Alignment with non-overlapping inversions in $O(n^3)$-time

Augusto F. Vellozo[1], Carlos E. R. Alves[2], and Alair Pereira do Lago[3]

[1] Instituto de Matemática e Estatística da Universidade de São Paulo (IME-USP)
Rua do Matão, 1010 - Cidade Universitária CEP:05508-090 São Paulo - SP - Brasil,
vellozo@ime.usp.br
[2] Universidade São Judas Tadeu (FTCE-USJT), Rua Taquari, 546, Mooca
CEP:03166-000 São Paulo - SP - Brasil,
prof.carlos_r_alves@usjt.br
[3] Instituto de Matemática e Estatística da Universidade de São Paulo (IME-USP)
Rua do Matão, 1010 - Cidade Universitária CEP:05508-090 São Paulo - SP - Brasil,
alair@ime.usp.br

Abstract. Alignments of sequences are widely used for biological sequence comparisons. Only biological events like mutations, insertions and deletions are usually modeled and other biological events like inversions are not automatically detected by the usual alignment algorithms.
Alignment with inversions does not have a known polynomial algorithm and a simplification to the problem that considers only non-overlapping inversions were proposed by Schöniger and Waterman [20] in 1992 as well as a corresponding $O(n^6)$ solution[4]. An improvement to an algorithm with $O(n^3 \log n)$-time complexity was announced in an extended abstract [1] and, in this present paper, we give an algorithm that solves this simplified problem in $O(n^3)$-time and $O(n^2)$-space in the more general framework of an edit graph.
Inversions have recently [4, 7, 13, 17] been discovered to be very important in Comparative Genomics and Scherer et al. in 2005 [11] experimentally verified inversions that were found to be polymorphic in the human genome. Moreover, 10% of the 1,576 putative inversions reported overlap RefSeq genes in the human genome. We believe our new algorithms may open the possibility to more detailed studies of inversions on DNA sequences using exact optimization algorithms and we hope this may be particularly interesting if applied to regions around known rearrangements boundaries. Scherer report 29 such cases and prioritize them as candidates for biological and evolutionary studies.

1 Introduction

Alignments of sequences are widely used for biological sequence comparisons and can be associated with a set of edit operations that transform one sequence to the other. Usually, the only edit operations that are considered are the *substitution* (mutation) of one symbol by another one, the *insertion* of one symbol

[4] In this case, n denotes the maximal length of the two aligned sequences.

and *deletion* of one symbol. If costs are associated with each operation, there is a classic $O(n^2)$ dynamic program that computes a set of edit operations with minimal total cost and exhibit the associated alignment, which has good quality and high likelihood for realistic costs.

Other important biological events like inversions are not automatically detected by the usual alignment algorithms and we can define a new edit operation, the *inversion* operation, which substitutes any segment by its *reverse complement* sequence. We can define a new alignment problem: given two sequences and fixed costs for each kind of edit operation, the *alignment with inversions* problem is an optimization problem that queries the minimal total cost[5] of an edit operations series that transforms one sequence to the other. Moreover, one may also be interested in the exhibition of its corresponding alignment and/or edit operations. Unfortunatley, the decision problem associated with alignment with inversions for an unlimited alphabet size is NP-hard as consequence of Jiang et al. [5].

Some simplifications of this problem have been studied and were proved to be NP-complete [3, 22]. Many approximation algorithms were also proposed [6, 16]. Another important simplification is the problem known as *sorting signed permutations by reversals* and polynomial algorithms were obtained in a sequence of papers [2, 14, 15, 21]. These approaches are mainly used for the study of inversions on sequences of genes, but new comparative results given by Sherer et al. [11] show also the importance of DNA inversion studies where those methods can not be used. Moreover, Sherer et al. reported 83 inversions that are contained within a gene.

Another important approach was introduced in 1992, by Schöniger and Waterman [20]. They introduced a *simplification hypothesis*: *all regions involved in the inversions do not overlap.* This simplification is realistic for local DNA comparisons on relatively close sequences. This led to the *alignment with non-overlapping inversions* problem and they presented a simple $O(n^6)$ dynamic programming solution for this problem and also introduced a *heuristic* for it that reduced the average running-time to something between $O(n^2)$ and $O(n^4)$.

Recently, independent works [8–10, 12] gave exact algorithms for alignments with non-overlapping inversions with $O(n^4)$-time and $O(n^2)$-space complexity. An algorithm with $O(n^3 \log n)$-time [1] was later announced. In this paper, we give an algorithm that solves this simplified problem in $O(n^3)$-time and $O(n^2)$-space.

2 Alignments with non-overlapping inversions

The standard alignment of two strings is called *standard alignment* in this text. This kind of alignment, when viewed as the process of transforming a string s in a string t, uses the well known string edit operations of insertion, deletion and substitution of symbols.

[5] In this work, we deal with the dual approach of maximization of similarity score.

The alignment of s and t is usually represented by the insertion of some *spaces* ($-$) in certain places of each string and the matching (alignment) of each symbol or space of s with the symbol or space in the corresponding position in t. If $s[i]$ and $t[j]$ are symbols from s and t, respectively, then a pair $(s[i], t[j])$ is a *substitution* of $s[i]$ by $t[j]$ (if they are equal we say it's a *match*), $(-, t[j])$ is the *insertion* of $t[j]$ and $(s[i], -)$ is the deletion of $s[i]$. Usually, there are costs associated with each edit operation and a score is given to the alignment based on the pairs that were formed.

An extra operation is considered here: the *inversion* of a substring. A string that suffers this operation has a substring removed, reverted, complemented and inserted back in its original place. For example, the inversion of the string ACCATGC gives GCATGGT.

When evaluating an alignment with inversions, there is a cost associated with the inversion operation. Besides that, insertions, substitutions and deletions may be applied in an inverted substring, incurring in additional costs.

In this paper we consider only *non-overlapping* inversions. This means that when aligning two strings we may consider multiple inversions in s, but any symbol of s may be involved in at most one inverted substring. When dealing with non-overlapping inversions, the order in which the inversions are performed is unimportant.

In the following sections, $\overline{s}$ is the inverted string s while $\overline{s[a..b]}$ is the inverted substring os s that starts in position a and ends in position b. These positions are taken from s, not $\overline{s}$, as would be the case in $\overline{s}[a..b]$ (notice the extension of the bar in each case).

3 Edit Graph

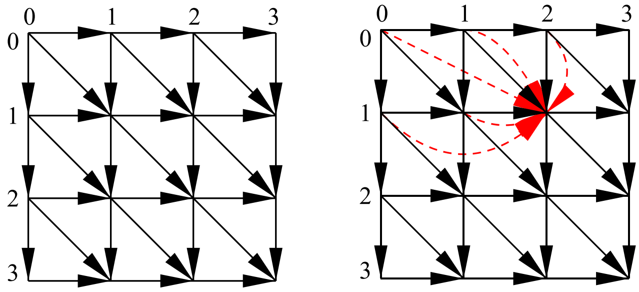
Let s and t be two sequences of lengths n and m respectively.

Definition 31 (Edit Graph of s and t) *Consider* $V = \{(i, j) | 0 \leq i \leq n, 0 \leq j \leq m\}$ *and* $E = E_H \cup E_D \cup E_V$*, such that,*

- $E_H = \{e_H^{i,j} = ((i, j-1), (i, j)) | 0 \leq i \leq n, 0 < j \leq m\}$ *is the set of horizontal edges that end on vertex* (i, j)*,*
- $E_D = \{e_D^{i,j} = ((i-1, j-1), (i, j)) | 0 < i \leq n, 0 < j \leq m\}$ *is the set of diagonal edges that end on vertex* (i, j)*,*
- $E_V = \{e_V^{i,j} = ((i-1, j), (i, j)) | 0 < i \leq n, 0 \leq j \leq m\}$ *is the set of vertical edges that end on vertex* (i, j)*.*

Consider the function $\omega : E \longrightarrow \mathbb{R} \cup \{-\infty\}$*, that associates each edge* $e \in E$ *with weight* $\omega(e)$*. The directed graph* $G = (V, E, \omega)$ *is the edit graph of* s *and* t*.*

In this work, the weight of edge $e_V^{i,j}$ is the score of the deletion of letter $s[i]$ when $s[1..i-1]$ is aligned with $t[1..j]$, the weight of edge $e_H^{i,j}$ is the score of the insertion of letter $t[j]$ when $s[1..i]$ is aligned with $t[1..j-1]$ and the weight of edge $e_D^{i,j}$ is the score of the substitution of letter $s[i]$ by letter $t[j]$ when



(a) Edit graph (b) Extended edit graph

Fig. 1. Examples of edit graph and extended edit graph. Edge weights are not shown, and the only extended edges shown are those that arrive at $(1, 2)$.

$s[1..i-1]$ is aligned with $t[1..j-1]$. These weights are usually defined by a function $\phi : \Sigma \cup \{-\} \times \Sigma \cup \{-\} \longrightarrow \mathbb{R} \cup \{-\infty\}$, $- \notin \Sigma$, such that $\omega(e_V^{i,j}) = \phi(s[i], -)$, $\omega(e_H^{i,j}) = \phi(-, t[i])$ and $\omega(e_D^{i,j}) = \phi(s[i], t[i])$, where Σ is the set of symbols used in the sequences.

Therefore, there is a one-to-one relation between paths in G and standard alignments of s against t. In others words, one path from $(0, 0)$ to (i, j) in G corresponds to one and only one standard alignment of $s[1..i]$ against $t[1..j]$. The score of an alignment without inversions is the total weight of its corresponding path in G.

We say that a path p from $u = (i, j)$ to $v = (i', j')$ is optimal if there is no other path from u to v with total weight greater than the weight of p. We denote $w_u^v = w_{i,j}^{i',j'}$ to be the weight of this optimal path path p. If there is no such a path from u to v, we denote $w_u^v = -\infty$.

Notice that the score of an optimal standard alignment of s against t is the weight of an optimal path from $(0, 0)$ to (i, j) in G.

Definition 32 (Extended edit graph of s and t) *Consider E_H, E_D, E_V and V as described in the definition of edit graph of s and t. Consider $E = E_H \cup E_D \cup E_V \cup E_X$ where $E_X = \bigcup_{i=0}^{n} \bigcup_{j=0}^{m} E_X^{i,j}$ and $E_X^{i,j}$ is the set of extended edges that end on vertex (i, j), that is*

$$E_X^{i,j} = \{e_{i',j'}^{i,j} = ((i', j'), (i, j)) \mid 0 \le i' \le i \le n, 0 \le j' \le j \le m \;\; e \;\; (i', j') \ne (i, j)\}.$$

The directed graph $G = (V, E, \omega)$ is the extended edit graph of s and t and the weight function ω is defined like in the edit graph, but extended to assign weights to the extended edges.

In this paper, the extended edges represent optimal standard alignments of substrings of t against inverted substrings of s.

Let G be an extended edit graph of s and t. The graph obtained by removing the extended edges from G is an edit graph of s and t. Like in edit graphs, an optimal path in an extended edit graph is a path with maximal weight.

4 The algorithm

Let $s = s[1..n]$ and $t = t[1..m]$ be the sequences to be aligned.

Let $\overline{G} = (V, \overline{E}, \overline{\omega})$ be the edit graph of $\overline{s}$ and t. This graph is used to evaluate the alignments of substrings of t and inverted substrings of s. In $\overline{G}$, the weights $\overline{\omega}(e_H^{i,j})$, $\overline{\omega}(e_D^{i,j})$ and $\overline{\omega}(e_V^{i,j})$ correspond, respectively, to the scores of insertion of $t[j]$, substitution of $\overline{s}[i] = \overline{s[n+1-i]}$ by $t[j]$ and deletion of $\overline{s}[i] = \overline{s[n+1-i]}$.

Let $G = (V, E, \omega)$ be the extended edit graph of s and t, such that

$$\begin{aligned}
\omega(e_H^{i,j}) &= \text{ score of insertion of } t[j],\\
\omega(e_V^{i,j}) &= \text{ score of deletion of } s[i],\\
\omega(e_D^{i,j}) &= \text{ score of substitution of } s[i] \text{ by } t[j],\\
\omega(e_{i',j'}^{i,j}) &= \overline{w}_{(n-i,j')}^{(n-i',j)} + \omega_{inv},
\end{aligned}$$

where ω_{inv} is a penalty value for inversions and $\overline{w}_{(n-i,j')}^{(n-i',j)}$ is the weight of an optimal path from $(n-i, j')$ to $(n-i', j)$ in $\overline{G}$. In others words $\overline{w}_{(n-i,j')}^{(n-i',j)}$ is the score of the standard alignment of $\overline{s[i'+1..i]}$ against $t[j'+1..j]$.

Since there is a one to one relation between paths in G and alignments with non-overlapping inversions of s against t, the weight of an optimal path from $(0,0)$ to (n,m) in G is the score of an optimal alignment with non-overlapping inversions of s against t.

The following definitions help us to understand how the weight of an optimal path from $(0,0)$ to (n,m) in G is obtained through Algorithm 1.

Definition 41 (Matrix B) *$B[i,j] = w_{0,0}^{i,j}$ is the weight of an optimal path from $(0,0)$ to (i,j) on G, $0 \le i \le n$ and $0 \le j \le m$.*

In others words $B[i,j]$ is the score of an optimal alignment with non-overlapping inversions of $s[1..i]$ against $t[1..j]$.

Definition 42 (Matrix $Out_{i'}^{i}$) *Given i' and i such that $0 \le i' \le i \le n$ we define the matrix $Out_{i'}^{i}[1..m, 1..m]$ of G as*

$$Out_{i'}^{i}[j',j] = \begin{cases} B[i',j'] + w_{i',j'}^{i,j}, & \textit{if } 0 \le j' \le j \le m,\\ -\infty & \textit{if } 0 \le j < j' \le m, \end{cases}$$

The element $Out_{i'}^{i}[j',j]$ stores the optimal alignment score of $s[1..i]$ against $t[1..j]$ such that $\overline{s[i'+1..i]}$ is aligned with $t[j'+1..j]$.

Definition 43 ($hDif_{i'}^{i,j}$ vector) *Let G be an edit graph. Given i' and the vertex (i,j) of G such that $0 \le i' \le i$, we define $hDif_{i'}^{i,j}$ of G by the vector of size j such that $hDif_{i'}^{i,j}[j'] = w_{i',j'}^{i,j} - w_{i',j'}^{i,j-1}$, $0 \le j' < j$.*

The vector $hDif_{i'}^{i,j}$ has an important property that is used by our algorithm: it is nondecreasing.

Lemma 44 *The vector $hDif_{i'}^{i,j}$ of an edit graph G is nondecreasing.*

Proof. Let (i', j_1), (i', j_2), (i, j_3) and (i, j_4) be vertices of G, such that $0 \leq j_1 < j_2 \leq j_3 < j_4 \leq m$. There is at least one common vertex v that belongs to the paths from (i', j_2) to (i, j_3) and from (i', j_1) to (i, j_4), as one can see at Figure 2. To simplify, we define: $a = w_{i',j_1}^{i,j_3}$, $b = w_{i',j_2}^{i,j_4}$, $c = w_{(i',j_1)}^{v}$, $d = w_v^{(i,j_4)}$, $e = w_{(i',j_2)}^{v}$ and $f = w_v^{(i,j_3)}$. As a and b are the optimal path scores then $a \geq c + f$ and $b \geq e + d$. Adding the two previous inequalities we have $a + b \geq c + f + e + d \Rightarrow b - (e + f) \geq (c + d) - a$. Consider $j_3 = j_4 - 1$. Therefore $w_{i',j_2}^{i,j_4} - w_{i',j_2}^{i,j_4-1} \geq w_{i',j_1}^{i,j_4} - w_{i',j_1}^{i,j_4-1} \Rightarrow hDif_{i'}^{i,j_4}[j_2] \geq hDif_{i'}^{i,j_4}[j_1]$. ∎

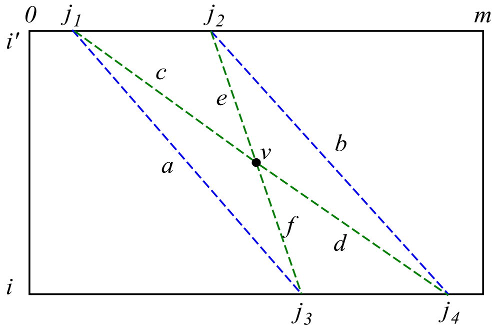


Fig. 2. Illustration of the proof of Lemma 44

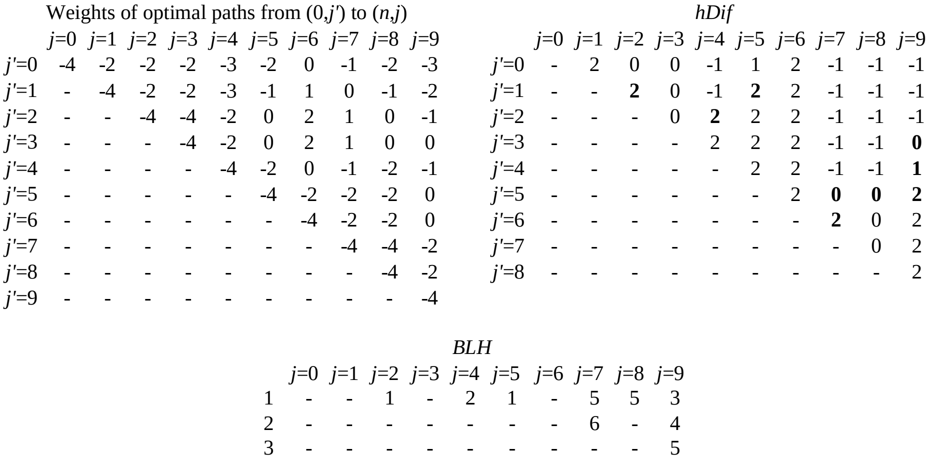
The number of times that $hDif_{i'}^{i,j}[j']$ increases when we sweep through $hDif_{i'}^{i,j}$ from $j' = 0$ to $j - 1$ is called $\psi H_{i'}^{i,j}$.

Usually, the adopted score system has integer values: r for rewarding a match, q for a mismatch and E for a gap. Usually $2E \leq q < r$. Using the edit graph notation, the weights of the edges can be defined as $\omega(e_D^{i,j}) = r$ if $s[i] = t[j]$, $\omega(e_D^{i,j}) = q$ if $s[i] \neq t[j]$ and $\omega(e_H^{i,j}) = \omega(e_V^{i,j}) = E \; \forall (i, j)$. In these cases $\psi H_{i'}^{i,j} \leq r - 2E$, so $\psi H_{i'}^{i,j}$ is limited by a constant. For instance, if the score system is the LCS (Longest Common Subsequence), $r = 1$ and $q = E = 0$, then $\psi H_{i'}^{i,j} \leq 1$. The Figure 3 shows a case where $\psi H_{i'}^{i,j} \leq 3$.

In this text, we consider $\psi H_{i'}^{i,j}$ limited by a constant.

We store the values of j' where occur each increment of $hDif_{i'}^{i,j}$ in a matrix called $BLH_{i'}^{i}$.

Definition 45 (**$BLH_{i'}^{i}$ matrix**) *Given i' and i such that $0 \leq i' \leq i \leq n$, we define the column j, $0 \leq j \leq m$, of $BLH_{i'}^{i}$ as a vector of size $\psi H_{i'}^{i,j}$ such that $BLH_{i'}^{i}[\alpha, j]$ is the α-th j' where $hDif_{i'}^{i,j}[j'] \neq hDif_{i'}^{i,j}[j' - 1]$, for j' from 1 to $j - 1$.*



Weights of optimal paths from (0,*j*') to (*n*,*j*)

	j=0	*j*=1	*j*=2	*j*=3	*j*=4	*j*=5	*j*=6	*j*=7	*j*=8	*j*=9
j'=0	-4	-2	-2	-2	-3	-2	0	-1	-2	-3
j'=1	-	-4	-2	-2	-3	-1	1	0	-1	-2
j'=2	-	-	-4	-4	-2	0	2	1	0	-1
j'=3	-	-	-	-4	-2	0	2	1	0	0
j'=4	-	-	-	-	-4	-2	0	-1	-2	-1
j'=5	-	-	-	-	-	-4	-2	-2	-2	0
j'=6	-	-	-	-	-	-	-4	-2	-2	0
j'=7	-	-	-	-	-	-	-	-4	-4	-2
j'=8	-	-	-	-	-	-	-	-	-4	-2
j'=9	-	-	-	-	-	-	-	-	-	-4

hDif

	j=0	*j*=1	*j*=2	*j*=3	*j*=4	*j*=5	*j*=6	*j*=7	*j*=8	*j*=9
j'=0	-	2	0	0	-1	1	2	-1	-1	-1
j'=1	-	-	**2**	0	-1	**2**	2	-1	-1	-1
j'=2	-	-	-	0	**2**	2	2	-1	-1	-1
j'=3	-	-	-	-	2	2	2	-1	-1	**0**
j'=4	-	-	-	-	-	2	2	-1	-1	**1**
j'=5	-	-	-	-	-	-	2	**0**	**0**	**2**
j'=6	-	-	-	-	-	-	-	**2**	0	2
j'=7	-	-	-	-	-	-	-	-	0	2
j'=8	-	-	-	-	-	-	-	-	-	2

BLH

	j=0	*j*=1	*j*=2	*j*=3	*j*=4	*j*=5	*j*=6	*j*=7	*j*=8	*j*=9
1	-	-	1	-	2	1	-	5	5	3
2	-	-	-	-	-	-	-	6	-	4
3	-	-	-	-	-	-	-	-	-	5

Fig. 3. In this example we used the sequences $s = AATG$ and $t = TTCATGACG$ to build an edit graph G. All vertical and horizontal edges of G have weight -1, the weight $\omega(e_D^{i,j}) = -1$ if $s[i] \neq t[j]$ and $\omega(e_D^{i,j}) = 1$ if $s[i] = t[j]$.

algorithm 1 Algorithm $O(n^3)$ that builds matrix B

```
BimN3($s, t$)
for $i$ from $0$ to $|s|$ do
        ▷ Get the optimal path ended with non-extended edges
        if $i = 0$ then
                $B[0, 0] \leftarrow 0$
        else    $B[i, 0] \leftarrow B[i-1, 0] + \omega(e_V^{i,j})$
        for $j$ from $1$ to $|t|$ do
                if $i = 0$ then
                        $B[0, j] \leftarrow B[0, j-1] + \omega(e_H^{i,j})$
                else    $aux \leftarrow \max(B[i, j-1] + \omega(e_H^{i,j}), B[i-1, j] + \omega(e_V^{i,j}))$
                        $B[i, j] \leftarrow \max(aux, B[i-1, j-1] + \omega(e_D^{i,j}))$
        ▷ Get the optimal path ended with extended edges
        for $i'$ from $i$ downto $0$ do
                $BLH \leftarrow buildBlh(\overline{G}, BLH, i')$
                $maxOut_{i'}^{i} \leftarrow getMaxOut(BLH, B, i')$
                for $j$ from $0$ to $|t|$ do
                        $B[i, j] \leftarrow \max(B[i, j], maxOut_{i'}^{i}[j] + \omega_{inv})$
return $B$
```

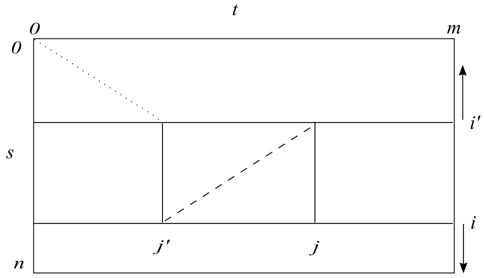


Fig. 4. Execution of Algorithm 1. The dotted line is a path from $(0,0)$ to (i',j') in G. The dashed line represents an alignment of $s[i'+1..i] \times t[j'+1..j]$.

The elements of matrix $BLH_{i'}^{i}$ are called *borderline points* in [18]. Figure 3 shows an example of $hDif$ and BLH.

Algorithm 1 builds matrix B and Figure 4 shows its execution.

The function $buildBlh(\overline{G}, BLH, i')$ builds the $BLH_{i'}^{i}$ matrix. It was developed based on the algorithm described in section 6 of [19] and runs in $O(m)$ time. Remembering that each column of a borderline matrix has $O(1)$ elements, the function $buildBlh(\overline{G}, BLH, i')$ builds each column of $BLH_{i'}^{i}$ based on the respective column of matrix $BLH_{i'+1}^{i}$ in constant time.

The function $getMaxOut(BLH, B, i')$ returns a vector with the maximum value of each column of $Out_{i'}^{i}$ in $O(m)$ time and was developed based on the algorithm described in subsection 6.2 of [18]. The linear time complexity of this function is attained through a procedure that sweeps through $BLH_{i'}^{i}$ and line i' of matrix B, both with $O(m)$ data.

Using these functions one can see that Algorithm 1 is correct and runs in $O(n^2m)$ time ($O(n^3)$ time, if $m = O(n)$).

5 Experiments

We implemented Algorithm 1 in Java. We worked with two sequences pair of different lengths, 867 and 95.319 bp (base pairs) of human and chimpanzee. These sequences are cited in [11]. The human/chimp sequences were downloaded from the University of California at Santa Cruz website (http://genome.ucsc.edu/). The sequences were taken from the November 2003 chimpanzee (panTro1) genome assembly and the May 2004 (hg17) human genome assembly[6].

The shortest pair is formed by human genome chr7:95119414-95120280 and chimpanzee genome chr6:96726524-96727390. The alignment obtained by the algorithm shows 98,6% of total identities and an inversion involving chr7-95119717-95119979 of human and chr6:96726825-96727087 of chimpanzee.

The longest pair is formed by human genome chr7:80523522-80618840 and chimpanzee genome chr6:81751455-81846825. To cope with sequences of this length faster we broke the sequences into fragments of 100 pairs each.

[6] `http://genome.ucsc.edu/cgi-bin/hgTrackUi?hgsid=59218717&g=netPanTro1`

The fragments were submitted to a standard alignment procedure, such that each fragment from the human genome was aligned against every fragment of the chimpanzee genome twice: inverted and not inverted. Our algorithm was used considering the sequences like sequences of fragments instead of sequences of base pairs. A match between two fragments occurs when their alignment has a score greater than a threshold. The alignment obtained by the algorithm shows 94,8% of total matches and an inversion involving chr7-80553522-80588821 of human and chr6:81781455-81816854 of chimpanzee. One can see this inversion at Figure 5 for fragment size 1000 for better resolution.

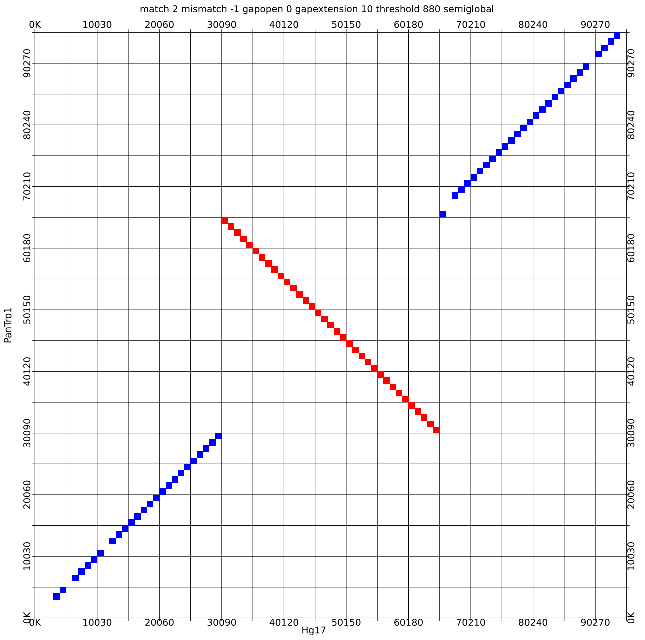


Fig. 5. Alignment of human genome chr7:80523522-80618840 and chimpanzee genome chr6:81751455-81846825. Fragment size is 1000 for better visualization.

We also tested the algorithm on simulated data for random DNA sequences with length in average 700. Each pair of sequences differ from each other by a number of indels ranging from 5% to 10%, mismatches ranging from 5% to 15%, and number of non-overlapping inversions ranging from 1 to 15. We obtained consistent results and detected all the inversions as one would expect.

We also implemented in Java the $O(n^3 \log n)$ algorithm described in [1], the $O(n^4)$ algorithm described in [9] and the sparse algorithm described in [10] that has complexity $O(r^2 \log^2 r)$, where $r = O(n^2)$ is the number of matches between symbols in one sequence against symbols in the other sequence. The tests showed that Algorithm 1 is, as it is expected, always faster than the algorithm

$O(n^3 \log n)$, which is in turn always faster than the algorithm $O(n^4)$. If the sequences to be aligned were DNA sequences then Algorithm 1 was faster than sparse algorithm, but if the sequences to be aligned were sequences of DNA fragments, where the number of matches is small, then the sparse algorithm was faster than the Algorithm 1.

6 Conclusion

In this paper we described a new algorithm that solves the alignment with non-overlapping inversions problem in $O(n^3)$-time and $O(n^2)$-space. We hope that this speed up opens the possibility to studies of inversions on DNA sequences by an exact optimization algorithm. Algorithms that are applied to the study of inversions of sequences of genes cannot be applied in theses cases, since they do not allow repetitions of symbols, nor insertions, nor deletions.

Our algorithm may be particularly interesting when applied to regions around known rearrangement boundaries, since many biologists suppose that inversions at DNA level are very probable in these cases.

Many studies have been done with inversions in DNA sequences.

7 Acknoledgements

This work was supported by Proj. Pronex-FAPESP/CNPq proc. 2003/09925-5.

References

1. Carlos E. R. Alves, Alair Pereira do Lago, and Augusto F. Vellozo. Alignment with non-overlapping inversions in $O(n^3 \log n)$-time. In *Proceedings of GRACO2005*, volume 19 of *Electron. Notes Discrete Math.*, pages 365–371 (electronic), Amsterdam, 2005. Elsevier.
2. David A. Bader, Bernard M. E. Moret, and Mi Yan. A linear-time algorithm for computing inversion distance between signed permutations with an experimental study. *Journal of Computational Biology*, 8(5):483–491, 2001.
3. Alberto Caprara. Sorting permutations by reversals and Eulerian cycle decompositions. *SIAM J. Discrete Math.*, 12(1):91–110 (electronic), 1999.
4. Cerdeño-Tárraga, Patrick, Crossman, Blakely, Abratt, Lennard, Poxton, Duerden, Harris, Quail, Barron, Clark, Corton, Doggett, Holden, Larke, Line, Lord, Norbertczak, Ormond, Price, Rabbinowitsch, Woodward, Barrell, and Parkhill. Extensive DNA inversions in the B. fragilis genome control variable gene expression. *Science*, 307(5714):1463–1465, Mar 2005.
5. Xin Chen, Jie Zheng, Zheng Fu, Peng Nan, Yang Zhong, Stefano Lonardi, and Tao Jiang. Assignment of orthologous genes via genome rearrangement. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, 2(4):302–315, 2005.
6. David A. Christie. A 3/2-approximation algorithm for sorting by reversals. In *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms (San Francisco, CA, 1998)*, pages 244–252, New York, 1998. ACM.

7. Cáceres, Ranz, Barbadilla, Long, and Ruiz. Generation of a widespread Drosophila inversion by a transposable element. *Science*, 285(5426):415–418, Jul 1999.
8. A. P. do Lago, C. A. Kulikowski, E. Linton, J. Messing, and I. Muchnik. Comparative genomics: simultaneous identification of conserved regions and their rearrangements through global optimization. In *The Second University of Sao Paulo/Rutgers University Biotechnology Conference*, Rutgers University Inn and Conference Center, New Brunswick, NJ, August 2001.
9. Alair Pereira do Lago, Ilya Muchnik, and Casimir Kulikowski. An $O(n^4)$ algorithm for alignment with non-overlapping inversions. In *Second Brazilian Workshop on Bioinformatics, WOB 2003*, Macaé, RJ, Brazil, 2003. http://www.ime.usp.br/~alair/wob03.pdf.
10. Alair Pereira do Lago, Ilya Muchnik, and Casimir Kulikowski. A sparse dynamic programming algorithm for alignment with non-overlapping inversions. *Theor. Inform. Appl.*, 39(1):175–189, 2005.
11. Feuk, MacDonald, Tang, Carson, Li, Rao, Khaja, and Scherer. Discovery of human inversion polymorphisms by comparative analysis of human and chimpanzee DNA sequence assemblies. *PLoS Genet*, 1(4):e56, Oct 2005.
12. Yong Gao, Junfeng Wu, Robert Niewiadomski1, Yang Wang, Zhi-Zhong Chen, and Guohui Lin. A space efficient algorithm for sequence alignment with inversions. In *Computing and Combinatorics, 9th Annual International Conference, COCOON 2003*, volume 2697 of *Lecture Notes in Computer Science*, pages 57–67. Springer-Verlag, 2003.
13. Graham and Olmstead. Evolutionary significance of an unusual chloroplast DNA inversion found in two basal angiosperm lineages. *Curr Genet*, 37(3):183–188, Mar 2000.
14. Sridhar Hannenhalli and Pavel A. Pevzner. Transforming cabbage into turnip: polynomial algorithm for sorting signed permutations by reversals. In *ACM Symposium on Theory of Computing*, pages 178–189. Association for Computing Machinery, 1995.
15. Sridhar Hannenhalli and Pavel A. Pevzner. Transforming cabbage into turnip: polynomial algorithm for sorting signed permutations by reversals. *J. ACM*, 46(1):1–27, 1999.
16. J. Kececioglu and D. Sankoff. Exact and approximation algorithms for sorting by reversals, with application to genome rearrangement. *Algorithmica*, 13(1-2):180–210, 1995.
17. Kuwahara, Yamashita, Hirakawa, Nakayama, Toh, Okada, Kuhara, Hattori, Hayashi, and Ohnishi. Genomic analysis of Bacteroides fragilis reveals extensive DNA inversions regulating cell surface adaptation. *Proceedings of the National Academy of Sciences U S A*, 101(41):14919–14924, Oct 2004.
18. Gad M. Landau and Michal Ziv-Ukelson. On the common substring alignment problem. *J. Algorithms*, 41(2):338–359, 2001.
19. Jeanette P. Schmidt. All highest scoring paths in weighted grid graphs and their application to finding all approximate repeats in strings. *SIAM J. Comput.*, 27(4):972–992 (electronic), 1998.
20. M. Schöniger and M. S. Waterman. A local algorithm for DNA sequence alignment with inversions. *Bulletin of Mathematical Biology*, 54(4):521–536, Jul 1992.
21. Eric Tannier and Marie-France Sagot. Sorting by reversals in subquadratic time. In *Combinatorial pattern matching*, volume 3109, pages 1–13, 2004. CPM 2004.
22. R. Wagner. On the complexity of the extended string-to-string correction problem. In *Seventh ACM Symposium on the Theory of Computation*. Association for Computing Machinery, 1975.