Advanced Studies in Biology, Vol. 4, 2012, no. 2, 85 – 93

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 Millller[2], with a linear-space variant of Waterman-Eggert's algorithm. Recent implementations as SWIFT[5] and YASS[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[1] and FASTA[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 $m/2$ 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 $b = \{b\}_{j=0}^{m-1}$. 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 μ 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 χ exact matching un-gapped local alignments are possible, while un-gapped local alignments with at least one mismatch are obtained combining ξ consecutive exact matching regions, for $\xi = 2,3,...,\chi-1,\chi$. For each alignment, a score is computed. The

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 1
Sequence b shifting order

Shift	Overlapping Sections
1	$a_0 a_1 \ldots a_{n-2} a_{n-1}$ $b_0 b_1 \ldots b_{m-2} b_{m-1}$
2	$a_0 a_1 \ldots a_{n-2} a_{n-1}$ $b_0 b_1 \ldots b_{m-2} b_{m-1}$
	.....
$n+m-2$	$a_0 a_1 \ldots a_{n-2} a_{n-1}$ $b_0 b_1 \ldots b_{m-2} b_{m-1}$
$n+m-1$	$a_0 a_1 \ldots a_{n-2} a_{n-1}$ $b_0 b_1 \ldots b_{m-2} b_{m-1}$

Exact nucleotide matching region(s) ζ, being found within overlapping sections at shift τ, $R_{\tau,\zeta}$, are represented with triples: $R_{\tau,\zeta}:(pb_{\tau,\zeta}, pa_{\tau,\zeta}, l_{\tau,\zeta})$, where $pb_{\tau,\zeta}$ is exact nucleotide matching region ζ starting position at sequence b, $pa_{\tau,\zeta}$ is region ζ starting position at sequence a, while $l_{\tau,\zeta}$ is region ζ length(the number of matching nucleotides), $l_{\tau,\zeta} \geq 1$. If within shift τ overlapping sections, χ 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 χ exact nucleotide matching regions are found

χ	Exact matching alignments	Un-gapped local alignments with at least one mismatch
$\chi=1$	$R_{\tau,1}$	/
$\chi>1$	$R_{\tau,1}, R_{\tau,2}, \ldots, R_{\tau,\chi-1}, R_{\tau,\chi}$	$R_{\tau,1}R_{\tau,2}, R_{\tau,2}R_{\tau,3}, \ldots, R_{\tau,\chi-2}R_{\tau,\chi-1}, R_{\tau,\chi-1}R_{\tau,\chi}$ $R_{\tau,1}R_{\tau,2}R_{\tau,3}, R_{\tau,2}R_{\tau,3}R_{\tau,4}, \ldots, R_{\tau,\chi-3}R_{\tau,\chi-2}R_{\tau,\chi-1}, R_{\tau,\chi-2}R_{\tau,\chi-1}R_{\tau,\chi}$ ..... $R_{\tau,1}R_{\tau,2}\ldots R_{\tau,\chi-2}R_{\tau,\chi-1}, R_{\tau,2}R_{\tau,3}\ldots R_{\tau,\chi-1}R_{\tau,\chi}$ $R_{\tau,1}R_{\tau,2}\ldots R_{\tau,\chi-1}R_{\tau,\chi}$

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.

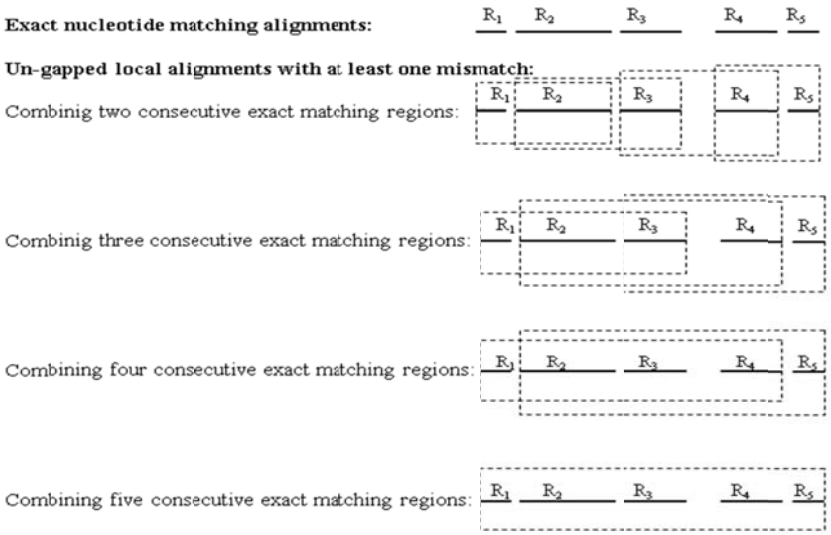

Figure 1. *All un-gapped local alignments being identified, if five exact nucleotide matching regions have been found*

Table 3

Alignment's score computation

Alignment	Score
$R_{\tau,\zeta} : (pb_{\tau,\zeta}, pa_{\tau,\zeta}, l_{\tau,\zeta})$	$\mu l_{\tau,\zeta}$
$R_{\tau,n+1}R_{\tau,n+2}..R_{\tau,n+k-1}R_{n+k}$	$\mu\sum_{i=1}^{k} l_{\tau,n+i} - \delta\left(\sum_{j=2}^{k}(pb_{\tau,n+j} - (pb_{\tau,n+j-1} + l_{\tau,n+j-1}))\right)$

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

Proof: At shift $k, k \geq n+m-[f_{\max}/\mu]$, at most $n+m-k$ matching nucleotides could be found. Since $\mu > 0$ and $n+m-k \leq [f_{\max}/\mu]$, then $\mu[f_{\max}/\mu] \geq \mu(n+m-k)$. Also is true that $\mu\left(\frac{f_{\max}}{\mu}\right) \geq \mu[f_{\max}/\mu]$, where $f_{\max} = \mu[f_{\max}/\mu] + r,\ f_{\max} \equiv r(\mathrm{mod}\ \mu)$. Combining the previous two inequalities, an inequality $\mu\left(\frac{f_{\max}}{\mu}\right) \geq \mu[f_{\max}/\mu] \geq \mu(n+m-k)$ is obtained, from where we get that $f_{\max} \geq \mu(n+m-k) = f$, where f is a maximum score of an un-gapped local alignment, that could have been found at shift $k, k \geq n+m-[f_{\max}/\mu]$.

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 $f_{\max}$, has been found, no un-gapped local alignment found at shift $k, k \geq n+m-[f_{\max}/\mu]$ is higher scoring than $f_{\max}$ 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 =ACTTGATTT and b =TTGTTCT 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

Shift	Shift overlapping sections	Exact nucleotide matching regions	Un-gapped local alignment(s) being found	Alignment's score
1	ACTTGATTT TTGTTCT	/	/	/
2	ACTTGATTT TTGTTCT	/	/	/
3	ACTTGATTT TTGTTCT	$R_{3,1}$: (5,1,2)	$R_{3,1}$: (5,1,2)	4
4	ACTTGATTT TTGTTCT	$R_{4,1}$: (6,3,1)	$R_{4,1}$: (6,3,1)	2
5	ACTTGATTT TTGTTCT	$R_{5,1}$: (4,2,1)	$R_{5,1}$: (4,2,1)	2
6	ACTTGATTT TTGTTCT	$R_{6,1}$: (3,2,2)	$R_{6,1}$: (3,2,2)	4
7	ACTTGATTT TTGTTCT	$R_{7,1}$: (3,3,1) $R_{7,2}$: (6,6,1)	$R_{7,1}$: (3,3,1) $R_{7,2}$: (6,6,1) $R_{7,1}R_{7,2}$: (3,3,1,6,6,1)	2 2 2
8	ACTTGATTT TTGTTCT	$R_{8,1}$: (1,2,1) $R_{8,2}$: (6,7,1)	$R_{8,1}$: (1,2,1) $R_{8,2}$: (6,7,1) $R_{8,1}R_{8,2}$: (1,2,1,6,7,1)	2 2 0
9	ACTTGATTT TTGTTCT	$R_{9,1}$: (0,2,3) $R_{9,2}$: (4,6,1) $R_{9,3}$: (6,8,1)	$R_{9,1}$: (0,2,3) $R_{9,2}$: (4,6,1) $R_{9,3}$: (6,8,1) $R_{9,1}R_{9,2}$: (0,2,3,4,6,1) $R_{9,2}R_{9,3}$: (4,6,1,6,8,1) $R_{9,1}R_{9,2}R_{9,3}$: (0,2,3,4,6,1,6,8,1)	6 2 2 7 3 **8**
10	ACTTGATTT TTGTTCT	$R_{10,1}$: (0,3,1) $R_{10,2}$: (3,6,2)	$R_{10,1}$: (0,3,1) $R_{10,2}$: (3,6,2) $R_{10,1}R_{10,2}$: (0,3,1,3,6,2)	2 4 4
11	ACTTGATTT TTGTTCT	$R_{11,1}$: (3,7,2)	$R_{11,1}$: (3,7,2)	4

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 τ 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_{max} are updated. If $\chi, \chi \geq 2$ exact nucleotide matching regions are found, then χ exact matching alignments and $\frac{\chi(\chi-1)}{2}$ alignments with at least one mismatch are possible, which are obtained calling: *CombineWithShift* $_{\tau}$ *OverlappingSections()* method. The previous method takes as an argument the number of consecutive exact nucleotide matching regions being combined, ξ. Calling this method in order to combine all ξ consecutive exact nucleotide matching regions, for $\xi = 2,3,\ldots,\chi-1,\chi$, 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_{max} are updated, with the newly found values. Here is given implementation's pseudo code.

```
OptimalUn-gappedLocalAlignment(inputs: a,b , output:<optimalAlignment>){
f_max=0
τ =0
while( f_max < a .length()+ b .length()-[f_max/ μ ]){
shift( b )
τ + +
χ =numberOfExactMatchingRegionsWithinShift τ OverlappingSections( a , b )
if( χ ==1){
<currentAlignment>.push_back( R_τ,1 )
if(f(<currentAlignment>)>f_max){
<optimalAlignment>.clear()
<optimalAlignment>.push_back(<currentAlignment>)
f_max=f(<currentAlignment>)
}
}
<currentAlignment>.clear()
else if( χ >1){
for(int j=1; j<= χ ; j++){
<currentAlignment>.push_back( R_τ, j )
If(f(<currentAlignment>)>f_max){
```

<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

Organism	Sequence length	Organism	Sequence length
Citrus dwarfing viroid	291	Citrus viroid III isolate 054-4uy	294
Alignment: 			
Coconut cadang-cadang viroid isolate GOS Palm	246	Coconut cadang-cadang viroid isolate oil palm variant 270	270
Alignment: 			

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 $O(nm^2)$ time, assuming that m is a length of the smaller nucleotide sequence.

Table 6
Measuring algorithm's time performance

Nucleotide subsequence 1	Length	Nucleotide subsequence 2	Length	Alignment time(seconds)
Classical swine fever virus strain 94.4/IL/94/TWN[1-200]	200	Classical swine fever virus 39[1-200]	200	1
Classical swine fever virus strain 94.4/IL/94/TWN[1-300]	300	Classical swine fever virus 39[1-200]	200	1
Classical swine fever virus strain 94.4/IL/94/TWN[1-400]	400	Classical swine fever virus 39[1-350]	350	4
Classical swine fever virus strain 94.4/IL/94/TWN[1-500]	500	Classical swine fever virus 39[1-400]	400	8
Classical swine fever virus strain 94.4/IL/94/TWN[1-600]	600	Classical swine fever virus 39[1-500]	500	17

Table7
Measuring algorithm's space complexity

Nucleotide subsequence 1	Length	Nucleotide subsequence 2	Length	Average total space required
Classical swine fever virus strain 94.4/IL/94/TWN[1-200]	200	Classical swine fever virus 39[1-200]	200	0,41×200
Classical swine fever virus strain 94.4/IL/94/TWN[1-300]	300	Classical swine fever virus 39[1-200]	200	0,48×200

Classical swine fever virus strain 94.4/IL/94/TWN[1-400]	400	Classical swine fever virus 39[1-350]	350	0,43×350
Classical swine fever virus strain 94.4/IL/94/TWN[1-500]	500	Classical swine fever virus 39[1-400]	400	0,43×400
Classical swine fever virus strain 94.4/IL/94/TWN[1-600]	600	Classical swine fever virus 39[1-500]	500	0,45×500

References

[1] S. Altschul, W. Gish, W. Miller, E. Myers and D. Lipman, Basic local alignment search tool, Journal of Molecular Biology, **215(3)** (1990), 403–410.

[2] X. Huang and W. Miller, A time-efficient, linear-space local similarity algorithm, Advances in Applied Mathematics, **12** (1991), 337-357.

[3] D. Lipman and W. Pearson, Rapid and sensitive protein similarity searches, Science, **227(4693)** (1985), 1435–1441.

[4] L. Noe and G. Kucherov, YASS: enhancing the sensitivity of DNA similarity search, Nucleic Acids Research, **33** (2005), 540-543.

[5] K. Rasmussen, J. Stoye and E. Myers, Efficient *q*-Gram Filters for Finding All *e*-Matches over a Given Length, Journal of Computational Biology, **13** (2006), 296-308.

[6] T. Smith and M. Waterman, Identification of common molecular subsequences, Journal of Molecular Biology, **147(1)** (1981), 195–197.

[7] M. Waterman and M. Eggert, A new algorithm for best subsequence alignments with application to tRNA-rRNA comparisons, Journal of Molecular Biology, **197** (1987), 723-728.

Received: December, 2011