Read Mapping Algorithms for Single Molecule Sequencing Data

Vladimir Yanovsky[1], Stephen M. Rumble[1] and Michael Brudno[1,2]

[1]Department of Computer Science and
[2]Donnelly Centre for Cellular and Biomolecular Research
University of Toronto
{volodyan,rumble,brudno}@cs.toronto.edu

Abstract. Single Molecule Sequencing technologies such as the Helioscope simplify the preparation of DNA for sequencing, while sampling millions of reads in a day. Simultaneously, the technology suffers from a significantly higher error rate, ameliorated by the ability to sample multiple reads from the same location. In this paper we develop novel rapid alignment algorithms for two-pass Single Molecule Sequencing methods. We combine the Weighted Sequence Graph (WSG) representation of all optimal and near optimal alignments between the two reads sampled from a piece of DNA with k-mer filtering methods and spaced seeds to quickly generate candidate locations for the reads on the reference genome. We also propose a fast implementation of the Smith-Waterman algorithm using vectorized instructions that significantly speeds up the matching process. Our method combines these approaches in order to build an algorithm that is both fast and accurate, since it is able to take complete advantage of both of the reads sampled during two pass sequencing.

1 Introduction

Next generation sequencing (NGS) technologies are revolutionizing the study of variation among individuals in a population. While classical, Sanger-style sequencing machines were able to sequence 500 thousand basepairs per run at a cost of over $1000 per megabase, new sequencing technologies, such as Solexa/Illumina and AB SOLiD can sequence 4 billion nucleotides in the same amount of time, at the cost of only $6 per megabase. The decreased cost and higher throughput of NGS technologies, however, are offset by both a shorter read length and a higher overall sequencing error rate.

Most NGS technologies reduce the cost of sequencing by running many sequencing experiments in parallel. After the input DNA is sheared into smaller pieces, DNA libraries are prepared by a Polymerase Chain Reaction (PCR) method, with many identical copies of the molecules created, and attached to a particular location on a slide. All of the molecules are sequenced in parallel using sequencing by hybridization or ligation, with each nucleotide producing a specific color as it is sequenced. The colors for all of the positions on the slide are recorded via imaging and are then converted into base calls.

In this paper we address a specific type of NGS technology – Single Molecule Sequencing (SMS). While the origins of SMS date back to 1989 [3], it is only now becoming practical. The Heliscope sequencer, sold by Helicos, is the first commercial product that allows for the sequencing of DNA with SMS. In a recent publication, the complete genome of the M13 virus was resequenced with the Heliscope [4]. The key advantage of the SMS methods over other NGS technologies is the direct sequencing of DNA, without the need for the PCR step described above. PCR has different success rates for different DNA molecules, and introduces substitutions into the DNA sequence as it is copied. Additionally, the direct sequencing of DNA via SMS significantly simplifies the preparation of DNA libraries. SMS methods should also lead to more accurate estimates of the quantity of DNA which is required for detection of copy number variations and quantification of gene expression levels via sequencing. SMS technologies have very different error distribution than standard Sanger-style sequencing. Because only one physical piece of DNA is sequenced at a time, the sequencing signal is much weaker, leading to a large number of "dark bases": nucleotides that are skipped during the sequencing process leading to deletion errors. Additionally, a nucleotide could be mis-read (substitution errors), or inserted, however these errors are more typically the result of imaging, rather than chemistry problems, and hence are rarer. A full description of SMS errors is in the supplementary material of [4].

One advantage of the Heliscope sequencer, as well as several other proposed SMS technologies (e.g. Pacific Biosciences' expected method) is the ability to read a particular piece of DNA multiple times (called multi-pass sequencing). The availability of multiple reads from a single piece of DNA can help confirm the base calls, and reduce the overall error rate. In practice these methods usually use two passes, as this offers a good tradeoff between error rate and cost. In this paper we introduce a novel rapid alignment algorithm for multi-pass single molecule sequencing methods. While we will use the most common case of two-pass sequencing to describe our algorithms, they can be easily extended to a larger number of passes. We combine the Weighted Sequence Graph (WSG) [11] representation of all optimal and near optimal alignments between the two reads sampled from a piece of DNA with k-mer filtering methods [9] and spaced seeds [1] to quickly generate candidate locations for the reads on the reference genome. We also propose a novel, fast implementation of the Smith-Waterman algorithm [12] using vectorized instructions that significantly speeds up the matching process. Our algorithms are implemented as part of the SHort Read Mapping Package (SHRiMP). SHRiMP is a set of tools for mapping reads to a genome, and includes specialized algorithms for mapping reads from popular short read sequencing platforms, such as Illumina/Solexa and AB SOLiD. SHRiMP is freely available at **http://compbio.cs.toronto.edu/shrimp**.

2 Algorithms

In this section we will describe the alignment algorithms we use for mapping two reads generated from a single DNA sequence with SMS to a reference genome. We will start (Sections 2.1 and 2.2) by reviewing how the Weighted Sequence Graph data structure [11] can be used to speed up the inherently cubic naive alignment algorithm to near-quadratic running time. We will then demonstrate how to combine the WSG alignment approach with standard heuristics for rapid alignment, such as seeded alignment and k-mer filters in Section 2.3. Finally, we will describe our implementation of the Smith-Waterman alignment algorithm with SSE vector instructions to further speed up our algorithm, making it practical for aligning millions of reads against a long reference genome.

2.1 Alignment with Weighted Sequence Graphs

Given unlimited computational resources, the best algorithm for mapping two

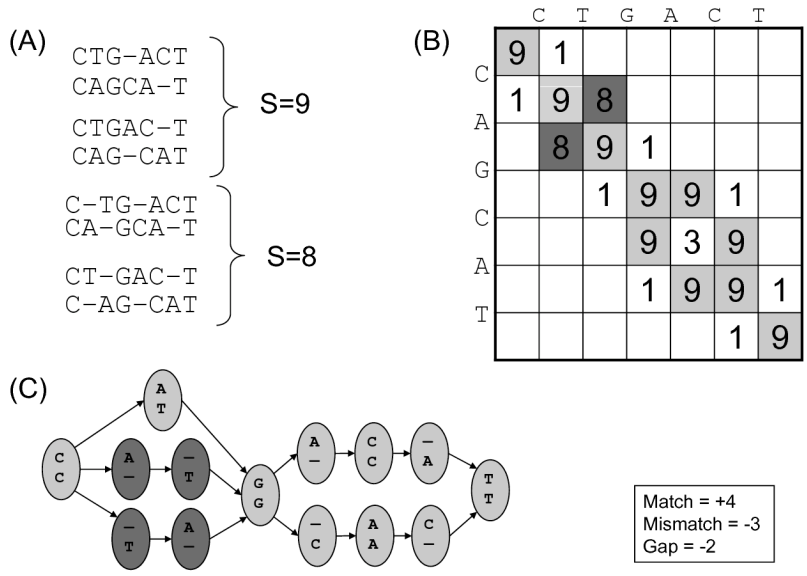

Fig. 1. An example of a WSG graph representing 1-suboptimal alignments of CTGACT with CAGCAT. **(A)**. Optimal alignments of score 9 and two 1-suboptimal of score 8; two more 1-suboptimal alignments are possible. **(B).** Cells of the dynamic programming matrix such that the best path through them is 9 are shaded in light grey, if it is 8 – in dark grey. **(C).** WSG corresponding to at worst 1-suboptimal alignments.

reads sampled from the same location to the reference genome is full three-way alignment. This algorithm would require running time proportional to the

product of the sequence lengths. Because the reads are typically short (∼30bp), the overall running time to map a single read pair to a genome of length n may be practical ($30 * 30 * n = 900n$), however the naive algorithm will not scale to aligning millions of reads that an SMS platform produces every day.

An alternative approach, suggested in [4], is to align the two reads to each other, thus forming a profile, which could then be aligned to the reference genome. This approach, akin to the standard "progressive" alignment approaches used, e.g., in CLUSTALW [2], has the advantage of significantly decreased runtime, as the running time becomes proportional to the product of the lengths of the genome and the longer read (∼30 times the length of the genome), however, because the profile only incorporates a single optimal alignment, it loses important information about possible co-optimal and suboptimal alignments between the sequences. For example, Figure 1 demonstrates two short sequences with two optimal alignments, as well as 4 more near-optimal ones. While the total number of optimal alignments between two sequences could be exponential in their lengths, Naor and Brutlag [7] and Hein [6] have suggested that all of these alignments can be represented compactly using a directed acyclic graph (DAG). Furthermore, Schwikowski and Vingron [11] have shown how to generalize the standard sequence alignment paradigms to alignment of DAGs, which they call weighted sequence graphs. Their original algorithm was stated in the context of the tree alignment problem, but is easily generalizable to any progressive alignment scoring scheme. Here we reintroduce the idea of representing alignments as graphs, and extend it for the SMS mapping problem.

Definitions The following definitions will prove useful:

- Given the sequence S of length $|S|$ over an alphabet $\Sigma = \{A, C, T, G\}$, we refer to its ith symbol as $S[i]$
- We say that a sequence S' over alphabet $\overline{\Sigma} = \Sigma \bigcup \text{`}-\text{`}$, where character `-` is called a gap, *spells* S if removing the gaps from S' results in sequence S. When this does not cause ambiguity, we will write S instead of S'.
- A k-mer is any sequence of length k.
- When addressing matrix indices, we use a notations "$i_1 : i_2$", "$: i_2$", "$i_1 :$" and ":" to represent the sets of indices j such that $i_1 \leq j \leq i_2$, $j \leq i_2$, $i_1 \leq j$ and all possible indices respectively. For example, given matrix M of size $K \times N$ we shall denote the i'th row of the matrix by $M[i;:]$ and its j'th column by $M[:;j]$.
- A *global alignment* of 2 sequences S_1 and S_2 over alphabet Σ is defined as a matrix $M = [2 \times N]$ such that $M[i;:]$ spells S_i for $i = 1, 2$.
- Given a real-valued score function $sc : \overline{\Sigma} \times \overline{\Sigma} \mapsto \Re$ we define the score SC of the alignment $M = [2 \times N]$ as

$$SC(M) = \Sigma_{j=1}^{N} sc(M[:;j])$$

- The *global alignment problem* is to find an alignment M such that $SC(M)$ is maximum. It is known [8] that for any cost function sc satisfying condition

$sc(-,-) < 0$ the problem is well-defined though its solution is not necessarily unique. We denote the maximum score with Opt.
- We call an alignment M ε-suboptimal if $SC(M) \geq Opt - \varepsilon$.

While we defined the alignment for two sequences, the problem for more than two sequences is defined similarly. In particular, the problem we consider in this work – an alignment of a pair of sequences S_1 and S_2 to a reference sequence R – is to find a high scoring alignment matrix $M = [3 \times N]$ such that $M[i;:] = S_i$ for i=1 and 2 respectively and $M[3;:]$ is a substring of R.

Representing (Sub)-optimal Alignments in a Graph A sequence S can be represented by a labeled directed graph $G'(S) = (V, E)$ with $|S| + 1$ nodes $\{s = V[0] \cdots V[|S|] = t]\}$ and $|S|$ edges: for every $0 \leq i < |S|$ there is an edge $V[i] \rightarrow V[i+1]$ labeled with $S[i]$ – the i'th symbol of sequence S; vertices s and t will be called the source and the sink. We obtain the graph $G(S)$ from graph G' by adding a self loop edge $V[i] \rightarrow V[i]$ labeled with a gap symbol '–' for every i. There is a one-to-one correspondence between the sequences over alphabet $\overline{\Sigma}$ spelling sequence S and the sequences produced by reading edge labels along the paths in $G(S)$ from s to t.

Given two strings, S_1 and S_2 and two graphs $A = G(S_1)$ and $B = G(S_2)$, their cross product $A \times B$ is defined as a graph with the vertex set $V = \{(v_A, v_B)\}$, for any v_A vertex of A and v_B vertex of B, and edge set $E = \{v_1 \rightarrow v_2\}$, for any $v_1 = (v_A^1, v_B^1)$ and $v_2 = (v_A^2, v_B^2)$, such that there is an edge $v_A^1 \rightarrow v_A^2$ in A and an edge $v_B^1 \rightarrow v_B^2$ in B. The edges of all graphs considered in this work will be labeled with strings over alphabet $\overline{\Sigma}$; the edge $v^1 \rightarrow v^2$ of the cross product graph will receive a label which is the concatenation of the labels of an edge $v_A^1 \rightarrow v_A^2$ in A and an edge $v_B^1 \rightarrow v_B^2$ in B.

This cross product graph, sometimes referred to as an edit graph, corresponds to the Smith-Waterman dynamic programming matrix of the two strings. It is easy to prove that there is a one-to-one correspondence between the paths from *source* $s = s_1 \times s_2$ to *sink* $t = t_1 \times t_2$ and alignments: if the sequence of edges along the path is $e_0 \cdots e_{N-1}$, then the corresponding alignment M is defined as $M(:;j) = label(e_j)$ for all values of j – recall that $label(e)$ is a pair of symbols. Note that the only cycles in the cross-product graph we just defined are the self loops labeled with '– –' at every node of the graph.

Now a *WSG* is defined as a subgraph of the cross-product graph such that its vertex set V contains vertices s, t and for every vertex $v \in V$ it is on some path from s to t. Intuitively, one may think of the WSG as a succinct representation of a set of alignments between the two strings. We use WSGs in Section 2.2 for the purpose of representing all high-scoring alignments.

2.2 Alignment of Read Pairs with WSG

During two-pass DNA sequencing, two reads are produced from every fragment of DNA. Both of the reads may have sequencing errors, the most common of which are skipped letters. These errors are nearly ten-fold more common than

mis-calls or insertions [4]. Our algorithm for aligning two-pass SMS reads is based on the intuition that in a high-scoring three-way alignment of a pair of reads, S_1 and S_2, to a reference sequence, the alignment of sequences S_1 and S_2 to each other should also be high-scoring. The algorithm proceeds as follows:

- Build a cross-product graph $G' = (V, E)$ representing the set of all possible alignments of S_1 and S_2, s is the source of G' , t is the sink.
- For every edge $e = u \rightarrow v$ in G' we compute the score of the highest scoring path from source s to sink t through the edge e; we denote this score as $w(e)$, the weight of the edge. To do this we use dynamic programming to compute the scores of the highest scoring paths from s to every vertex of G' and of the highest scoring paths, using the edges of G' reversed, from t to every vertex of G. Time complexity of this step is $O(E)$.
- For a given suboptimality parameter ε, we build WSG G_2 from G' by discarding all edges such that

$$w(e) < Opt - \varepsilon$$

  where Opt denotes the score of the highest scoring path from s to t in G'. Observe that while all ε-suboptimal alignments of S_1 and S_2 correspond to paths from s to t, in G_2, the converse is not true: not all paths from s to t are ε-suboptimal.
- Align WSG G_2 to the reference genome: compute the cross product of G_2 and the linear chain that represents the reference sequence, obtaining a WSG spelling all possible three way alignments M, such that $M[1:2;:]$ is one of the alignments, represented by G_2. Finally, use dynamic programming (similar to Smith and Waterman [12]) to search this graph for the locally optimal path. The score of this path is the score of mapping the pair of reads at the given location.

While the use of the WSG leads to a significant speedup compared with the full 3-dimensional dynamic programming, this is still insufficient for the alignment of many reads to a long reference genome, as the resulting algorithm is at best quadratic. In the subsequent section we combine the WSG alignment approach with standard heuristics for rapid sequence alignment in order to further improve the running time.

2.3 Seeded Alignment Approach for SMS Reads

Almost all heuristic alignment algorithms start with seed generation – the location of short, exact or nearly exact matches between two sequences. Whenever one or more seeds are found, the similarity is typically verified using a slow, but more sensitive alignment algorithm. The seeds offer a tradeoff between running time and sensitivity: short seeds are more sensitive than long ones, but lead to higher running times. Conversely, longer seeds result in faster searches, but are less sensitive. Our approach for SMS seeded alignment differs in two ways from standard methods. First, we generate potential seeds not from the observed

reads, but rather from the ϵ-suboptimal WSGs representing their alignments. Second, we combine spaced seeds [1] with k-mer-filtering techniques [9] to further speed up our algorithm.

Seed Generation Intuitively, given a read pair and an integer k, the seeds we want are k-long substrings (k-mers) likely to appear in the DNA segment represented by the pair and segment's matching location on the reference sequence. While k-mers present in each individual read may not match the genome directly due to the deletion errors introduced during the sequencing process, it is much less likely that both reads have an error at the same location. Consequently, we generate the potential seeds not from the the reads directly, but from the high-scoring alignments between them. Because every high-scoring alignment between the reads corresponds to a path through the WSG, we take all k-long subpaths in the WSG and output the sequence of edge labels along the path. Recall that each edge of the WSG is labeled by a pair of letters (or gaps), one from each read. If one of the reads is gapped at the position, we output the letter from the other read. If neither read is gapped we output a letter from an arbitrary read. While the number of k-mers we generate per pair of reads can be large if the two reads are very different, in practice the reads are similar and the WSG is small and "narrow". In our simulations (described in the Results section) we saw an average of 27 k-mers per read-pair for reads of approximately 30 nucleotides.

Genome Scan Given the sets of k-mers generated from the WSGs of read-pairs, we build a lookup table, mapping from the k-mers to the reads. We used spaced seeds [1], where certain pre-determined positions are "squeezed" from the k-mers to increase sensitivity. We then scan the genome, searching for matches between k-mers present in the genome and the reads. If a particular read has as many or more than a specified number of k-mer matches within a given window of the genome, we execute a vectorized Smith-Waterman step, described in the next section.

Unlike some local alignment programs that build an index of the genome and then scan it with each query (read), we build an index of the reads and query this index with the genome. This approach has several advantages: first, it allows us to control memory usage, as our algorithm never needs memory proportional to the size of the genome, while the large set of short reads can be easily divided between many machines in a compute cluster. Secondly, our algorithm is able to rapidly isolate which reads have several k-mer matches within a small window by using a circular buffer to store all of the recent positions in the genome that matched the read. The genome scan algorithm is illustrated in Figure 2A.

2.4 Vectorized Smith-Waterman Implementation

While the genome scan described in the previous section significantly reduces the number of candidate regions, many false positives are encountered. To further reduce the number of potential mapping positions given to the WSG-based aligner,

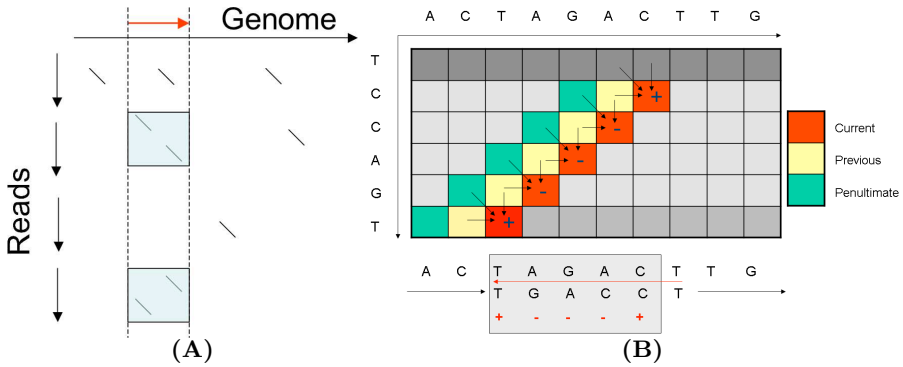

Fig. 2. A. Overview of the k-mer filtering stage within SHRiMP: A window is moved along the genome. If a particular read has a preset number of k-mers within the window the vectorized Smith-Waterman stage is run to align the read to the genome. **B.** Schematic of the vectorized-implementation of the Smith-Waterman algorithm. The red cells are the vector being computed, on the basis of the vectors computed in the last step (yellow) and the next-to-last (green). The match/mismatch vector for the diagonal is determined by comparing one sequence with the other one reversed (indicated by the red arrow below). To get the set of match/mismatch positions for the next diagonal the lower sequence needs to be shifted to the right.

we first align both reads to the candidate region using a vectorized implementation of the Smith-Waterman algorithm. Most contemporary mobile, desktop and server-class processors have special vector execution units, which perform multiple simultaneous data operations in a single instruction. For example, it is possible to add the eight individual 16-bit elements of two 128-bit vectors in one machine instruction. Over the past decade, several methods have been devised to significantly enhance the execution speed of Smith-Waterman-type algorithms by parallelizing the computation of several cells of the dynamic programming matrix. The simplest such implementation by Wozniak [13] computes the dynamic programming matrix using diagonals (See Figure 2B). Since each cell of the matrix can be computed once the cell immediately above, immediately to the left, and at the upper-left corner have been computed, one can compute each successive diagonal once the two prior diagonals have been completed. The problem can then be parallelized across the length of supported diagonals, often leading to a vectorization factor of 4 to 16. The only portion of such an approach that cannot be parallelized is the identification of match/mismatch scores for every cell of the matrix. These operations are done sequentially, necessitating 24 independent, expensive data loads for 8-cell vectors. The approach becomes increasingly problematic as vector sizes increase because memory loads cannot be vectorized; when the parallelism grows, so does the number of lookups.

Rognes and Seeberg [10] developed an alternative algorithm with the following innovations: first, they avoided the aforementioned loads by pre-calculating

a 'query profile' before computing the matrix. By pre-allocating vectors with the appropriate scores in memory, they needed only load a single score vector on each computation of a vector of cells. The up-front cost in pre-calculation greatly reduced the expense of computing scores in the crucial, tight inner-loop of the algorithm. The query profile, however, requires that scores be computed per-column, rather than for each diagonal. This creates data dependencies between several cells within the vector. However, the authors took advantage of the fact that often cells contain the value 0 when computing the Smith-Waterman recurrence, and any gap continuation would therefore be provably suboptimal. This enables the conditional skipping of a significant number of steps. Although highly successful for protein sequences, where only 1/20 elements on average have a positive score, it is significantly less beneficial for DNA sequences where usually at least 1/4 elements in the matrix match. Farrar [5] most recently introduced a superior method: by striping the query profile layout in memory, he significantly reduced the performance impact of correcting data dependencies.

We propose an alternate method, where the running time of the fully vectorized algorithm is independent of the number of matches and mismatches in the matrix, though it only supports fixed match/mismatch scores (rather than full scoring matrices). Since SHRiMP only applies a vectorized Smith-Waterman scan to short regions of confirmed k-mer hits, alternative approaches that benefit by skipping computation in areas of dissimilarity are unable to take significant advantage. Figure 2B demonstrates the essence of our algorithm: by storing one of the sequences backwards, we can align them in such a way that a small number of logical instructions obtain the positions of matches and mismatches for a given diagonal. We then construct a vector of match and mismatch scores for every cell of the diagonal without having to use expensive and un-vectorizable load instructions or pre-compute a "query profile". In our tests (see Table 1), our approach surpasses the performance of Wozniak's original algorithm [13] and performs on par with Farrar's method [5]. The advantages of our method over Farrar's approach are simplicity, independence of the running time and the scores used for matches/mismatches/gaps, and linear scalability for larger vector sizes. The disadvantages of our method are that it cannot support full scoring matrices (i.e. it is restricted to match/mismatch scores) and is slower for queries against large reference sequences where significant areas of dissimilarity are expected. However, the former is less important for DNA alignment and the latter does not apply to SHRiMP.

The vectorized Smith-Waterman approach described above is used to rapidly determine if both of the reads have a strong match to a given region of the genome. The locations of the top n hits for each read on the genome are stored in a heap data structure, which is updated after every invocation of the vectorized Smith-Waterman algorithm if the heap is not full, or if the attained score is greater than or equal to the lowest scoring top hit. Once the whole genome is processed, the highest scoring n matches are re-aligned using the full WSG alignment algorithm described in Section 2.2.

Processor type	Unvectorized	Wozniak	Farrar	SHRiMP
P4 Xeon	97	261	335	338
Core 2	105	285	533	537

Table 1. Performance (in millions of cells per second) of the various Smith-Waterman implementations, including a regular implementation, Wozniak's diagonal implementation with memory lookups, Farrar's striped algorithm and our diagonal approach without score lookups (SHRiMP). We used each method within SHRiMP to align 50 thousand reads to a reference genome with default parameters. The improvement of the Core 2 architecture for vectorized instructions lead to a significant speedup for our and Farrar's approaches, while the Wozniak algorithm's slight improvement is due to the slow memory lookups.

3 Results

In order to test the efficacy of our read mapping algorithm we designed a simulated dataset with properties similar to those expected from the first generation SMS sequencing technologies, such as Helicos [4]. We sampled 10,000 pairs of reads from human chromosome one (the total length of the chromosome is approximately 1/10th of the human genome). The first read was sampled to have a mean length of 30 bases, with standard deviation of 8.9, the second one had a mean length of 26 bases with standard deviation 8.6. The lengths of both reads ranged from 12 to 58 bases. We introduced errors into each of the the reads, with 7% deletion errors and 0.5% insertion and substitution errors uniformly at random. We also introduced Single Nucleotide Polymorphisms (SNPs) at a rate of 1%.

We used three approaches to map the read pairs back to the reference. Our main approach was the WSG-based alignment algorithm described in Section 2.2. For comparison we also implemented two alternatives: mapping the reads individually, using the version of SHRiMP for Illumina/Solexa, but with the same parameters as were used for WSG-based mapping in order to properly capture the high expected deletion rates, and a profile-based approach, which follows the overall framework of the WSG approach, but only considers a single top alignment (one path in the WSG) for further mapping to the reference genome. The methods are labeled WSG, $SEPARATE$, and $PROFILE$, respectively, in Table 2.

We evaluated the performance of these algorithms based on three criteria: the fraction of the reads that were not mapped at all (lower is better), the fraction of the reads that were not mapped uniquely (lower is better: while some reads have several equally good alignments due to repetitive DNA, others map with equal scores to non-repetitive DNA segments, and this should be minimized), and the percentage of the reads mapped uniquely and correctly (higher is better). When evaluating the two reads separately, we considered a hit unique if either of the two reads had a unique top hit, and we considered a unique top hit correct if either of the top hits was correct. We ran all algorithms with three seed weights:

8, 9, and 10, with each spaced seed having a single wildcard character (zero) in the middle of the seed (e.g. 1111101111 was the spaced seed of weight 9).

As one can see in Table 2, the approach of mapping the reads separately, while leading to very good sensitivity (almost all reads aligned), has poor specificity (only 70-78% of the reads have a unique top hit, with 61-64% both unique and correct). The WSG and $PROFILE$ approaches perform similarly, with the WSG approach slightly more sensitive and having a higher fraction of unique correct mappings for all seed sizes.

Type	SEPARATE			PROFILE			WSG		
seed weight	8	9	10	8	9	10	8	9	10
no hits %	**0.000**	**0.131**	**2.945**	1.741	4.905	10.52	1.609	4.314	10.15
multiple %	30.12	26.45	21.13	**10.20**	9.342	8.353	10.44	**9.127**	**8.258**
unique cor %	63.90	63.00	61.09	78.96	74.90	69.66	**79.17**	**75.84**	**70.85**
runtime	28m16	**8m48**	**4m22**	**27m17**	11m32	6m58	30m59	12m13	7m13

Table 2. Comparison of the WSG-based alignment with the two alternative approaches. The first two rows describe the percentage of reads not mapped anywhere on the genome or mapped in multiple places (with equal alignment scores). The third row is the percentage of the unique hits that are correct. The sub-columns under each method are the number of matching positions in the spaced seed (we allowed for a single non-matching character in the middle of the seed). The last row shows the running time in minutes and seconds on a 2.66 GHz Core2 processor. The best result in each cathegory is in boldface.

4 Discussion

In this paper we propose a novel read alignment algorithm for next-generation Single Molecule Sequencing (SMS) platforms. Our algorithm takes advantage of spaced k-mer seeds and effective filtering techniques to identify potential areas of similarity, a novel, vectorized implementation of the Smith-Waterman dynamic programming algorithm to confirm the similarity and a weighted sequence graph-based three way final alignment algorithm. Our approach is implemented as part of SHRiMP, the SHort Read Mapping Package, and is freely available at **http://compbio.cs.toronto.edu/shrimp**.

While the overall emphasis of the current paper was read mapping, we believe that several of our methods may have broader applications. For example, our implementation of the Smith-Waterman algorithm can be used in other sequence alignment methods, and unlike previous vectorized implementations the running time of our algorithm is independent of the scoring parameters. The weighted sequence graph model could also be useful for ab initio genome assembly of SMS data, where a variant of our method could be used for overlap graph construction.

References

1. Andrea Califano and Isidore Rigoutsos. Flash: A fast look-up algorithm for string homology. In *Proceedings of the 1st International Conference on Intelligent Systems for Molecular Biology*, pages 56–64. AAAI Press, 1993.
2. R. Chenna, H. Sugawara, T. Koike, R. Lopez, T. J. Gibson, D. G. Higgins, and J. D. Thompson. Multiple sequence alignment with the clustal series of programs. *Nucleic Acids Res*, 31(13):3497–3500, July 2003.
3. J.H. Jettand et al. High-speed DNA sequencing: an approach based upon fluorescence detection of single molecules. *Journal of biomolecular structure and dynamics*, 7(2):301–309, 1989.
4. Timothy D. Harris et al. Single-molecule DNA sequencing of a viral genome. *Science*, 320(5872):106–109, April 2008.
5. M. Farrar. Striped Smith-Waterman speeds database searches six times over other SIMD implementations. *Bioinformatics*, 23(2):156–161, 2007.
6. J. Hein. A new method that simultaneously aligns and reconstructs ancestral sequences for any number of homologous sequences, when the phylogeny is given. *Molecular Biology and Evolution*, 6(6):649–668, 1989.
7. D. Naor and D.L. Brutlag. On near-optimal alignments of biological sequences. *Journal of Computational Biology*, 1(4):349–266, 1994.
8. S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequences of two proteins. *JMB*, 48:443–453, 1970.
9. Kim R. Rasmussen, Jens Stoye, and Eugene W. Myers. Efficient q-gram filters for finding all epsilon-matches over a given length. *Journal of Computational Biology*, 13(2):296–308, 2006.
10. T. Rognes and E. Seeberg. Six-fold speed-up of smith-waterman sequence database searches using parallel processing on common microprocessors. *Bioinformatics*, 16(8):699–706, 2000.
11. Benno Schwikowski and Martin Vingron. Weighted sequence graphs: boosting iterated dynamic programming using locally suboptimal solutions. *Discrete Appl. Math.*, 127(1):95–117, 2003.
12. T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147:195–197, 1981.
13. A. Wozniak. Using video-oriented instructions to speed up sequence comparison. *Computer Applications in the Biosciences*, 13(2):145–150, 1997.