Mathematics of the Design of a Parallel Mapping Assembly Algorithm

Combining Smith-Waterman and Hirschberg's LCS Methods

Jaime Seguel
Electrical and Computer Engineering Department, University of Puerto Rico at Mayaguez, PR 00681, U.S.A.

Keywords: Sequence Alignment, Parallel Computation, Recursion, Mapping Assembly.

Abstract: This paper focuses on mathematical definitions and results that prove the correctness of a parallel algorithm for mapping assembly. The mathematical concepts and facts discussed here establish the reach and limitations of a combination of Smith-Waterman local alignment method and Hirschberg's divide-and-conquer longest common subsequence determination method. The parallel algorithm, whose correctness is proved, is a general method that works best for solving the problem of the local alignment of a short and a very large sequence, such as an entire genome. The method is thus, suitable for mapping assembly, where millions of short sequence segments, the so-called reads, are aligned with a whole genome.

1 INTRODUCTION

Sequencing is the process of determining the precise order of the characters that compose a DNA, mRNA or a protein string. Sanger sequencing (Sanger, Coulson, 1975), a pioneer sequencing method, remained the method of choice up to the advent of *next generation sequencing* (NGS) (Weijia Soon et. al., 2013). NGS are parallel processes that produce millions of short sequence segments, the so-called *reads,* at once. The shear amount and short length of the reads renders Sanger's assembling algorithms time and space inefficient.

Assembly algorithms are classifiable in two main groups: *de novo* assembly, and *mapping assembly* methods. De novo assembly reconstructs the sequence directly from the reads, through combinatorial graph algorithms (Li et. al., 2010). Mapping assembly, instead, aligns the reads against a reference genome. Mapping assembly is normally faster than de novo assembly but both methods incur inaccuracies and ambiguities.

The quality of the sequence returned by a mapping assembly algorithm depends on the accuracy of the underlying pairwise alignment method. An alignment of two strings, S_1 and S_2 over an alphabet Σ is a $2 \times q$ array, $q \geq \max \{|S_1|, |S_2|\}$; with the characters of S_1 in the first row and the characters of S_2 in the second, both placed in the order that they appear in the original sequences. There are two kinds of alignment, namely gapped and unpgapped alignments. In an ungapped alignment no symbols or blank spaces are inserted between characters. Blanks may be placed before or after the sequence character provided that no column of the alignment consists solely of blanks. An optimal ungapped alignment places the maximum number of characters that are similar in the same column. A gapped alignment, or simply *alignment*, fills the alignment array with the characters of an extended alphabet $\Sigma \cup \{-\}$. Here "–", the *gap character,* is not an element in Σ. No blank spaces are allowed but one or more consecutive gap characters may separate the characters of the sequences. No gap character is to be aligned with a gap character, either.

Alignments are built on the basis of scoring frameworks that consists of a *substitution matrix* and a *gap insertion penalty function*. The substitution matrix, denoted $M = [M(a, b)]$, assigns a score to the substitution of each pair a, b of symbols in Σ. The penalty for a *gap insertion*, in turn, is assigned with a function of the form

$$\gamma(g) = -d - (g - 1)e; \quad (1)$$

where e and d are constants, and g is the *gap length*, this is, the number of gaps symbols inserted between two consecutive sequence characters. The *score of an alignment* is the sum of the substitution scores of each sequence's character alignment, minus the sum of all penalties for gap inserted.

Seguel J..
Mathematics of the Design of a Parallel Mapping Assembly Algorithm - Combining Smith-Waterman and Hirschberg's LCS Methods.
DOI: 10.5220/0004883802210226
In *Proceedings of the International Conference on Bioinformatics Models, Methods and Algorithms* (BIOINFORMATICS-2014), pages 221-226
ISBN: 978-989-758-012-3

The *local alignment problem* (LAP) is the search for a pair of subsequences, one from S_1 and the other from S_2, whose alignment achieves maximum score. LAP is a well-posed optimization problem and the Smith-Waterman (SW) algorithm (Smith, Waterman, 1981) solves LAP exactly, in $O(|S_1||S_2|)$ time and space. However, because of the shear length of genome sequences and the ever-increasing number of reference sequences available for comparisons, the heuristic method called *Basic Local Alignment Sequence Tool* (BLAST) (Altschul, 1997), is preferred by practitioners. BLAST solves LAP approximately, by finding high scoring pairs (HSP) instead of alignments. A HSP is the extension of a *word hit*, which is the ungapped alignment of the sequence with a short word whose score is greater than or equal to a user defined threshold. Word hits are extended with a local sequence alignment method, such as Smith-Waterman, up until their scores drop below another user-defined threshold. Although BLAST returns the results much faster than Smith-Waterman, it is still deemed too slow for most post-genomic era processing demands. Mapping assembly is an important post-genomic instance of this demand. Post BLAST tools are designed for rapidly aligning a sequence to an entire genome, on a desktop computer. One such tool is MUMmer (Delcher et. al., 2002). MUMmer's speed rests on efficient suffix tree representations of the sequences. Suffix trees identify perfect matches, which are more restrictive than ungapped alignments, in linear time and space. MUMmer aligns short reads to a genome using a specialized routine called NUCmer. As in BLAST, NUCmer approximate solutions are extensions of exact matches produced with gapped or ungapped alignment methods. In general both BLAST and NUCmer, explore a subspace of the alignment space, and therefore, often return a suboptimal alignment.

The tradeoff between speed and exactness is highly sensitive when it comes to mapping assembly. Different approximate alignments often result in completely unrelated mapping (Li, Homer, 2010). The advantages of an exact algorithmic solution became apparent soon after the introduction of BLAST. Comparative studies (Shpaer et. al., 1996) report a significantly lower number of false positives and negatives in SW responses. Also, several experiments have reportedly shown a significant higher risk for BLAST to miss a sequence alignment that is detectable by Smith-Waterman. Approximate local alignment solutions (Phillippy et. al., 2008) create a need for long and exhaustive post-assembly processing (Rahman, 2013). This motivates the exploration of accurate mapping assembly algorithms that are time and space efficient, as well.

This article examines some mathematical principles behind the design of a parallel method based on Smith-Waterman and Hirschberg's longest common subsequence algorithm (Hirschberg, 1975). The idea of combining a sequence alignment method and Hirschberg's algorithm is not new. Combinations of Hirschberg and Needleman-Wunsch, a global alignment method that preceded Smith-Waterman, have been reported without in-depth discussions of the mathematics of their design. The algorithms that resulted from this combination are proved to save memory space to the cost of a slight increase in computation time. The author is not aware of specific reports on combinations of Hirschberg and Smith-Waterman.

The parallel algorithm, whose principles are discussed here, uses Hirschberg's division phase to partition the genome into string segments that are distributed over a set of processors. Each processor has a copy of the read strings. The optimal alignments of a read and the genome segments are computed in parallel, with Smith-Waterman. The method compares each of the processor's results and returns the alignment with the maximum score. Although this idea is similar to the one inspired by a combination of Needleman-Wunsch and Hirschberg, the actual design of the Smith-Waterman/Hirschberg algorithm has at least two important differences. The first difference is that, being a global alignment; Needleman-Wunsch has to be recomputed at each step of Hirschberg's division phase. Otherwise, the global alignment may not be retrievable from the segments. As shown in Corollary 2, the division phase of the Smith-Waterman/Hirschberg combination does not require Smith-Waterman computations, at least up to a certain depth in the division tree. The second difference is in the conquer phase. Unlike the Needleman-Wunsch/Hirschberg's conquer phase, which is just the concatenation of the alignments that solve each sub problem; Smith-Waterman/Hirschberg's conquer phase does require some extra processing. This is due to the fact that the best local alignment might correspond to the alignment of a read over two contiguous genome segments. Most of the theory developed in this article concerns the reconstruction of the local alignment of a read with a genome from its alignments with a pair of contiguous genome segments.

The mathematical concepts and facts discussed here come from observations made in the design of

PALMA (Parallel Algorithm for Local Mapping Assembly) by the author and collaborators. PALMA is currently under implementation.

The rest of this paper is organized as follows: Section 2 revisits Smith-Waterman and Hirschberg's longest common subsequence algorithm. Section 3 discusses the mathematical principles that allow a perfect solution of the border problem. Section 4 provides some conclusions and future work.

2 SMITH-WATERMAN AND HIRSCHBERG'S LCS

This section is a brief review of Smith-Waterman and the division phase of Hirschberg's Longest Common Subsequence algorithms.

2.1 Smith-Waterman

Smith-Waterman solves LAP in two main steps. First, it computes recursively the scores of subsequence alignments with a positive score. The results are stored in a $|S_1|\times|S_2|$ dynamic programming matrix D = [D(k, j)]. Then, tracing back D from the maximum entry up until the first zero retrieves the alignment. Trace back may be simplified with the help of an auxiliary matrix of pointers P = [P(k, j)], where each P(k, j) points to the cell whose value produced the maximum D(k, j) through the local alignment recursive relation. The next pseudo code implements Smith-Waterman local alignment recursion as a pair of nested loops.

```
SW (S1, S2, M, e, d)
//Initialization
For k←0 to |S1|; D(k,0)←0
For j←0 to |S2|; D(0,j)←0
g1←1 and g2←1
//Dynamic (D) and Backtrack (P) matrix
//computations
For k ← 1 to |S1|
  For j ← 1 to |S2|
    D(k,j)←max{0,
               D(k-1,j-1)+M(S1[k],S2[j])
               D(k-1,j)-d-g1×e,
               D(k,j-1)-d-g2×e}
    If D(k,j)=0 Or
       D(k,j)=D(k-1,j-1)+ M(S1[k],S2[j])
          g1 ← 1, g2 ← 1
         And P(k, j) ← diagonal
    Else If D(k,j)=D(k-1,j)-d-g1×e
          g1 ← g1 + 1, g2 ← 1
          And P(k,j) ← left
    Else g1 ← 1, g2 ← g2 + 1
          And P(k,j) ← up
  End for
End for
Return D and P.
```

The alignment is reconstructed from a tuple of indices of D referred here a *path segment*. This tuple is produced with the following routine:

```
Backtrack (D(k,j), P)
If D(k,j) = 0 Return ((k,j))
Else
  π ← ( )
  While D(k,j) > 0
      If P(k,j) = "diagonal"
            k ← k – 1 and j ← j – 1
     Else If P(k, j) = left
           k ← k – 1
     Else j ← j – 1
     π ← insert(k,j) as a new leftmost
     element in tuple π
  End While
End If-Else
Return π
```

In general, the trace back computation can be started at any entry of D. If D(k, j) is the maximum entry in D, Backtrack returns the optimal local alignment.

2.2 Division Phase of Hirschberg's Longest Common Subsequence Algorithm

Hirschberg's LCS algorithm is a divide-and-conquer method for finding the longest common subsequence (LCS) of two sequences. The principle behind the method is deceptively simple. In general, let S* be sequence S in reversed order. Then, the longest common subsequence of S_1 and S_2 equals the longest common subsequence of S_1* and S_2*. This fact allows splitting the search for the LCS in two independent searches of roughly half the size of the original. The first searches the LCS of the first half of S_1 and S_2 while the second, the LCS of the first half of S_1* and S_2*. The division phase is a recursive repetition of this string split and reversal operation. The conquer phase, in turn, composes the LCS segments found at the end of the division phase. A detailed discussion of this method is beyond the scope of this article. Here we concentrate on the algorithm's decomposition phase.

For a fixed but arbitrary pair of nonnegative integers p and q, p < q, let's denote S[p…q] the segment of S that starts in S[p] and ends in S[q]; and S[q…p] the reversal of S[p…q]. The next general decomposition method, which is inspired in Hirschberg's decomposition phase, is the core operation in the decomposition phase of our parallel algorithm.

```
Hirschberg Decomposition (S1, S2, h)
S ← S1; T ← S2
```

```
RHD(S, T, h)
If h < 0
     Return (S, T)
Else
     h ← h - 1;
     T1 ← T[1…ceil(|T|/2)]
     T2 ← T[|T|…ceil(|T|/2) - 1]
     RHD(S, T1, h)
     RHD(S*, T2, h)
```

In this context, h is positive integer that denotes the height of the decomposition tree.

The basic Hirschberg's principle does translate to alignment problems in the sense that an alignment and its reversal have the same score. However, the recursive splitting may incur loses of information that impede a perfect reconstruction.

3 MATHEMATICAL FRAMEWORK

In this section we state the reconstruction problem in mathematical terms and state and prove some results.

3.1 Basic Definitions

Backtrack returns an (r+1)-tuple $((k_0, j_0),\ldots,(k_r, j_r))$ of indices of D referred as *path segment* (PS). A PS is characterized by $D(k_0, j_0) = 0$ and $D(k_i, j_i) > 0$ for i = 1,…,r; if the tuple has more than one pair. The *left projection* of a PS is defined to be the sequence segment $S_1[k_1\ldots k_r]$ while its right projection, the sequence segment $S_2[j_1\ldots j_r]$. A PS $((k_0, j_0),\ldots,(k_r, j_r))$ is said to be a *longest path segment* (LPS) if and only if

i. $D(k_r, j_r) + M(S_1[k_r+1], S_2[j_r+1]) \leq 0$, and
ii. $D(k_r + 1, j_r) - g_1 \leq 0$, and
iii. $D(k_r, j_r + 1) - g_2 \leq 0$.

Let $I = \{k_1, \ldots, k_r\} \times \{j_1,\ldots,j_r\}\}$. A path segment $((k_0, j_0),\ldots,(k_r, j_r))$ is called *maximal score path segment* (MSPS) if

$$D(k_r, j_r) = \max \{D(k, j): (k, j) \text{ in } I\}. \quad (2)$$

In general, an MSPS is a sub path of an LPS. Therefore, the maximum value $D(k_r, j_r)$ in (2) does not necessarily correspond with the value of D in the last index of an LPS that contains an MSPS. Such maximal value is referred as *maximal local score* (MLS).

The basic idea behind the parallel local alignment method can be restated now as the use of Hirschberg Decomposition to partition and distribute the reference genome among a given number of processors, and the use of SW in each processor to compute in parallel the MSPS that corresponds to the highest MLS in each segment. As remarked above, a problem with this strategy is that the division of the genome may split some MSPS in two or more segments forcing thus a reconstruction process. Such reconstruction is the result of joining an MSPS segment with its complementary segment, which, because of Hirschberg's decomposition, is in reversed order.

The *reverse* of $\pi = ((k_0, j_0),\ldots,(k_r, j_r))$, a path segment for the alignment of S_1 and S_2, is defined as $\pi^* = ((|S_1|- k_r, |S_2| - j_r), \ldots , (|S_1| - k_0, |S_2| - j_0))$.

3.2 Theoretical Results

Given a pair of sequences S_1 and S_2, we denote by $D^* = [D^*(k, j)]$ the dynamic programming matrix returned by the application of SW to S_1^* and S_2^*.

The next Theorem is a fundamental result.

Theorem 1. Let S_1 and S_2 be sequences over the same alphabet. Let $D = [D(k, j)]$ and $D^* = [D^*(k, j)]$ be as defined above. Let $\pi = ((k_0, j_0),\ldots,(k_r, j_r))$ be an LPS. Then: For all $k_0 \leq k \leq k_r$ and $j_0 \leq j \leq j_r$;

a) $D(k, j) + D^*(|S_1| - k, |S_2| - j) \leq MLS$; and
b) $D(k, j) + D^*(|S_1| - k, |S_2| - j) = MLS$ if and only if (k, j) is in an MSPS.

Proof. By definition of the SW algorithm, D(k, j) is the highest score of the alignment of the prefixes $S_1[k_0\ldots k]$ and $S_2[j_0\ldots j]$ of the projections of π. Also by definition of SW and definition of sequence reversal, the value $D^*(|S_1| - k, |S_2| - j)$ is the score of the alignment of the suffixes $S_1[k+1\ldots k_r]$ and $S_2[j+1\ldots j_r]$ of the same projections, but in reversed order. Since the alignment of a pair of sequences has the same characters and gaps insertions than the alignment of the same pair but in reversed order; the latter score equals the score of the alignment of the suffixes in their original order. Therefore, $D(k, j) + D^*(|S_1| - k, |S_2| - j)$ is the score of an alignment of $S_1[k_0\ldots k_r]$ and $S_2[j_0\ldots j_r]$. Clearly, this score is always less than or equal to the maximal local score. This proves a).

As for the proof of claim b)**;** let B be the maximum score in the block D(k, j), (k, j) in $\{k_0,\ldots,k_r\}\times\{j_0,\ldots,j_r\}$. Let $\pi_1 = ((k_0, j_0),\ldots,(k_m, j_m))$ be an MSPS in this block. By definition of MSPS, if (k, j) is in π_1, then by the previous argument, $D(k, j) + D^*(|S_1| - k, |S_2| - j) = B$. Reciprocally, if (k, j) is not in π_1, $D(k, j) + D^*(|S_1| - k, |S_2| - j) < B$. This proves b).

In general, the reverse of a PS in D is not necessarily a PS in D*. However, this relation holds

for MSPS, as demonstrated in next theorem.

Theorem 2. Let π be an MSPS for the alignment of S_1 and S_2. Then, π^* is an MSPS for the alignment of S_1^* and S_2^*.

Proof. Let $\pi = ((k_0, j_0),\ldots,(k_r, j_r))$. Since π is an MSPS, it satisfies equation (2). By applying Theorem 1 b),

$$D(k, j) + D^*(|S_1| - k, |S_2| - j) = D(k_r, j_r); \quad (3)$$

for all (k, j) in π. In order to demonstrate that π^* is an MSPS for the alignment of S_1^* and S_2^* we need to show that:

i. $D^*(|S_1| - k_r, |S_2| - j_r) = 0$,
ii. $D^*(|S_1| - k, |S_2| - j) > 0$ for (k, j) in π, $(k, j) \neq (k_r, j_r)$, and
iii. $D^*(|S_1| - k_0, |S_2| - j_0) = MLS$.

By substituting (k_r, j_r) in equation (3) we get

$$D(k_r, j_r) + D^*(|S_1| - k_r, |S_2| - j_r) = D(k_r, j_r). \quad (4)$$

Therefore, $D^*(|S_1| - k_r, |S_2| - j_r) = 0$; and i. is proved. By substituting (k_0, j_0) in equation (4) we get $D(k_0, j_0) + D^*(|S_1| - k_0, |S_2| - j_0) = D(k_r, j_r)$. But since $D(k_0, j_0) = 0$, $D^*(|S_1| - k_0, |S_2| - j_0) = D(k_r, j_r)$ follows. Now, $D(k_r, j_r)$ is the maximum score for the alignment of $S_1[k_0\ldots k_r]$ and $S_2[j_0\ldots j_r]$. Since the score of a local alignment and its reversal are the same. $D(k_r, j_r)$ is also the maximum score for the alignment of $S_1[k_0\ldots k_r]^*$ and $S_2[j_0\ldots j_r]^*$. This proves iii. Finally, since for each (k, j) in π, $(k, j) \neq (k_r, j_r)$ and (k_0, j_0); $0 < D(k, j) < D(k_r, j_r)$, by equation (4) we conclude that ii. is also true.□

Theorem 1 provides a solution for the reconstruction of a split MSPS.

Let $\pi = ((k_0, j_0),\ldots,(k_r, j_r))$ be an MSPS for the alignment of S_1 and S_2. Assume that $S_2[j_0\ldots j_r]$ is split into $S_2[j_0\ldots j_m]$ and $S_2[j_m + 1 \ldots j_r]$ for some $j_0 < j_m < j_r$. By computing the alignment of S_1 and $S_2[j_0\ldots j_m]$ and that of S_1^* and $S_2[j_m + 1 \ldots j_r]^*$ we get the PS $\pi_1 = ((k_0, j_0),\ldots,(k_m, j_m))$ and $\pi_2 = ((|S_1| - k_r, |S_2| - j_r),\ldots,(|S_1| - k_0, |S_2| - j_0))$. By joining (π_1, π_2^*) we reconstruct the original MSPS. Unfortunately, recursive splitting may not allow a perfect path reconstruction as further divisions of $S_2[j_0\ldots j_m]$ or $S_2[j_m + 1 \ldots j_r]^*$ may not be MSPS. The previous considerations prove the next Corollary.

Corollary 1. If a PS in the alignment of $S_1[k_0\ldots k_r]$ and $S_2[j_0\ldots j_r]$ is restricted to $\pi_1 = ((k_0, j_0),\ldots,(k_m, j_m))$ for some j_m, $j_0 < j_m < j_r$, then it can be reconstructed by joining it with the reversed of the PS $\pi_2 = ((|S_1| - k_r, |S_2| - j_r),\ldots,(|S_1| - k_0, |S_2| - j_0))$ of the alignment of $S_1[k_0\ldots k_r]^*$ and $S_2[j_m + 1\ldots j_r]^*$ if π is an MSPS.

The next negative result proves the existence of cases in which perfect path reconstruction is not possible.

Lemma. Let $\pi = ((k_0, j_0),\ldots,(k_r, j_r))$ be a PS for the alignment of S_1 and S_2. Let j_m be an index, $0 < j_m < j_r$. Then, if $D(k_r, j_r) \leq D(k_{r-1}, j_{r-1})$, π cannot be reconstructed from $\pi_1 = ((k_0, j_0),\ldots,(k_m, j_m))$ and $\pi_2 = ((|S_1| - k_r, |S_2| - j_r),\ldots,(|S_1| - k_0, |S_2| - j_0))$ through the process described in the previous Corollary.

Proof. Since by hypothesis $D(k_r, j_r) \leq D(k_{r-1}, j_{r-1})$, the value of $M(S_1[k_r], S_2[j_r]) \leq 0$. Therefore, $D^*(|S_1| - k_r, |S_2| - j_r) = 0$; and thus, $(|S_1| - k_r, |S_2| - j_r)$ is not part of path π_2. As a consequence, (k_r, j_r) is not in π_2^*.

Corollary 2. Let $|S_1| < |S_2|$. Let M be the substitution matrix in a scoring framework and let $Q = \max\{M(a, b): a, b \text{ in } \Sigma\}$. Let $\gamma(g) = -d - (g - 1) \times e$ be the gap penalty mapping in the same scoring framework. Then, if

$$h \geq \log_2 (|S_1| + (Q|S_1| - d)/ e + 1); \quad (5)$$

the h-level Hirschberg Decomposition ensures perfect reconstruction.

Proof. Under the hypothesis of Corollary 2, the maximum score for an ungapped alignment of S_1 and S_2 is $Q|S_1|$. The maximal length g of gap is thus constrained by $Q|S_1| - d - (g - 1)e = 0$.

It follows that the maximal length of a gap is bounded by $g \leq (Q|S_1| - d)/e + 1$.

Thus, no PS is longer than $|S_1| + (Q|S_1| - d)/e + 1$. Therefore, the constraint imposed on parameter h ensures that no PS is split more than once. The claim follows from Corollary 1.□

4 CONCLUSIONS

The theoretical framework discussed in this article proves the correctness of a parallel algorithm for the computation of the local alignment of short and very long biological sequences. The parallelism improves with the difference in length of the sequences. The method partitions the large sequence using Hirschberg Decomposition up to the limit stated in equation (5), and computes in parallel the best local alignment of the short sequence with the segments of the large one, using the Smith-Waterman algorithm. If no border problems are encountered, the method is embarrassingly parallel as no inter processor communications are necessary. On the other hand, if Hirschberg Decomposition has split an MSPS, the reversed segment of the MSPS must be sent to the processor that holds its complement in natural order. Then, the receiving processor must reverse the received path segment, concatenate the segments together into a single LPS, and compute the underlying MSPS. All these processes are

executed in parallel and in linear time. Thus, the rate of growth of the execution time of the parallel algorithm is $O(|S_1||S_2|/W)$, where W is the number of workers or processor elements in the computing platform. According to equation (5), the limit of W is $W \leq |S_1| + (Q|S_1| - d)/ e + 1$ to ensure perfect reconstruction. Thus, in the limit of W, the parallel method should be close to $O(|S_2|)$ time and space.

Work is underway to use these ideas in the implementation of a parallel method for mapping assembly. This implementation is being developed in C language with MPI and OpenMP. In the mapping assembly program, special care is being taken to pipeline efficiently the millions of short reads into the parallel algorithm.

ACKNOWLEDGEMENTS

This work was supported in part by the NIH-MARC 5T36GM095335-02 award.

REFERENCES

Sanger R., Coulson A., 1975. A rapid method for determining sequences in DNA by primed synthesis with DNA polymerase. In *J. Ml. Biol*. 94(3), 441-448.

Weijia Soon W., Hariharan M., Snyder M., 2013. High-throughput sequencing for biology and medicine. In *Mol. Syst. Biology* 9:640 doi:10.1038/msb.2012.61.

Li R. et. al., 2010. De novo assembly of human genomes with massively parallel short read sequencing. In *Genome Research,* 20 (2), 265-272.

Smith T., Waterman M., 1981. Identification of common molecular subsequences. In *J. of Mol. Biol.* 147, 195-197.

Altschul S., Madden T., Shaffer A., Zhang J., Zhang Z., Miller W., Lipman D., 1997. Gapped BLAST and PSI-BLAST: A new generation of protein database search programs. In *Nuc. Ac. Res*. 25 (17), 3389-3402.

Delcher A., Phillippy A., Carlton J., Salzberg S., 2002. Fast algorithms for large-scale genome alignment comparison. In *Nuc. Ac. Res.,* 30(11), 2478-2483.

Li H., Homer N., 2010. A survey of sequence alignment algorithms for next-generation sequencing. In *Brief. Bioinform.,* 11(5) 473-483.

Shpaer E., Robinson M., Yee D., Candlin J., Mines R., Hunkapiller T., 1996. Sensitivity and selectivity in protein similarity searches: a comparison of smith-waterman in hardware to blast and fasta. In *Genomics,* 38(2), 179-191.

Rahman A., Pachter L., 2013. CGAL: computing genome assembly likelihoods. In *Gen. Biol*. 14:R8, doi:10.1186/gb-2013-14-1-r8.

Hirschberg D., 1975. A linear space algorithm for computing maximal common subsequences. In *Comm. ACM,* 18(6), 341-343, 1975.