Alignment with non-overlapping inversions in $O(n^3 \log n)$-time (extended abstract)

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

Abstract

Alignment of sequences is widely used for biological sequence comparisons, and only biological events like mutations, insertions and deletions are usually modeled. Other biological events like inversions are not automatically detected by the usual alignment algorithms.

Alignment with inversions has not a known polinomial algorithm and a simplification to the problem that considers only non-overlapping inversions were proposed by Schöniger and Waterman [15] in 1992 as well as a corresponding $O(n^6)$ solution [1]. Recent works [8,7,5,6] improved these results to $O(n^4)$-time and $O(n^2)$-space complexity. In this present extended abstract, we announce an algorithm that solves this simplified problem in $O(n^3 \log n)$-time and $O(n^2)$-space.

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 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*

[1] In this paper, n denotes the maximal length of the two aligned sequences.

* Departamento de Ciência da Computação da Universidade de São Paulo
**Faculdade de Tecnologia e Ciências Exatas da Universidade São Judas Tadeu

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 of an edit operations set that transforms one sequence to the other. Moreover, one may also be interested in the exhibition of its correspondent alignment and/or edit operations. To the best of our knowledge, the computational complexities of alignment with inversions problem is unknown.

Some simplifications of this problem have been studied and were proved to be NP-complete [16,3]. Many approximation algorithms were also proposed [13,4]. Another important simplification is the problem known as *sorting signed permutations by reversals* and polynomial algorithms were obtained [9,10,11,12,2].

Another important approach was introduced in 1992, by Schöniger and Waterman [15]. They introduced a *simplification hypothesis*: *all regions involving in the inversions do not overlap*. This led to the *alignment with non-overlapping inversions* problem and they presented a $O(n^6)$ 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, indepent works [8,7,5,6] gave exact algorithms for alignments with non-overlapping inversions with $O(n^4)$-time and $O(n^2)$-space complexity. In this present extended abstract, we announce an algorithm that solves this simplified problem in $O(n^3 \log n)$-time and $O(n^2)$-space.

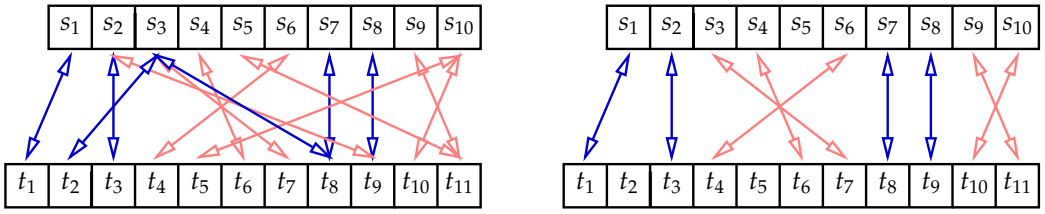
2 The Algorithm

Let $\omega : A \times A \longrightarrow \mathbb{R} \cup \{-\infty\}$ be any *weight* function, where A is the alphabet. Let $\_$ be the inversion operation. Let $s = s_1 \cdots s_k$ and $t = t_1 \cdots t_{k'}$ be two words. We define the *matching graph of s and t* as the weighted and colored bipartite graph $G = G(s, t, \omega, \_) = (V, E)$, where $V = \{s_1, \ldots, s_k, t_1, \ldots, t_{k'}\}$ and E is a double copy of $K_{|s|,|t|}$: for any pair of vertices s_i and t_j we link them with one *blue(dark gray) edge of weight* $\omega(s_i, t_j)$ and one *red(light gray) edge of weight* $\omega(s_i, \overline{t_j})$. An edge with weight that is not $-\infty$ is called a *match*. A match is called *direct match* if it is blue and it is called *inverted match* if it is red.

Given $u \in V^*$ a nonempty factor of s and $v \in V^*$ a nonempty factor of t, we call $B = (u, v)$ a *block*. We say that the edges (s_i, t_j) and (s'_i, t'_j) *cross* each other if $(i-i')(j-j') < 0$, we say that they *touch* each other if $(i-i')(j-j') = 0$ and we say that they are *parallel* if $(i - i')(j - j') > 0$. Let $M \subseteq E$ be any set of matches edges in a matching graph. M is called a *matching* if any two edges of M do not touch each other. Moreover, M is called a *direct matching* if it

has only direct matches and any two are parallels. Furthermore, M is called an *inverted matching* if it has only inverted matches and any two of them cross each other. The *restriction of M to a block $B = (s[i'..i], t[j'..j])$* is the submatching of all edges of M with vertices in $s[i'..i]$ and $t[j'..j]$). Finally, M is called a *blockwise inverted matching*, or simply a *bimatching* if

- there is $l \geq 1$;
- there are l blocks $B_i = (u_i, v_i)$ such that $s = s_1 s_2 \dots s_k = u_1 u_2 \dots u_l$ and $t = t_1 t_2 \dots t_{k'} = v_1 v_2 \dots v_l$;
- for any $i \in \{1, \dots, l\}$, the restriction M_i of M to the block B_i is either a direct matching or an inverted matching;
- $M = \cup_{i=1}^{l} M_i$.



(a) Matching graph (edges with weight $-\infty$ are not drawn)

(b) Bimatching with four blocks

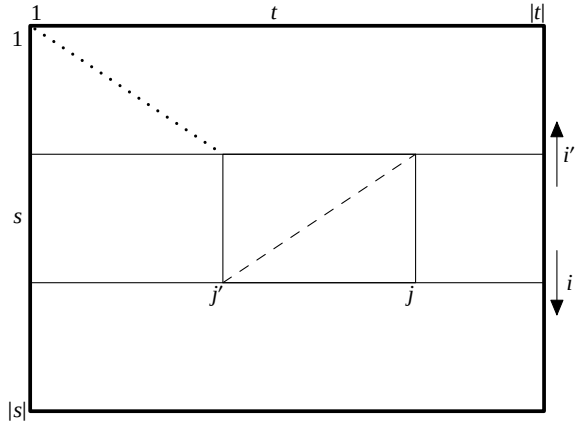
Fig. 1. Examples of Matching graph and bimatching

It is quite common for alignments the establishment of penalties for biological events like mutations, insertions or deletions. In order to be more general, we attribute a *inversion penalty* $I \geq 0$ for every inverted submatching M_i. Hence, we define the *weight of a bimatching* M as: $\omega(M) = \sum_{e \in M} \omega(e) - \iota(M)I$, where $\iota(M)$ is the *number of inversions of* M. Penalties for gaps can also be easily introduced [6]. We are interested in solving the following problem:

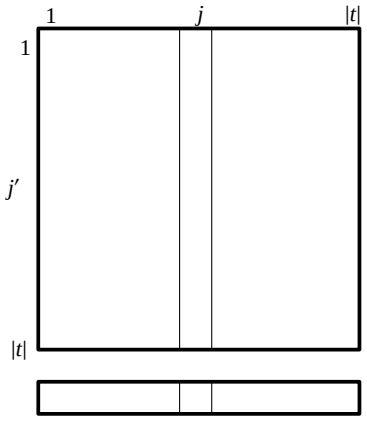
Problem 1 *Given a matching graph $G = G(s, t, \omega, \_)$, we want to compute the maximal weight $\omega(M)$ for all possible bimatchings M. As usual, we may also be interested in a bimatching of maximal weight.*

Such a bimatching M^* of maximal weight is called an *optimal bimatching* of s and t. Its weight $\omega(M^*)$ is denoted by $\mathrm{BIM}(s, t)$ and is computed by Algorithm 1. As usual, one can easily recover M^*. Detailed definitions for this problem and solutions can be found at [7,6].

Algorithm 1 performs as follows. The loop controled at line 3 computes row i of matrix B. The values attributed to this row at line 5 consider the case in which the last block of an optimal bimatching associated with $B[i, j]$ is direct. At line 6, we fix i' and the weights of the alignments of $\overline{s[i'..i]}$ and all factors of t are stored in matrix R. In fact, at line 7 a data structure that represents



(a) Obtainment of row i of $B_{|s|\times|t|}$



(b) Obtainment of $M_{|t|}$ from $A_{|t|\times|t|}$

Fig. 2. Algorithm representation. A bimatching of $s[1\,..\,i]$ and $t[i\,..\,j]$ may be composed by a bimatching $B[i'-1, j'-1]$ (dotted line) and an inverted matching $R[j', j]$ (dashed line).

Algorithm 1 A $O(n^3 \log n)$-time and $O(n^2)$-space algorithm for BIM

BIM(s, t)

1 ▷ Compute the table $B[i, j] = \text{BIM}(s[1\,..\,i], t[1\,..\,j])$
2 Let $B[i, j]$ be 0 for $i = 0$ or $j = 0$
3 **for** i **from** 1 **to** $|s|$ **do**
4 **for** j **from** 1 **to** $|t|$ **do**
5 $B[i, j] \leftarrow \max(B[i-1, j], B[i, j-1], B[i-1, j-1] + \omega(s[i], t[j]))$
6 **for** i' **from** i **downto** 1 **do**
7 $R_{|t|\times|t|} \leftarrow \mathit{AllFactorAlignment}(\overline{s[i'\,..\,i]}, t)$
8 ▷ $R[j', j]$ = score of the alignment of $\overline{s[i'\,..\,i]}$ and $t[j'\,..\,j]$
9 Define $A_{|t|\times|t|}$ from $R_{|t|\times|t|}$ and row $i'-1$ of B
10 ▷ $A[j', j] = R[j', j] + B[i'-1, j'-1]$
11 $M_{|t|} \leftarrow \mathit{ColumnMaxima}(A_{|t|\times|t|})$
12 **for** j **from** 1 **to** $|t|$ **do**
13 $B[i, j] \leftarrow \max(B[i, j], M[j] - I)$
14 **return** B

R is incrementally built in time $O(n \log n)$ [14] from the data structure built at the previous iteration at line 6. Queries are done in $O(\log n)$ time.

Based on R we now consider matrix A as stated in the algorithm. Matrix A is not actually built, but queries are done in $O(\log n)$ time using R and B. Element $A[j', j]$ is the score of a best bimatching in the matching graph of $s[1..i]$ and $t[1..j]$ where the last block is $(s[i'..i], t[j'..j])$ and its corresponding restriction is an inverted matching. Next, we solve a Column Maxima Problem in matrix A at line 11 (i.e., we find the maximum of each column of A) in order to compute, for all j, the score of the best bimatching for $s[1..i]$ and $t[1..j]$ where the last block is $(s[i'..i], t[j'..j])$ for some j'. At line 13 we update row i of B from bimatchings considered in M. Matrix A is totally monotone, so the Column Maxima Problem can be solved with $O(n)$ queries in $O(n \log n)$ time [1].

3 Conclusion

We gave a new algorithm for the alignment with non-overlapping inversions problem, improving the time complexity of an exact solution from $O(n^4)$ to $O(n^3 \log n)$.

References

[1] Alok Aggarwal, Maria M. Klawe, Shlomo Moran, Peter Shor, and Robert Wilber. Geometric applications of a matrix-searching algorithm. *Algorithmica*, 2(2):195–208, 1987.

[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. In *Algorithms and data structures (Providence, RI, 2001)*, volume 2125 of *Lecture Notes in Comput. Sci.*, pages 365–376. Springer, Berlin, 2001.

[3] Alberto Caprara. Sorting permutations by reversals and Eulerian cycle decompositions. *SIAM J. Discrete Math.*, 12(1):91–110 (electronic), 1999.

[4] 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.

[5] 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.

[6] Alair Pereira do Lago, Ilya Muchnik, and Casimir Kulikowski. A sparse dynamic programming algorithm for alignment with non-overlapping inversions. *Theoretical Informatics and Applications*.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] Haim Kaplan, Ron Shamir, and Robert E. Tarjan. Faster and simpler algorithm for sorting signed permutations by reversals. In *Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (New Orleans, LA, 1997)*, pages 344–351, New York, 1997. ACM.

[12] Haim Kaplan, Ron Shamir, and Robert E. Tarjan. A faster and simpler algorithm for sorting signed permutations by reversals. *SIAM J. Comput.*, 29(3):880–892 (electronic), 2000.

[13] 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.

[14] 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.

[15] 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.

[16] 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.