Aligning Flowgrams to DNA Sequences

Marcel Martin and Sven Rahmann

Genome Informatics, Institute of Human Genetics, Faculty of Medicine
University of Duisburg-Essen, Essen, Germany
marcel.martin@tu-dortmund.de, sven.rahmann@uni-due.de

Abstract

A read from 454 or Ion Torrent sequencers is natively represented as a *flowgram*, which is a sequence of pairs of a nucleotide and its (fractional) intensity. Recent work has focused on improving the accuracy of base calling (conversion of flowgrams to DNA sequences) in order to facilitate read mapping and downstream analysis of sequence variants. However, base calling always incurs a loss of information by discarding fractional intensity information. We argue that base calling can be avoided entirely by directly aligning the flowgrams to DNA sequences. We introduce an algorithm for *flowgram-string alignment* based on dynamic programming, but covering more cases than standard local or global sequence alignment. We also propose a scoring scheme that takes into account sequence variations (from substitutions, insertions, deletions) and sequencing errors (flow intensities contradicting the homopolymer length) separately. This allows to resolve fractional intensities, ambiguous homopolymer lengths and editing events at alignment time by choosing the most likely read sequence given both the nucleotide intensities and the reference sequence. We provide a proof-of-concept implementation and demonstrate the advantages of flowgram-string alignment compared to base-called alignments.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems, J.3 Life and Medical Sciences

Keywords and phrases flowgram, sequencing, alignment algorithm, scoring scheme

Digital Object Identifier 10.4230/OASIcs.GCB.2013.125

1 Introduction

Pyrosequencing or Sequencing by Synthesis Pyrosequencing, also sequencing by synthesis, is a technology for DNA sequencing that does not sequence single nucleotides, but one run of nucleotides (*homopolymer*), at a time. There are two commercial sequencing technologies that use this approach: 454 (now owned by Roche) [10, 5] and Ion Torrent's "Ion semiconductor sequencing" (now owned by Life Technologies) [6].

Sequencing by synthesis starts with a single-stranded DNA template with an initial sequencing primer. Nucleotides of a single type are added and extend the primer if the next free bases on the template strand are complementary. The activity of the enzyme that catalyzes this reaction can be measured optically through its intermediate release of pyrophosphate (454). Alternatively, a change in pH value caused by the incorporation of nucleotides into the double strand can be measured directly with a semiconductor chip (Ion Torrent). The intensity of the measured signal is, in principle, proportional to the number of bases incorporated. All remaining free nucleotides are then removed and a different type of nucleotide is added. By cyclically flowing all four nucleotides and measuring the signal intensity, the sequence of the template DNA fragment can be reconstructed.

Using initial key sequences for each read, the signal intensities are normalized such that an intensity of 1.0 represents the incorporation of a single base. Due to the linearity of the

German Conference on Bioinformatics 2013 (GCB'13).
Editors: T. Beißbarth, M. Kollmar, A. Leha, B. Morgenstern, A.-K. Schultz, S. Waack, E. Wingender; pp. 125–135

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

signal, 2.0 represents two bases and so on. This linearity of signal intensity can be maintained up to eight bases on a 454 system [5], but errors can occur at lower intensities.

The sequencing results from both mentioned technologies are natively output not as regular DNA sequences, but as so-called *flowgrams*, which associate each nucleotide homopolymer with its measured fractional intensity. It is therefore possible, for example, that a nucleotide homopolymer was measured at an "intensity of 2.4" (see Section 2).

Information Loss from Base Calling By rounding intensities to the nearest integer, a regular DNA sequence can be inferred (for a thymine at 2.4, this could be `TT` or `TTT`), a step known as *base calling*. Subsequently, standard read-mapping and sequence alignment algorithms can be used to compare the obtained sequence reads with reference sequences.

Recent work has focused on improving base calling from straightforward rounding to the nearest integer towards more elaborate statistical methods based on HMMs [3]. Nevertheless, base calling always incurs a loss of information by replacing the fractional intensity with a sequence of integer length. For example, the distinction between a `C` observed at an intensity of 5.4 vs. an intensity of 4.6 is lost. Both are called as `CCCCC`, but in the first case, alignment to six `C`s is much more plausible than in the latter case.

Previous Work Avoiding Base Calling We put forward the hypothesis that it makes more sense to invent alignment algorithms that directly work on flowgrams, instead of on a base-called sequence. A few publications on flowgram-based alignment already exist, but none clearly separates the two processes of sequence editing and flowgram under- and overcalling.

- Vacic et al. [12] model the distribution of flowgram intensities and derive a probabilistic model to compute the log-odds score that a given flowgram originates from a given genomic sequence. Their software FLAT is intended for mapping sequenced small RNA molecules to a reference and not for aligning diverged DNA sequences, so they do not take into account editing events. We use a similar way of deriving log-odds scores for differences between aligned reference and flow intensity.
- Quince et al. [9] use an algorithm adapted from global alignment [8] to align two flowgrams, first converting the reference sequence into flowspace. The authors' idea is to introduce gaps only in steps of four in order to take into account the cyclic nature of the flow order. The remaining description in the paper is brief, but one can deduce that a single flow is aligned to a homopolymer. It is unclear how editing is handled. The cost function used is $-\log P(f \mid \ell)$, where $P(f \mid \ell)$ is the probability of observing flow intensity f given a homopolymer of length ℓ.
- Lysholm et al. [4] propose a different method of aligning flowgrams, which is an extension of the Smith-Waterman local alignment algorithm [11] and can handle substitutions and indels with affine gap costs. FAAST's alignment is computed between the reference string and the base-called flowgram. Its modified scoring system reduces gap costs at points of uncertain homopolymer lengths.

Our Contributions In contrast to previous work, we do neither convert the reference into flowspace nor the flowgram to a string. Instead, we present the first algorithm that directly aligns a flowgram to a reference sequence, being aware of two processes in between: sequence editing between the reference and the (unknown) sequenced sample, and sequencing errors resulting in imprecise flow intensities.

After stating basic definitions (Section 2), we introduce a dynamic programming algorithm for optimal flowgram-string alignment (Section 3). The key component is a detailed scoring

scheme that models both sequence editing events and flow intensity measurement errors; it is described in detail in Section 4, where we also explain how the scoring parameters can be set to reasonable values. In Section 5, we demonstrate how flowgram-string alignment improves upon aligning a base-called sequence. A discussion and outlook on future work concludes the paper. A proof-of-concept implementation is available as a Python module from `http://www.rahmannlab.de/software`.

2 Basic Definitions and Ideas

Let Σ be the DNA alphabet. Let b^ℓ be the character $b \in \Sigma$ repeated ℓ times. Such a string is called a *homopolymer* of length ℓ.

The output of a 454 or Ion Torrent sequencer for a single read is a sequence of pairs of a nucleotide character and an intensity, called a flowgram [5], which we now define formally.

▶ **Definition 1** (Flow). A *flow* is a pair (b, f), where $b \in \Sigma$ is the *flow character* (or *flow nucleotide*) and $f \in \mathbb{R}_0^+$ is the *flow intensity*.

In analogy to exponentiation, we also write a single flow as $b_i^{f_i}$, that is, as the flow character followed by the flow intensity as a superscript. For example, instead of $(\mathtt{A}, 3.4)$, we write $\mathtt{A}^{3.4}$.

▶ **Definition 2** (Flowgram). A *flowgram* is a finite sequence $F = (F_1, F_2, \ldots, F_m)$ of flows $F_i = (b_i, f_i)$. The *flowgram length* is m. For $k = 1, \ldots, m-1$, we require that $b_i \neq b_{i+1}$.

The four nucleotides are typically added in repeating cycles. While our algorithm does not depend on it, we assume in the following that the order is $(\mathtt{T}, \mathtt{A}, \mathtt{C}, \mathtt{G}, \ldots)$, the typical order used in 454 instruments (that for Ion Torrent is different), and that the first flow character is always `T`. We may also say that a read is in *flowspace* to indicate that it is a flowgram.

Given flowgram $F = (F_1, \ldots, F_m)$, the sequence $F_{j\ldots k} := (F_j, \ldots, F_k)$ is a *subflowgram*. For $j = 1$, it is a *flowgram prefix*, and for $k = m$, it is a *flowgram suffix*.

▶ **Example 3.** A possible measured flowgram for the sequence `TTCGG` is $\mathtt{T}^{2.3}\mathtt{A}^{0.1}\mathtt{C}^{0.9}\mathtt{G}^{1.9}$.

The data output by both 454 and Ion Torrent sequencers contains flowgrams and uses the Standard Flowgram Format (`.sff`). The cycle order is stored once globally, and each flow intensity is stored as a 16-bit unsigned integer value and scaled such that a value of 100 represents an intensity of 1, thus allowing values from 0.00 up to $(2^{16} - 1)/100 = 655.35$.

▶ **Definition 4** (Canonical flowgram). Given a string s, the *canonical flowgram* for s is the flowgram that arises when we substitute all runs of character b of length n with the flow b^n and insert appropriate flows of intensity zero in between or in the beginning in order to get the correct order of nucleotides according to cycle order.

▶ **Example 5.** The canonical flowgram for `ACTT` (using cycle order `TACG`) is $\mathtt{T}^0\mathtt{A}^1\mathtt{C}^1\mathtt{G}^0\mathtt{T}^2$.

▶ **Definition 6** (Canonical DNA sequence). Given a flowgram F, let $\bar{F}$ be the flowgram with each intensity rounded to the nearest integer. The *canonical DNA sequence* for F is the DNA sequence which has the canonical flowgram $\bar{F}$, if it exists, and is undefined otherwise.

Note that a canonical DNA sequence for $F = \mathtt{T}^{1.1}\mathtt{A}^{0.1}\mathtt{C}^{0.4}\mathtt{G}^{0.2}\mathtt{T}^{2.3}$ does not exist, as rounding leads to $\bar{F} = \mathtt{T}^1\mathtt{A}^0\mathtt{C}^0\mathtt{G}^0\mathtt{T}^2$, but `TTT` has a canonical flowgram starting with $\mathtt{T}^3$. (A base caller that is cleverer than rounding [3] calls `TCTT` in this example instead of a non-existing canonical DNA sequence.)

reference r substring t	editing $\longrightarrow$	read/sample s homopolymer b^ℓ	sequencing $\longrightarrow$	flowgram F flow b^f
(known)		(unknown)		(observed)

Figure 1 Differences between an observed flowgram F and a reference sequence r arise from two different processes that cannot be distinguished by the observer: Sequence editing, responsible for differences between the sequenced sample and the reference in databases, and sequencing errors (intensity overcalls and undercalls) incurred during the sequencing process.

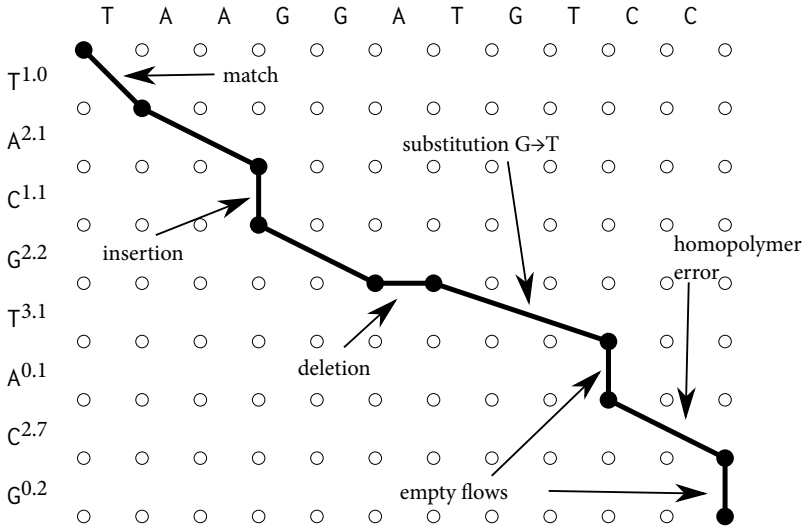

Figure 2 A visualization of the alignment of Example 8. The black path represents the alignment. The path may skip an arbitrary number of columns, but it cannot skip rows.

The problem just illustrated is sufficient motivation to find methods that skip base calling and directly align a flowgram to a reference sequence. Our main idea is to conceptually model a two-stage process (sequence editing, errors during sequencing) within one model and scoring function. It is best visualized with Figure 1.

We align a flowgram directly to a reference, without converting it to a string. Fractional intensities and ambiguous run lengths are resolved at alignment time by choosing the most likely read sequence given flowgram and reference sequence. In contrast to Vacic et al.'s work [12], we also model differences due to editing events. Every observed flow b^f must be explained by a substring t of the reference. The substring and the flow need not necessarily agree: If there is a non-b character in t, then there is a substitution or an insertion, and if the absolute difference between f and $|t|$ is sufficiently large, then there is an insertion or deletion event or a homopolymer error. Thus, a flow b^f can be explained as a sequenced homopolymer b^ℓ (where ℓ is integer), which in turn is an edited version of t (cf. Figure 1).

3 A Flowgram-String Alignment Algorithm

3.1 Alignments

▶ **Definition 7** (Flowgram-string alignment)**.** A *flowgram-string alignment* $\mathcal{F}$ between a flowgram F of length m and a string s of length n is a finite sequence of pairs $\mathcal{F} = (F'_i, t_i)$, where each F'_i is a flow or the space character $(-)$ such that the concatenation of all non-space

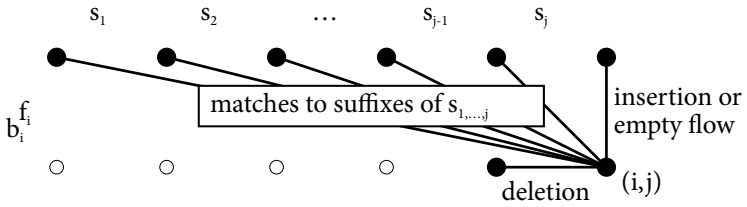

Figure 3 Visualization of different types of edges of the alignment graph and of the recurrence for cell (i, j) in the dynamic programming matrix.

F'_i is the flowgram F, and where the t_i are (possible empty) substrings of s such that their concatenation is equal to s.

▶ **Example 8.** Given are flowgram $F = \mathtt{T}^{1.0}\mathtt{A}^{2.1}\mathtt{C}^{1.1}\mathtt{G}^{2.2}\mathtt{T}^{3.1}\mathtt{A}^{0.1}\mathtt{C}^{2.7}\mathtt{G}^{0.2}$ and the string $s =$ TAAGGATGTCC. Using a notation in which F'_i is written above t_i, separating elements (F'_i, t_i) with a vertical line, a possible alignment is the following (see also Figure 2):

$\mathtt{T}^{1.0}$	$\mathtt{A}^{2.1}$	$\mathtt{C}^{1.1}$	$\mathtt{G}^{2.2}$	–	$\mathtt{T}^{3.1}$	$\mathtt{A}^{0.1}$	$\mathtt{C}^{2.7}$	$\mathtt{G}^{0.2}$
T	AA	ε	GG	A	TGT	ε	CC	ε

We see that flowgram-string alignment can describe all editing events: $\mathtt{T}^{3.1}$ aligned to TGT involves a mismatch (G instead of T); $\mathtt{C}^{1.1}$ aligned to ε means that there is an insertion; and the space aligned to an A is a deletion. We will also see below that, with the proper scoring function, flowgram-string alignment can distinguish between homopolymer errors and insertions. The scoring function will inform us whether the rightmost flow $\mathtt{G}^{0.2}$ aligned to an empty string ε of the reference needs to be interpreted as a homopolymer error of 0.2 or as an insertion. Our alignment algorithm picks the option with the better score.

A flowgram-string alignment describes how (1) editing events and (2) sequencing errors due to over- or undercalling add up to result in an observed flowgram. In contrast to previous flowgram alignment ideas, there is no need to convert the reference to a flowgram or to convert the flowgram into a string. Instead, a flowgram-string alignment describes a direct relationship between flowgram and reference.

3.2 The Flowgram-String Alignment Graph

A flowgram-string alignment can be interpreted as a path through a graph of $(m+1) \times (n+1)$ vertices $(i, j) \in \{0, \ldots, m\} \times \{0, \ldots, n\}$ that has different types of edges with different scores (see Figures 2 and 3). The score of an alignment $\mathcal{F}$ is the sum of the scores of the individual edges used by the alignment, and so finding the optimal alignment is equivalent to finding a highest-scoring path. The following two edge type exist:

- horizontal edges that connect (i, j) to $(i, j-1)$. These represent deletions, i.e., the flowgram indicates that a nucleotide from the reference is missing in the sample. The score $del < 0$ is assigned to these edges.
- vertical and diagonal edges that connect (i, j) to $(i-1, j-k)$ for all $k \in \{0, \ldots, j\}$. For $k = 0$, the edge is vertical and interpreted as either an empty flow aligned to an empty substring, or as an insertion, where the flowgram indicates a homopolymer not present in the reference. These edges use a complex scoring function $v(b, f, t)$ for aligning flow b^f to substring $t = s_{k+1\ldots j}$. This scoring function is central to our method and discussed in detail in Section 4.

3.3 Recurrence

Let (b, f) be a flow and $t \in \Sigma^*$ a string. We assume that scoring parameter del and scoring function $v(b, f, t)$ are available (see Section 4).

Let $S(i, j)$ be the optimal score between the length-i prefix of flowgram F of total length m and the length-j prefix of string s of total length $n = |s|$. The recurrence for $S(i, j)$ follows from the structure of the alignment graph, in which the optimal flowgram-string alignment is a highest-scoring path, analogously to standard global alignment. Other variants (local, free end gaps, etc.) are possible; for ease of exposition, we focus on the global case. We have

$$\begin{aligned} S(0, j) &= j \cdot del, \\ S(i, 0) &= \sum_{k=1}^{i} v(b_k, f_k, \varepsilon), \\ S(i, j) &= \max \left\{ \begin{array}{l} S(i, j-1) + del, \\ \max\limits_{k=0,\ldots,j} \left(S(i-1, k) + v(b_i, f_i, s_{k+1\ldots j}) \right) \end{array} \right\}. \end{aligned} \tag{1}$$

The two cases for $S(i, j)$ correspond to the two types of edges. The inner maximization corresponds to the vertical and diagonal edges, in which the score of aligning the current flowgram to all suffixes of $s_{1\ldots j}$ (including the empty suffix for case $k = j$) is found. It is the main difference to regular global alignment. With dynamic programming, $S(m, n)$ can be computed in time $\mathcal{O}(mn^2)$, assuming v can be evaluated in constant time.

4 Scoring

The score $v(b, f, t)$ for pairing flow (b, f) with string t must take into account two different processes that cannot be distinguished by an observer (Figure 1). First, editing events occur that change a substring t of the reference into b^ℓ, but ℓ is unknown. Second, an intensity f is measured for b^ℓ. We score the first process by $s_{\text{edit}}(b, \ell, t)$, which is the score of an optimal alignment between t and b^ℓ. The score $\sigma(f, \ell)$ is assigned to measuring intensity f for a homopolymer run of length ℓ; we assume that it does not depend on the nucleotide b.

Since ℓ is unknown, to obtain $v(b, f, t)$ we maximize over all possible lengths in order to pick the most plausible explanation:

$$v(b, f, t) := \max_{\ell=0,1,2,\ldots} \left(s_{\text{edit}}(b, \ell, t) + \sigma(f, \ell) \right) \tag{2}$$

As we will see in the two following subsections, this potentially infinite maximization is in fact finite, since a value of $\ell \gg \max\{|t|, f\}$ will yield a strongly negative score in both terms and cannot achieve the maximum. In practice, positive scores are only obtained if $f \approx |t|$ for a choice of ℓ close to both f and $|t|$.

To reconstruct the most plausible process to flow b^f via homopolymer b^ℓ from sequence t, we also store the value of ℓ maximizing $v(b, f, t)$ in (2),

$$L(b, f, t) := \operatorname{argmax}_{\ell=0,1,2,\ldots} \left(s_{\text{edit}}(b, \ell, t) + \sigma(f, \ell) \right). \tag{3}$$

It is this (unknown but inferred) value of $\ell = L(b, f, t)$ that links the two processes of sequence editing and sequencing shown in Figure 1.

As we will show, the score $v(b, f, t)$, and hence $L(b, f, t)$, depends on b and t only through the number $e = e(t, b)$ of characters in t that are equal to b and the number $\bar{e} = \bar{e}(t, b)$ of characters different from b (see Section 4.1). Therefore we can write $v(b, f, t) = v'(f, e, \bar{e})$ and

$L(b, f, t) = L'(f, e, \bar{e})$. A table of L' (or tables of both L' and v' for realistic flow intensities $f \in \{0.00, 0.01, \ldots, 9.99\}$ and values of e and $\bar{e}$, both in $\{0, \ldots, 9\}$, i.e., 100 000 values overall, is pre-computed. As the recurrence (1) considers different substrings t that differ in length by 1, the bs in t can be counted in amortized constant time for each t, so each value of $v(b, f, t)$ is available in constant time. The non-tabulated rare cases can be computed on demand without measurably affecting the running time.

4.1 Scoring of Editing Events

In this section, we derive the edit score $s_{\text{edit}}(b, \ell, t)$ to align two sequences: b^ℓ and t. This is, in fact, a classical sequence alignment problem, with the special property that one sequence b^ℓ is a homopolymer. Instead of using a standard global alignment algorithm every time when s_{edit} is called, we can give a closed formula because of the special structure.

We assume that scores for insertion (*ins*), deletion (*del*), mismatch (*mis*) and match (*mat*) are available and fulfill $ins, del < mis < 0 < mat$.

Let e and $\bar{e} = |t| - e$ be the number of characters in t that are equal to b and not equal to b, respectively. If $|t| = \ell$, the score is composed of only match (e times) and mismatch ($\bar{e}$ times) scores. If t is longer than ℓ, then $|t| - \ell$ characters must be deleted from t to obtain length ℓ, and it is advantageous to delete only non-b characters, as long as there are any. If t is shorter than ℓ, we have e matches and $\bar{e}$ mismatches, and $\ell - |t|$ characters must be inserted into t. Thus the score for aligning t to b^ℓ can be expressed as

$$s_{\text{edit}}(b, \ell, t) = \begin{cases} e \cdot mat + \bar{e} \cdot mis & \text{if } \ell = |t|, \\ e \cdot mat + \bar{e} \cdot mis + (\ell - |t|) \cdot ins & \text{if } \ell > |t|, \\ \min\{e, \ell\} \cdot mat + \max\{\ell - e, 0\} \cdot mis + (|t| - \ell) \cdot del & \text{if } \ell < |t|. \end{cases}$$

The parameter values for *mat*, *mis*, *ins*, *del* must be compatible with the scores for scoring flow intensities f against substring lengths ℓ, which we discuss next. We come back to choosing appropriate values in Section 4.3.

4.2 Scoring of Flow Intensities Against Substring Lengths

Here we describe how to set the scores $\sigma(f, \ell)$ for scoring the event that a flow of intensity f is aligned to a DNA sequence of length ℓ. Our approach is similar to that of Vacic et al. [12], but we go further by analyzing the resulting empirical scores parametrically.

Intuitively, the score should be positive if $f \approx \ell$ and drop into the negative range when $|f - \ell|$ gets large. A consistent set of score values is obtained by using log-odds scores [2, 7], having their roots in the theory of score matrices for amino acids, such as the famous PAM matrices [2]. There the score Σ_{ij} between amino acids i and j is computed as the log-odds $\Sigma_{ij} = \log(P_{ij}/(\pi_i \cdot \pi_j))$, where P_{ij} is the probability of observing i and j paired in an alignment and π_i, π_j are the background frequencies of amino acids i, j, respectively. Moreover, the joint probabilities P_{ij} depend on the divergence time t of the aligned sequences, and so different score matrices $\Sigma_{ij}^{(t)}$ are used for differently diverged sequences.

Here we follow a similar idea for deriving scores for evaluating differences between f and ℓ. We estimate frequencies from (assumedly correctly) aligned flowgrams to DNA sequences. For ease of exposition, we do not discuss different divergence times, and we assume that the flowgrams have been obtained from the DNA reference by sequencing, or at least from a very closely related reference sequence.

Given a large number of such aligned flowgram-DNA alignments, we construct a count matrix $C = (C_{f,\ell})$ for all reasonable genomic lengths $\ell \in \{0, 1, 2, \ldots\}$ and flow intensities

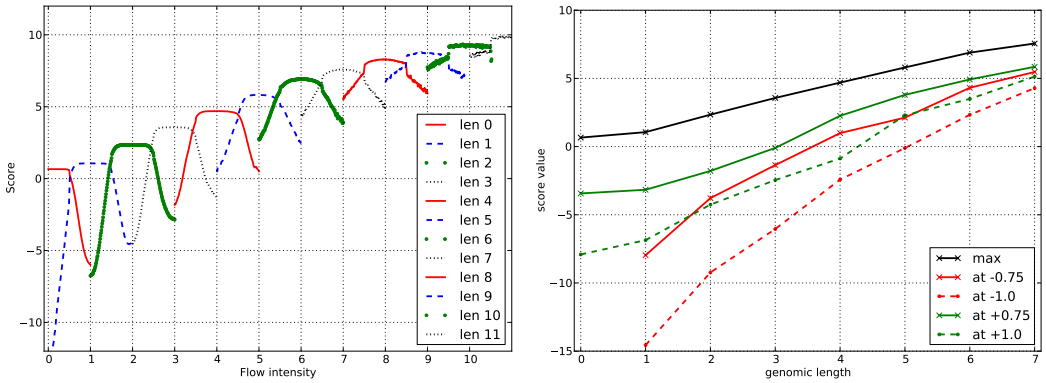

Figure 4 Left: Empirically determined score functions $\sigma(f, l)$ for each genomic sequence length ℓ (see legend) from carefully crafted alignments of flowgrams against an *Arabidopsis* reference. Right: As each function on the left panel can be described by a piecewise affine function with three components, one of them constant, we estimated five parameters from five characteristic score values: for each length ℓ, the score values at $f \in \{\ell - 1.0, \ell - 0.75, \ell, \ell + 0.75, \ell + 1.0\}$. The plot shows these five score values as a function of ℓ.

$f \in \{0.00, 0.01, 0.02, \ldots, 1.00, \ldots\}$, such that $C_{f,\ell}$ counts the number of times we observe a flow of intensity f aligned to a genomic sequence of length ℓ. We obtain a joint probability matrix $P = (P_{f,\ell})$ by dividing C through the sum of its entries. Background frequencies $\pi = (\pi_\ell)$ for genomic lengths are obtained as marginal probabilities $\pi_\ell = \sum_f P_{f,\ell}$, and similarly background frequencies $\tau = (\tau_f)$ for flow intensities. The score component for aligning a flow of intensity f to homopolymer of length ℓ is defined in units of nats as

$$\sigma(f, \ell) := \log \frac{P_{f,\ell}}{\tau_f \cdot \pi_\ell}.$$

To obtain such scores, we used three `.sff` files containing *Arabiopsis* reads (from an unspecified strain), provided by the chair of Genome Research at Bielefeld University. To measure only the effects of homopolymer errors, only reads aligning close-to-perfectly to the *A. thaliana* reference sequence were considered further, and the empirical joint distribution of flow intensities f and homopolymer lengths ℓ was tabulated where $f \in [\ell - 1, \ell + 1]$. The resulting scores are shown in Figure 4, one curve for each ℓ with sufficient data. Unsurprisingly, the maximum score occurs at flow $f = \ell$ for each ℓ. More remarkably, the score stays almost constant at the same level in the interval $f \in [\ell - 0.5, \ell + 0.5]$. At $\ell \pm 0.5$, there appears to be a sudden drop in the scoring function, beyond which we can observe an affine-linear course in the intervals $[\ell - 1.0, \ell - 0.5]$ and $[\ell + 0.5, \ell + 1.0]$. Therefore, for each ℓ, the score function can be described by five parameters, namely the values of $S_{\ell,f}$ for $f \in \{\ell - 1.0, \ell - 0.75, \ell, \ell + 0.75, \ell + 1.0\}$. Scores at other values of f are obtained by linear resp. constant interpolation (or extrapolation outside the 1.0-neighborhood). The parameters for lengths $\ell \leq 7$ are shown in Figure 4 (right). As empirical data becomes sparser for larger ℓ, it is advisable to extrapolate the parameters instead of relying on data.

In summary, we implement $\sigma(f, \ell)$ for each ℓ as a piecewise affine function consisting of three components, given by the empirically determined parameters shown in Figure 4 (right).

	(A)	(B)
original reference r	TAAGAAC	TAAACACCCGG
mutated sample s	TAAAAAC	TAAAACCCCGG
flowgram F	$\mathtt{T}^{0.9}\mathtt{A}^{4.4}\mathtt{C}^{1.1}$	$\mathtt{T}^{0.9}\mathtt{A}^{3.4}\mathtt{C}^{3.4}\mathtt{G}^{2.0}$
base-called	TAA-AAC	TAAA--CCCGG
reconstructed	TAAAAAC	TAAAACCCCGG

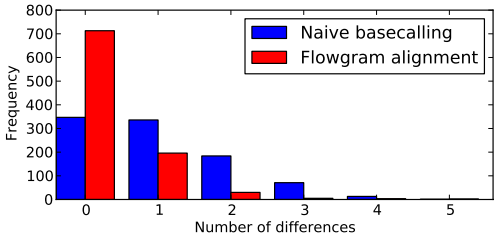

Figure 5 Left: Example errors made by alignment after base calling in contrast to flowgram-string alignment: (A) The G → A substitution is mistakenly reported as a 1 bp deletion because of the low flow value. For flowgram-string alignment with the surrounding context, the case that AAGAA generated $\mathtt{A}^{4.4}$ is plausible. (B) Similarly, but slightly more complex, the CA → AC flip is mistakenly reported as a 2 bp deletion after base calling. For our method, however, two mismatches plus small intensity errors are more plausible than two deletions. Right: Histograms of the number of differences due to sequencing and base-calling errors for naive base calling (blue) and our method (red). Note how the red distribution is shifted towards zero.

4.3 Parameters for Editing Events

It remains to appropriately set the match, mismatch, insertion and deletion score parameters *mat*, *mis*, *ins* and *del*, respectively. These depend on the assumed degree of divergence of the sequenced sample and the reference and can be obtained by (approximate) log-odds.

Assuming 3% divergence (i.e., 97% matches, as opposed to about 30% in alignments of random sequences), and rare insertions/deletion with a rate of 1/3000, it is reasonable to use

- $mat \approx \log(0.97/0.3) = 1.173 \approx 1.2$,
- $mis \approx \log(0.03/0.7) = -3.1498 \approx -3.1$,
- $ins = del \approx \log\big((1/3000)/C\big) \approx -8.0$ with some $C \approx 1$.

These are the scores that we use for evaluation; other assumptions will result in different scores. It is important to use the same logarithm (and scaling, if any) as for $\sigma(f, \ell)$ in order to keep both score components compatible.

5 Evaluation

Before we evaluate flowgram-string alignment against base-called alignment, let us illustrate typical miscalls made by base calling. Obviously, the most common case is that a homopolymer length is simply off by 1 because of rounding in the wrong direction. This can always be corrected by post-processing the alignments. However, there are more complex errors, such as spurious indels in the middle or between two homopolymers, as illustrated in Figure 5 (left).

We now demonstrate that flowgram-string alignment reduces the number of differences between observed sequence and reference that are due to sequencing errors, but leaves actual mutation events untouched (see Figure 1). We simulate DNA fragments of *E. coli K12* (NC_000913); call this the *original* data. We introduce mutations by adding 3% substitutions and 0.05% indels (*mutated* data). Then the 454 sequencing process is simulated with *flowsim* [1]. Reads in the resulting `.sff` file are base-called by rounding flow intensities (*basecalled*) and aligned to the original sequence. Alternatively, we use our flowgram alignment algorithm to align each flowgram to the *original* sequence. During the process, the most likely base-space *mutated* sequence is reconstructed using function $L(b, f, t)$ from (3) (*reconstructed*).

All differences between *mutated* and *basecalled* are necessarily due to sequencing errors and wrong base calls. Similarly, all differences between *mutated* and *reconstructed* are due to

sequencing errors and errors by our alignment method. Figure 5 compares histograms of the number of differences, measured by unit-cost edit distance. The differences are considerably reduced for flowgram-string alignment in comparison to base-calling: The distribution is shifted towards the left side. Thus, flowgram alignment is able to distinguish editing events and true mutations.

6 Discussion and Conclusion

We presented a dynamic programming alignment algorithm that optimally aligns flowgrams output by Roche/454 or Ion Torrent sequencers to DNA reference sequences directly, without explicit base calling. Our approach can also be interpreted as calling bases *conditional* on the reference we align to, i.e., doing both steps at the same time instead of sequentially. Our algorithm is based on a two-stage process model (Figure 1) that explains both sequence editing and homopolymer sequencing errors. In particular, in the process, we can reconstruct the most plausible homopolymer length ℓ for each flow b^f and thus separate flow intensity over- and under-calling from sequence editing. Our method is the first one that cleanly separates the two processes.

A major challenge is to design a scoring scheme for flowgram-DNA alignment that is of low complexity (i.e., has few parameters) and statistically well-founded. We here started from a classical log-odds framework [2] that was also used by Vacic et al. [12]. Going a step further, we noted that for each length ℓ, the score function has a simple three-component piecewise affine form that can be described by only five parameters. This yields the first low-complexity scoring scheme for directly aligning 454 flowgrams to DNA sequences.

There are several ways to extend this work in the future. For example, finding a more robust way to estimate the divergence rate between reference and sample than guessing it before computing alignments would be of interest. On the practical side, several optimizations of the basic alignment algorithm are possible, improving the running time from $O(mn^2)$ to $O(mn)$ by restricting the considered predecessors in each node (i, j) of the alignment graph (cf. Figure 3). It is clear that for a flow b^f, the best choices for t and ℓ have $|t| \approx \ell \approx f$. Extending the algorithm to be able to use affine gap costs would be of high practical relevance. This is not entirely trivial, as gaps could extend over several flows, which in the current model can only be considered separately.

Our approach should of course work on Ion Torrent datasets as well, using a different scoring function. The 454 technology can be used for bisulfite amplicon sequencing to determine CpG methylation. The resulting datasets contain long `T`- or `A`-homopolymers after bisulfite conversion and have different characteristics than standard 454 datasets. Thus it is of interest to estimate scoring parameters for this application.

Acknowledgements We thank Bernd Weisshaar (Genome Research, Bielefeld University) for providing `.sff` files from which we estimated the scoring function parameters.

References

1 Susanne Balzer, Ketil Malde, Anders Lanzén, Animesh Sharma, and Inge Jonassen. Characteristics of 454 pyrosequencing data–enabling realistic simulation with flowsim. *Bioinformatics*, 26(18):i420–i425, September 2010.

2 M. Dayhoff, R. Schwartz, and B. Orcutt. A model of evolutionary change in proteins. *Atlas of Protein Sequences and Structure*, 5:3345–352, 1978.

3 David Golan and Paul Medvedev. Using state machines to model the Ion Torrent sequencing process and to improve read error rates. *Bioinformatics*, 29(13):i344–i351, July 2013.

4 Fredrik Lysholm, Björn Andersson, and Bengt Persson. FAAST: Flow-space assisted alignment search tool. *BMC Bioinformatics*, 12:293, 2011.

5 M. Margulies *et al.* Genome sequencing in microfabricated high-density picolitre reactors. *Nature*, 437(7057):376–380, September 2005.

6 Barry Merriman, Ion Torrent R&D Team, and Jonathan M. Rothberg. Progress in Ion Torrent semiconductor chip based sequencing. *Electrophoresis*, 33(23):3397–3417, December 2012.

7 Tobias Müller, Sven Rahmann, and Marc Rehmsmeier. Non-symmetric score matrices and the detection of homologous transmembrane proteins. *Bioinformatics*, 17(Suppl.1):S182–S189, 2001.

8 Saul B. Needleman and Christian D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443–453, March 1970.

9 Christopher Quince, Anders Lanzén, Thomas P. Curtis, Russell J. Davenport, Neil Hall, Ian M. Head, L Fiona Read, and William T. Sloan. Accurate determination of microbial diversity from 454 pyrosequencing data. *Nature Methods*, 6(9):639–641, September 2009.

10 M. Ronaghi, S. Karamohamed, B. Pettersson, M. Uhlén, and P. Nyrén. Real-time DNA sequencing using detection of pyrophosphate release. *Analytical Biochemistry*, 242(1):84–89, November 1996.

11 T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1):195–197, March 1981.

12 Vladimir Vacic, Hailing Jin, Jian-Kang Zhu, and Stefano Lonardi. A probabilistic method for small RNA flowgram matching. *Pacific Symposium on Biocomputing*, pages 75–86, 2008.