A New, Space-Efficient Local Pairwise Alignment Methodology

Done Stojanov, Aleksandra Mileva and Sašo Koceski

Faculty of Computer Science, University “Goce Delčev”,
Tošo Arsov 14, Štip, Republic of Macedonia
e-mail: {done.stojanov, aleksandra.mileva, saso.koceski@ugd.edu.mk}

Abstract
Aligning large nucleotide sequences requires significant portion of the memory, what in some cases is found as a limiting factor. Local pairwise alignment methods require $O(m)$ space, most of them being dynamic programming-based. In this paper we present a new, space-efficient methodology and an algorithm for optimal un-gapped local pairwise alignment, using non dynamic programming-based approach.

A highest scoring un-gapped local alignment is found, identifying and combining exact nucleotide matching regions, shifting one of the sequences over the other. The number of shifts is score-dependent, what is a time complexity advantage. An implementation of the proposed methodology was tested on EMBL-EBI viroid genomes, always identifying an optimal un-gapped local alignment, requiring on average $m/2$ space.

Keywords: Un-gapped, optimal, pairwise alignment

Introduction

Several local pairwise sequence alignment techniques have been proposed since 1981, when Smith and Waterman[6] published their work, showing than an optimal local pairwise alignment can be found using dynamic programming. Because of the long running time, this approach is mainly applicable on short nucleotide sequences, always identifying exact highest scoring local pairwise alignment. In spite of the Smith-Waterman’s algorithm, which is one-alignment approach, Waterman and Eggert[7] in 1987 presented an algorithm, which identifies $k, k > 1$ best local pairwise alignments. Waterman-Eggert’s approach
was modified in 1991, by Huang and Milller\cite{2}, with a linear-space variant of Waterman-Eggert’s algorithm. Recent implementations as SWIFT\cite{5} and YASS\cite{4} output more than one local pairwise alignments, without restraining to find the optimal one. The first one identifies local alignments, limiting alignment’s length and error rate, while the second is based on conserved motifs identification. Previous implementations, do not try to find an optimal local pairwise alignment, since they perform alignments under constraints they pose.

In contrast to the previous approaches, BLAST\cite{1} and FASTA\cite{3} are heuristic, word searching methods, searching for high scoring local alignments between query sequence and database sequences, not always identifying an optimal local alignment.

Depending on whether gaps are allowed or not in an alignment, there are gapped and un-gapped alignments. In this paper, a new alignment methodology is presented, requiring on average $\frac{m}{2}$ space and $O(nm^2)$ time in order to find an optimal un-gapped local pairwise alignment, based on linear score-computing metrics.

1 Methodology

In this paper we propose a new, sequence shifting based methodology, for optimal un-gapped local alignment, revealing un-gapped subsequences being mostly conserved, applicable on two nucleotide sequences $a = \{a_i\}_{i=0}^{n-1}$ and $b = \{b_j\}_{j=0}^{m-1}$. According to the methodology we propose, a highest scoring common un-gapped region can be found shifting sequence $b$ at most $n + m - 1$ times over the sequence $a$, as Table 1 shows. Linear metrics, awarding $\mu$ for nucleotide match, while penalizing nucleotide mismatch with $-\delta$ is going to be used, $\mu > 0$, $\delta > 0$. At each shift, sequence $b$ section overlaps sequence $a$ section. Parallel nucleotides, within shift overlapping sections are compared, in order to find exact nucleotide matching region or regions. Exact nucleotide matching region, is a region within shift overlapping sections with one or more consecutive matching nucleotides. Three cases are possible at each shift:

1. No matching nucleotides have been found.
2. Exact nucleotide matching region has been found.
3. Two or more nucleotide matching regions have been found.

If exact nucleotide matching region has been found, then an exact matching un-gapped local alignment is found, within shift overlapping sections. If $\chi$, $\chi \geq 2$ exact nucleotide matching regions have been found, then $\chi$ exact matching un-gapped local alignments are possible, while un-gapped local alignments with at least one mismatch are obtained combining $\xi$ consecutive exact matching regions, for $\xi = 2, 3, \ldots, \chi - 1, \chi$. For each alignment, a score is computed. The
A new, space-efficient local pairwise alignment methodology

highest scoring un-gapped local alignment being found, shifting sequence \( b \) at most \( n + m - 1 \) times, is the optimal un-gapped local pairwise alignment.

<table>
<thead>
<tr>
<th>Shift</th>
<th>Overlapping Sections</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( a_0 \ a_1 \ ... \ a_{n-2} \ a_{n-1} ) ( b_0b_1 ... b_{m-2}b_{m-1} )</td>
</tr>
<tr>
<td>2</td>
<td>( a_0 \ a_1 \ ... \ a_{n-2} \ a_{n-1} ) ( b_0b_1 ... b_{m-2}b_{m-1} )</td>
</tr>
<tr>
<td></td>
<td>( \ldots \ldots \ldots )</td>
</tr>
<tr>
<td>( n+m-2 )</td>
<td>( a_0a_1...a_{n-2}a_{n-1} ) ( b_0b_1 ... b_{m-2}b_{m-1} )</td>
</tr>
<tr>
<td>( n+m-1 )</td>
<td>( a_0a_1...a_{n-2}a_{n-1} ) ( b_0b_1 ... b_{m-2}b_{m-1} )</td>
</tr>
</tbody>
</table>

Table 1
Sequence \( b \) shifting order

Exact nucleotide matching region(s) \( \zeta \), being found within overlapping sections at shift \( \tau \), \( R_{r,\zeta} \), are represented with triples: \( R_{r,\zeta} := (pb_{r,\zeta}, pa_{r,\zeta}, l_{r,\zeta}) \), where \( pb_{r,\zeta} \) is exact nucleotide matching region \( \zeta \) starting position at sequence \( b \), \( pa_{r,\zeta} \) is region \( \zeta \) starting position at sequence \( a \), while \( l_{r,\zeta} \) is region \( \zeta \) length(the number of matching nucleotides), \( l_{r,\zeta} \geq 1 \). If within shift \( \tau \) overlapping sections, \( \chi \) exact nucleotide matching regions have been found, un-gapped local alignments being found are shown in Table 2.

Table 2
All alignments being found, if \( \chi \) exact nucleotide matching regions are found

Exact matching alignments equal exact nucleotide matching regions, while un-gapped local alignments with at least one mismatch, are obtained combing consecutive \( \xi = 2, 3, \ldots, \chi - 1 \), \( \chi \) exact nucleotide matching regions. Figure 1 demonstrates all un-gapped local alignments identification, if five exact nucleotide matching regions have been found. Table 3 shows alignment’s score
computation.

**Figure 1.** All un-gapped local alignments being identified, if five exact nucleotide matching regions have been found

**Table 3**

Alignment’s score computation

<table>
<thead>
<tr>
<th>Alignment</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_{t,n} = (p_{b_t}, p_{a_t})$</td>
<td>$\mu R_{t,n}$</td>
</tr>
<tr>
<td>$R_{t,n+1} R_{t,n+2} \cdots R_{t,n+k} R_{n+k}$</td>
<td>$\sum_{i=1}^{k} l_{t,n+i} - \delta \sum_{j=2}^{k} \left( p_{b_{t,n+j}} - (p_{b_{t,n+j-1}} + l_{t,n+j-1}) \right)$</td>
</tr>
</tbody>
</table>

**Proposition 1:** No shift $k, k \geq n+m-[f_{\text{max}} / \mu]$ alignment has to be examined, since no alignment, being identified within shift $k$ overlapping sections is higher scoring than $f_{\text{max}}$, where $f_{\text{max}}$ is an optimal alignment’s score.

**Proof:** At shift $k, k \geq n+m-[f_{\text{max}} / \mu]$, at most $n+m-k$ matching nucleotides could be found. Since $\mu > 0$ and $n+m-k \leq [f_{\text{max}} / \mu]$, then $\mu f_{\text{max}} / \mu \geq \mu(n+m-k)$.

Also is true that $\mu \left( f_{\text{max}} / \mu \right) \geq \mu [f_{\text{max}} / \mu]$, where $f_{\text{max}} = \mu [f_{\text{max}} / \mu] + r$, $f_{\text{max}} = r (\mod \mu)$.

Combining the previous two inequalities, an inequality $\mu \left( f_{\text{max}} / \mu \right) \geq \mu [f_{\text{max}} / \mu] \geq \mu(n+m-k)$ is obtained, from where we get that $f_{\text{max}} \geq \mu(n+m-k) = f$, where $f$ is a maximum score of an un-gapped local alignment, that could have been found at shift $k, k \geq n+m-[f_{\text{max}} / \mu]$.

Proposition 1 implies that not all sequence $b$ shifts are required, in order to find the optimal un-gapped local pairwise alignment. Ones an un-gapped highest scoring local alignment, with scoring $f_{\text{max}}$, has been found, no un-gapped local alignment found at shift $k, k \geq n+m-[f_{\text{max}} / \mu]$ is higher scoring than $f_{\text{max}}$ and therefore no time has to be wasted examining them.
2 Example

Methodology being presented will be demonstrated on a concrete example, taking nucleotide segments \(a = \text{ACTTGATT}T\) and \(b = \text{TTGTCT}\) as test sequences. Metrics, awarding +2 for nucleotide match and penalizing nucleotide mismatch with -1, is going to be used. An optimal (highest scoring) un-gapped local alignment has to be found.

Table 4
Methodology demonstration, step by step

<table>
<thead>
<tr>
<th>Shift</th>
<th>Shift overlapping sections</th>
<th>Exact nucleotide matching regions</th>
<th>Un-gapped local alignment(s) being found</th>
<th>Alignment’s score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ACTTGATT TTGTCT</td>
<td>/</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>2</td>
<td>ACTTGATT TTGTCT</td>
<td>/</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>3</td>
<td>AC^{}TTGAT^{}TTGTCT^{}</td>
<td>R(_{3,1}): (5,1,2)</td>
<td>R(_{3,1}): (5,1,2)</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>AC^{}TTGAT^{}TTGTCT^{}</td>
<td>R(_{4,1}): (6,3,1)</td>
<td>R(_{4,1}): (6,3,1)</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>AC^{}TTGAT^{}TTGTCT^{}</td>
<td>R(_{5,1}): (4,2,1)</td>
<td>R(_{5,1}): (4,2,1)</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>AC^{}TTGAT^{}TTGTCT^{}</td>
<td>R(_{6,1}): (3,2,2)</td>
<td>R(_{6,1}): (3,2,2)</td>
<td>4</td>
</tr>
<tr>
<td>7</td>
<td>AC^{}TTGAT^{}TTGTCT^{}</td>
<td>R(_{7,1}): (3,3,1)</td>
<td>R(_{7,1}): (3,3,1)</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>R(_{7,2}): (6,6,1)</td>
<td>R(_{7,2}): (6,6,1)</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>R(_{7,3}): (3,3,1,6,6,1)</td>
<td>R(_{7,3}): (3,3,1,6,6,1)</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>AC^{}TTGAT^{}TTGTCT^{}</td>
<td>R(_{8,1}): (1,2,1)</td>
<td>R(_{8,1}): (1,2,1)</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>R(_{8,2}): (6,7,1)</td>
<td>R(_{8,2}): (6,7,1)</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>R(_{8,3}): (1,2,1,6,7,1)</td>
<td>R(_{8,3}): (1,2,1,6,7,1)</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>AC^{}TTGAT^{}TTGTCT^{}</td>
<td>R(_{9,1}): (0,2,3)</td>
<td>R(_{9,1}): (0,2,3)</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>R(_{9,2}): (4,6,1)</td>
<td>R(_{9,2}): (4,6,1)</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>R(_{9,3}): (6,8,1)</td>
<td>R(_{9,3}): (6,8,1)</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>R(_{9,4}): (0,2,3,4,6,1)</td>
<td>R(_{9,4}): (0,2,3,4,6,1)</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>R(_{9,5}): (4,6,1,6,8,1)</td>
<td>R(_{9,5}): (4,6,1,6,8,1)</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>R(_{9,6}): (0,2,3,4,6,1,6,8,1)</td>
<td>R(_{9,6}): (0,2,3,4,6,1,6,8,1)</td>
<td>8</td>
</tr>
<tr>
<td>10</td>
<td>AC^{}TTGAT^{}TTGTCT^{}</td>
<td>R(_{10,1}): (0,3,1)</td>
<td>R(_{10,1}): (0,3,1)</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>R(_{10,2}): (3,6,2)</td>
<td>R(_{10,2}): (3,6,2)</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>R(_{10,3}): (0,3,1,3,6,2)</td>
<td>R(_{10,3}): (0,3,1,3,6,2)</td>
<td>4</td>
</tr>
<tr>
<td>11</td>
<td>AC^{}TTGAT^{}TTGTCT^{}</td>
<td>R(_{11,1}): (3,7,2)</td>
<td>R(_{11,1}): (3,7,2)</td>
<td>4</td>
</tr>
</tbody>
</table>

As Table 4 shows, an optimal (highest scoring) un-gapped local alignment is found at shift 9, combining exact nucleotide matching regions: \(R\(_{9,1}\), R\(_{9,2}\), and R\(_{9,3}\). According Proposition 1, shift 12, shift 13, shift 14 and shift 15 don’t have to be examined; since the highest score of an un-gapped local alignment could have been identified within overlapping sections at those shifts can’t exceed 8.
3 Implementation

Methodology being presented is implemented in C++. Implementing this methodology requires two data vectors, one: \(<currentAlignment>\), holding alignments identifying variable integer data, other: \(<optimalAlignment>\), holding integer data for an optimal alignment being found until then. If within shift \(\tau\) overlapping sections, an exact nucleotide matching region is found, its score is calculated and compared with highest score for an alignment being found until then. If its score is higher, then a new optimal un-gapped local alignment has just been found and therefore \(<optimalAlignment>\) and \(f_{\text{max}}\) are updated. If \(\chi, \chi \geq 2\) exact nucleotide matching regions are found, then \(\chi\) exact matching alignments and \(\chi(\chi-1)/2\) alignments with at least one mismatch are possible, which are obtained calling: \(\text{CombineWithinShift}_{\tau}, \text{OverlappingSections}()\) method. The previous method takes as an argument the number of consecutive exact nucleotide matching regions being combined, \(\xi\). Calling this method in order to combine all \(\xi\) consecutive exact nucleotide matching regions, for \(\xi = 2, 3, ..., \chi - 1, \chi\), all possible un-gapped local alignments with at least one mismatch are obtained. If there is a higher scoring un-gapped local alignment, then \(<optimalAlignment>\) and \(f_{\text{max}}\) are updated, with the newly found values. Here is given implementation’s pseudo code.

\[
\text{OptimalUn\text{-}gappedLocalAlignment}(\text{inputs: } a, b, \text{ output: } <optimalAlignment>){}

\begin{align*}
&f_{\text{max}} = 0 \\
&\tau = 0 \\
\text{while} (f_{\text{max}} < a \text{.length()} + b \text{.length()} - [f_{\text{max}}/\mu]) \{
&\text{shift}(b) \\
&\tau++ \\
&\chi = \text{numberOfExactMatchingRegionsWithinShift}_{\tau}, \text{OverlappingSections}(a, b) \\
&\text{if}(\chi == 1)\{ \\
&<currentAlignment>.push\_back(R_{\tau}) \\
&\text{if}(f(<currentAlignment>) > f_{\text{max}})\{ \\
&<optimalAlignment>.clear() \\
&<optimalAlignment>.push\_back(<currentAlignment>) \\
&f_{\text{max}} = f(<currentAlignment>) \\
&\}
&\}
&<currentAlignment>.clear() \\
&\text{else if}(\chi > 1)\{ \\
&\text{for}(\text{int } j = 1; j <= \chi; j++)\{ \\
&<currentAlignment>.push\_back(R_{\tau}) \\
&\text{if}(f(<currentAlignment>) > f_{\text{max}})\{
\end{align*}

}
<optimalAlignment>.clear()
<optimalAlignment>.push_back(<currentAlignment>)
f_{max}=f(<currentAlignment>)
}
<currentAlignment>.clear()
}
NonExactMatchingAlignments=\chi - 1
CombineExactMatchingRegions=2
while(NonExactMatchingAlignments>=1){
temp=NonExactMatchingAlignments
while(temp>=1){
<currentAlignment>.push_back(CombineWithinShift_{\tau}
OverlappingSections(CombineExactMatchingRegions))
If(f(<currentAlignment>)>f_{max}){
<optimalAlignment>.clear()
<optimalAlignment>.push_back(<currentAlignment>)
f_{max}=f(<currentAlignment>)
}
<currentAlignment>.clear()
temp--
}
CombineExactMatchingRegions++
NonExactMatchingAlignments--
}
}
}
}

4 Results

Algorithm’s implementation in C++ was tested on Acer Aspire 5570Z notebook, with Genuine Intel T2080 at 1.73 GHz and 1024 MB RAM. Complete RNA sequences of four viroids were aligned. Nucleotide sequences were taken from EMBL-EBI database. Obtained results are given in Table 5. In both of the cases, the optimal un-gapped local alignment is found, using metrics, which awards +2 for nucleotide match, while penalizing nucleotide mismatch with -1.
Table 5
Results of performed alignments

<table>
<thead>
<tr>
<th>Organism</th>
<th>Sequence length</th>
<th>Organism</th>
<th>Sequence length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Citrus dwarfing viroid</td>
<td>291</td>
<td>Citrus viroid III isolate</td>
<td>294</td>
</tr>
<tr>
<td>Coconut cadang-cadang viroid isolate GOS Palm</td>
<td>246</td>
<td>Coconut cadang-cadang viroid isolate oil palm variant</td>
<td>270</td>
</tr>
</tbody>
</table>

5 Discussion

Different length Classical swine fever virus strain 94.4/IL/94/TWN and Classical swine fever virus 39 subsequences were aligned in order to measure algorithm’s running time and space complexity. Obtained results (Table 6 and 7), clearly show that on average m/2 memory units are required during the algorithm’s execution, which takes O(nm²) time, assuming that m is a length of the smaller nucleotide sequence.

Table 6
Measuring algorithm’s time performance

<table>
<thead>
<tr>
<th>Nucleotide subsequence 1</th>
<th>Length</th>
<th>Nucleotide subsequence 2</th>
<th>Length</th>
<th>Alignment time(seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classical swine fever virus strain 94.4/IL/94/TWN[1-300]</td>
<td>300</td>
<td>Classical swine fever virus 39[1-200]</td>
<td>200</td>
<td>1</td>
</tr>
<tr>
<td>Classical swine fever virus strain 94.4/IL/94/TWN[1-500]</td>
<td>500</td>
<td>Classical swine fever virus 39[1-400]</td>
<td>400</td>
<td>8</td>
</tr>
<tr>
<td>Classical swine fever virus strain 94.4/IL/94/TWN[1-600]</td>
<td>600</td>
<td>Classical swine fever virus 39[1-500]</td>
<td>500</td>
<td>17</td>
</tr>
</tbody>
</table>

Table 7
Measuring algorithm’s space complexity

<table>
<thead>
<tr>
<th>Nucleotide subsequence 1</th>
<th>Length</th>
<th>Nucleotide subsequence 2</th>
<th>Length</th>
<th>Average total space required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classical swine fever virus strain 94.4/IL/94/TWN[1-300]</td>
<td>300</td>
<td>Classical swine fever virus 39[1-200]</td>
<td>200</td>
<td>0,48×200</td>
</tr>
</tbody>
</table>
A new, space-efficient local pairwise alignment methodology

<table>
<thead>
<tr>
<th>Classical swine fever virus strain</th>
<th>Classical swine fever virus</th>
<th>350</th>
<th>0.43×350</th>
</tr>
</thead>
<tbody>
<tr>
<td>94.4/IL/94/TWN[1-400]</td>
<td>400</td>
<td>350</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Classical swine fever virus strain</th>
<th>Classical swine fever virus</th>
<th>400</th>
<th>0.43×400</th>
</tr>
</thead>
<tbody>
<tr>
<td>94.4/IL/94/TWN[1-500]</td>
<td>500</td>
<td>400</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Classical swine fever virus strain</th>
<th>Classical swine fever virus</th>
<th>500</th>
<th>0.45×500</th>
</tr>
</thead>
<tbody>
<tr>
<td>94.4/IL/94/TWN[1-600]</td>
<td>600</td>
<td>500</td>
<td></td>
</tr>
</tbody>
</table>

References


Received: December, 2011