate for all positions is a constant p, then the probability that an k-mer perfect match exists is given by

$$\tau(k,p) = \sum_{m=0}^{n}[\sum_{s=1}^{K(m)}(-1)^{s+1}C_{m+1}^{s}C_{n-ks}^{m}](1-p)^m p^{n-m},$$

$$\text{where} \qquad K(m) = \max\{s; n-ks \geq m\} \wedge (m+1).$$

In fact, $\tau(k,p)$ increases with p and decreases with k.

We first apply this lemma to the calculation of $\tau = \tau(k;p)$, where p is the matching rate between a read and the region where it is from and it depends on the polymorphism rate and sequencing error rate. Let X, Y, S represent the reference, individual genome and short read respectively. It can be shown that the mismatch rate at the site (X_i, Y_i, S_i) is $(1-\beta_i)\gamma + (1-\gamma)\beta_i + \gamma\beta_i w_i$, where $\gamma = Pr(X_i \neq Y_i)$ is the polymorphism rate, or simply the SNP rate if we skip the INDEL for the moment; and $\beta_i = Pr(Y_i \neq S_i)$, the miscall rate; $w_i = Pr(S_i \neq X_i | Y_i \neq S_i, Y_i \neq X_i)$. We note that in this context we use the jargon "polymorphism rate" γ as a measure of genomic discrepancies between the target individual and the reference, but not as a measure of population genetics. Since $\gamma\beta_i$ is small, we have the approximation to the match probability: $p_i \approx 1-\gamma-\beta_i$. For the moment, we replace β_i by their average. We show the curves of τ under different settings in Fig 6(a). For example, the green solid line corresponds to the case of 100bp-reads with a 0.99 match rate. In this case, the sensitivity is satisfactory even when $k = 30$.

Next we apply the lemma to the random situation. In $\eta = \tau(k; p_\eta)$, we set the match rate p_η to be the sum of the squares of the base composition rates (usually around 0.25). In Fig 6(a), the red line, representing the trend of η, drops to zero quickly even when the seed is short, and to some extent it displays the specificity of SEME.

Soft Counting Criterion. The extension stage validates the anchor position by checking the the number of inconsistencies between the reference and the read. For now, we simply exclude INDEL positions. MAQ [3] evaluates a mapping result by calculating the posterior probability that the read comes from the region, and it regards the hit with the highest posterior as the correct result. The posterior can be maximized effectively by minimizing the sum of quality values of mismatched bases. We note that mismatches could be caused by miscalls as well as polymorphisms, and base-calling errors are not strong evidence of incorrect mapping. Instead of hard counting of mismatches, we propose a soft counting method that imputes the "possible polymorphism" fractions from all mismatch sites using quality values and an appropriate

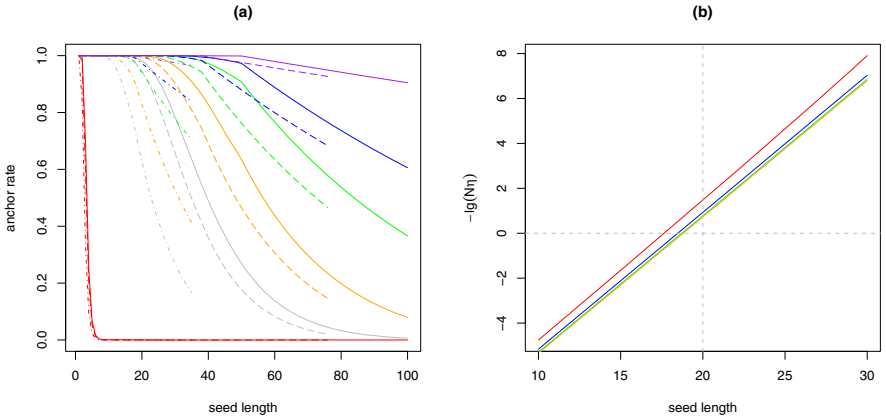

Fig. 7. (a) The probability of finding a perfect k-mer match between two sequences of the same length with respect to k. The solid lines, dashed lines and dotted lines represent the occasions for 100bp, 76bp, 35bp sequences; and the red, gray, orange, green, blue and purple lines respectively correspond to the common match rate 0.25, 0.95, 0.975, 0.99, 0.995, 0.999. **(b)** $-\lg(N\eta)$ vs. seed length, where η is the probability that the read is anchored to a random subsequence of the same length. The red, blue, green, and orange lines correspond to the cases for 35bp, 76bp, 100bp, and 110bp reads respectively. We use it as a measure of specificity to guide seed length selection.

polymorphism rate, aiming at reducing the effect of miscalls on mismatch sites. Consequently, we evaluate the mapping result based on the sum of the imputed "possible polymorphism" fractions. Specifically, according to the setup above, the mismatch rate is $1-p_i = (1-\beta_i)\gamma+\beta_i[(1-\gamma)+\gamma w_i]$. We impute the "possible polymorphism" fractions at mismatch sites as

$$\frac{(1-\beta_i)\gamma}{(1-\beta_i)\gamma+\beta_i[(1-\gamma)+\gamma w_i]},$$

which can well be approximated by $\frac{\gamma}{\beta_i+\gamma}$. If quality scores are available and can be interpreted as probabilities, we have $\beta_i = 10^{-\frac{q_i}{10}}$, see [15,16]. Our statistic is defined to be

$$\sum_{i \text{ at mismatch sites}} \frac{\gamma}{\beta_i+\gamma}.$$

Under the assumption of independence, its distribution is binomial$(n-k,\gamma)$. Consequently, we can convert the statistic score of an alignment into a p-value. In this case, the larger the p-value, the stronger evidence of accepting the mapping.

Since the seed and extension part do not overlap, we can regard them as approximately independent. θ is the type one error probability of the associated test of the hypothesis: the anchor is correct. We can set the significance level α of this test to ensure a reasonable sensitivity. In fact, $\tau(1-\alpha)$ is an upper bound for the sensitivity of SEME. The curves in Fig 6(a) show the sensitivity excluding the factor $(1-\alpha)$ under different settings and can serve as a guidance for seed length selection.

We can similarly calculate the sum of inconsistencies of an extension alignment under the random sequence assumption, and its asymptotic distribution is normal.

The chance that we accept an incorrect anchor should be small as validated by our simulations. Essentially $1 - \theta^*$ is the power of the associated test. For the moment, we use $N\eta$ as a conservative bound of specificity when choosing seed length. As a matter of fact, $N\eta$ is also the average number of anchor places across the whole genome by chance. The curves of $-\lg(N\eta)$ with respect to seed length are shown in Fig 6(b).

3.1 Seed Length Determination

The seed length selection is a trade off between sensitivity and specificity. Shorter seeds increase sensitivity, but may lead to many incorrect anchor places; longer seeds increase specificity, but the seed may be mapped nowhere. To ensure both of them, the seed length should be in a proper range.

According to Fig 6(b), the curves corresponding to different read lengths are close to each other, especially as the size is larger than 76bp. Only when the seed length is chosen to be at least 19 or 20, the average number of anchor positions by chance would be smaller than 1. If we would ensure specificity larger than 0.999, the lower bound should be up to 24 or 25. Of course, this estimate might be conservative because $N\eta$ is a conservative bound of specificity. On the other hand, slightly larger lower bound can help avoid false positive anchors, which are expected to be rejected in the extension stage. The upper bound is chosen according to the sensitivity curve and our tolerance. For 100-bp reads with an average 0.99 match rate, $[20, 32]$ is a proper range of seed length. In practice, the binary search algorithm simply find the maximum exact match length up to 32 nucleotide for each seed. If it is above 20, we then evaluate the significance based on this exact match length.

If the seed is 20-mer, the sensitivity for 35bp reads with a match rate 0.975 is 0.83, while it grows to $1 - 6.95 \times 10^{-4}$ for 100bp reads with a match rate 0.975. If we choose 32-mer seed and assume the match rate is 0.99, then the sensitivity is 0.75, 0.98, and 0.99587 respectively for 35bp, 76bp and 100bp reads. For shorter reads with lower quality, the seed length may even drop to less than 20 to ensure a fair sensitivity. This is the reason why in the early days of NGS, the strategy using single seed of perfect match did not work while it is feasible nowadays as the read length and sequencing quality improves.

4 Results

In our examples, the reference genome is the human genome *hg18*. We report comparisons of SEME with Bowtie2 (Version 2.0.0-beta7) and SOAP2[17,18] on three data sets from NCBI database, each of which includes 2 million reads. To make fair comparisons, we implement SEME by mimicing the parameters in the '–sensitivity' mode for Bowtie2 and '-v4 mode for SOAP2 respectively.

Mapping rates and time are two key measures for evaluating read mappers. We show the comparison of Bowtie2 and SEME on the left in Table 2. For data set 1, the running time of SEME reduces to about 1/4 whereas the mapping rate of SEME is 14.6(88.0-73.4)% higher than that of Bowtie2. Comparing with data set 1, the running time of Bowtie2 for data set 2 is a little more while that of SEME reduces further by a third. This phenomenon is due to the fact that the number of Bowtie2's trial seeds is fixed for

Table 2. Comparison of SEME, Bowtie2 and SOAP2. n is the read length of each data set. The three data sets are from NCBI database, namely, short read archives SRR003196, SRR033622 and SRR054721. They are all generated by the Illumina Platform. Each data set contain two million short reads. Left: Comparison with Bowtie2; right: Comparison with SOAP2

n	Programme	Time(s)	Map rate(%)
76	Bowtie2	508	73.4
	SEME	124	88.0
75	Bowtie2	542	96.1
	SEME	81	98.8
100	Bowtie2	787	95.5
	SEME	95	99.2

n	Programme	Time(s)	Map rate(%)
76	SOAP2	290	39.4
	SEME	164	50.4
75	SOAP2	207	80.5
	SEME	112	87.0
100	SOAP2	261	74.5
	SEME	161	81.7

each short read while that of SEME mainly depends on the quality of each short read, that is, better quality, less trial seeds. Notice that the read lengths of data set 1 and 2 are about the same and the quality of the latter is better. From data set 2 to 3, the read length extends to 100, whereas the quality are similar. We can see that the mapping time of Bowtie2 increases, but SEME remains about the same, which verifies our analysis.

We show the comparisons of SOAP2 and SEME on the right in Table 2. Since the -v4 mode only allow 4 mismatches, mapping rates of both SOAP2 and SEME for all 3 data sets are lower than those in the comparison with BOWTIE2. Not only does the running time of SEME reduces to about one half, but also it has a 7-11% gain in mapping rates.

In sum, compared with Bowtie2, SEME runs 4.1-8.3 times faster depending on quality of data sets; Compared with SOAP2, SEME runs twice faster while the mapping rate of SEME is substantially higher.

5 Discussion

SEME has two key features. The first one is its novel mapping algorithm, which obeys the "seed-and-extend" paradigm. A common approach of the seed stage is to map multiple seeds at the same time and then make them to full alignments. The number of these multiple seeds is usually fixed from read to read. Different from this approach, SEME maps seeds sequentially. The number of seeds need to be mapped depends on the distribution of mismatch sites on the short read. The scan function of SEME efficiently minimizes the average number of trial seeds. In the extension stage, SEME can detect the pattern, position and length of small INDELs by means of auto match function and auto match vectors without enumerating all possible combinations or carrying out local alignment algorithm. Time complexity of the extension stage is linear with respect to the read length.

The second feature is that SEME has its own statistical evaluation of mapping reliability, which is critical for NGS, especially its applications to medicine. Compared to the vast amount of algorithmic development, not much associated statistics was found in the literature so far. A statistical evaluation of a mapping result is justifiable only if the model on which the analysis is based captures the data characteristics and follows the mapping algorithm closely. Our statistical analysis of the "seed-and-extend"

scheme essentially boils down the evaluation of specificity and sensitivity to the matching chance of two n-length sequences, where n is the short read length. We decompose the probabilities into two parts: one corresponds to the seed stage and the other corresponds to the extension stage. Since we stick to perfect match in the seed stage, the calculation of the exact probability is relatively easy, see Lemma. In the extension stage, the sum of "possible polymorphism" fractions can approximately be described by either a binomial or a normal distribution.

The random sequence assumption is definitely far from a perfect description of any common natural genome because it ignores more complicated issues such as duplications and homologs. Appropriate simulations may complement the model-based analysis to some extent. We carried out limited simulations, and the results are quite comparable with the analytical results in terms of the values of τ and r in Equation (3).

SEME is very flexible due to its data structure and sequential scan strategy. Depending on the mapping context, the condition of the short reads and the requirement of the mapping result, we can adjust the seed length, the scan scheme and the upper bound of trial seeds. The optimization of the scan scheme depends on several factors such as the sequencing quality pattern, and we are conducting more investigations. We implement the method in C++, but the process of improving the code is ongoing.

Other than the straightforward mapping problem, we did not elaborate on SEME's variants that we are working on. For example, by encoding and decoding 'C' and 'T' with a common letter, we can use SEME to map short reads and allow methylation sites. However, with this setup, the values of sensitivity and specificity, the seed length need to be re-evaluated. For now, SEME deals with pair-end data by treating them as independent reads. How to integrate information from both ends to speed up mapping is an interesting problem to be considered. The detection of alternative splicing site inside a single seed is a more challenging task. As the Illumina read length goes beyond 160bp, the ideas of SEME described in this report may help solve the problem. Particularly, we emphasize that the statistical evaluation is important for justifying the significant of any new genomic discovery.

Acknowledgements. We thank Dr. Yong Zhang and Dr. Zhixiang Yan from BGI-Shenzhen for their help. We thank Bo Wang and Lixian Yang for their help on programming. This work was supported by the National Center for Mathematics and Interdisciplinary Sciences, CAS, the Program of "One hundred talented people", CAS, and grant 91130008 from Chinese National Science Foundation.

References

1. Li, H.: A survey of sequence alignment algorithms for next-generation sequencing. Brief Bioinformatics 11, 473–483 (2010)
2. Ben, L., et al.: Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. Genome Biology 10, R25 (2009)
3. Li, H., Ruan, J., Durbin, R.: Mapping short DNA sequencing reads and calling variants using mapping quality scores. Genome Res. 18, 1851–1858 (2008)
4. Hui, J., Wing-Hung, W.: SeqMap: mapping massive amount of oligonucleotides to the genome. Bioinformatics 24, 2395–2396 (2008)

5. MiSeq Personal Sequencer - Illumina, http://www.illumina.com/systems/miseq.ilmn
6. Xun, G., Wen-Hsiung, L.: The Size Distribution of Insertions and Deletions in Human and Rodent Pseudogenes Suggests the Logarithmic Gap Penalty for Sequence Alignment. J. Mol. Evol. 40, 464–473 (1994)
7. Ryan, E.M., Christopher, T., et al.: Luttig, An initial map of insertion and deletion (INDEL) variation in the human genome. Genome Res. 16, 1182–1190 (2006)
8. Smith, T.F., Waterman, M.S.: Identification of common molecular subsequences. J. Mol. Biol. 147, 195–197 (1981)
9. Altschul, S.F., Gish, W., Miller, W., Myers, E.W., Lipman, D.J.: Basic local alignment search tool. J. Mol. Biol. 215, 403–410 (1990)
10. Karlin, S., Altschul, S.F.: Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes. Proc. Natl. Acad. Sci. U S A 87, 2264–2268 (1990)
11. Waterman, M.S.: General methods of sequence comparison. Bull. Math. Biol. 46, 473–500 (1984)
12. Waterman, M.S.: Introduction to Computational Biology: Maps, Sequences and Genomes. Chapman & Hall, London (1995)
13. Ross, A.L., Haiyan, H., Waterman, M.S.: Distributional regimes for the number of k-word matches between two random sequences. PNAS 99, 13980–13989 (2002)
14. Warren, J.E., Gregory, R.G.: Statistical Methods in Bioinformatics: An introduction. Springer, New York (2001)
15. Brent, E., Phil, G.: Base-Calling of Automated Sequencer Traces Using Phred. II. Error Probabilities. Genome Res. 8, 186–194 (1998)
16. Ming, L., Magnus, N., Lei, M.L.: Adjust quality scores from alignment and improve sequencing accuracy. Nucleic Acids Research 32, 5183–5191 (2004)
17. Ruiqiang, L., Yingrui, L., Karsten, K., Jun, W.: SOAP: short oligonucleotide alignment program. Bioinformatics 24, 713–714 (2008)
18. Ruiqiang, L., Chang, Y., Yingrui, L., et al.: SOAP2: an improved ultrafast tool for short read alignment. Bioinformatics 25, 1966–1967 (2009)
19. Zaharia, M., Bolosky, W.J., Curtis, K., Fox, A., Patterson, D., Shenker, S., Stoica, I., Karp, R.M., Sittler, T.: Faster and More Accurate Sequence Alignment with SNAP. arXiv:1111.5572 [cs.DS] (2011)