

Alignments with Non-overlapping Moves, Inversions and Tandem Duplications in $O(n^4)$ Time

Christian Ledergerber

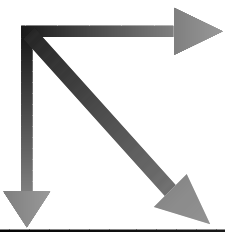
Christophe Dessimoz

ledergec@student.ethz.ch

Group of Gaston Gonnet (CBRG)

Department of Computer Science

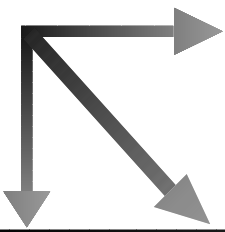
ETH Zurich



Alignments are used to

- compute the similarity of two strings
- identify homologous characters

Therefore edit operations should model evolutionary events.



Standard alignment:

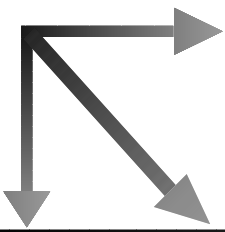
$$NW(ACCE, AAE) = \begin{array}{cccc} A & C & C & E \\ A & _ & A & E \end{array}$$

Score of standard alignment:

$$\delta(ACCE, AAE) = \sigma(A, A) + \sigma_I + \sigma(C, A) + \sigma(E, E)$$

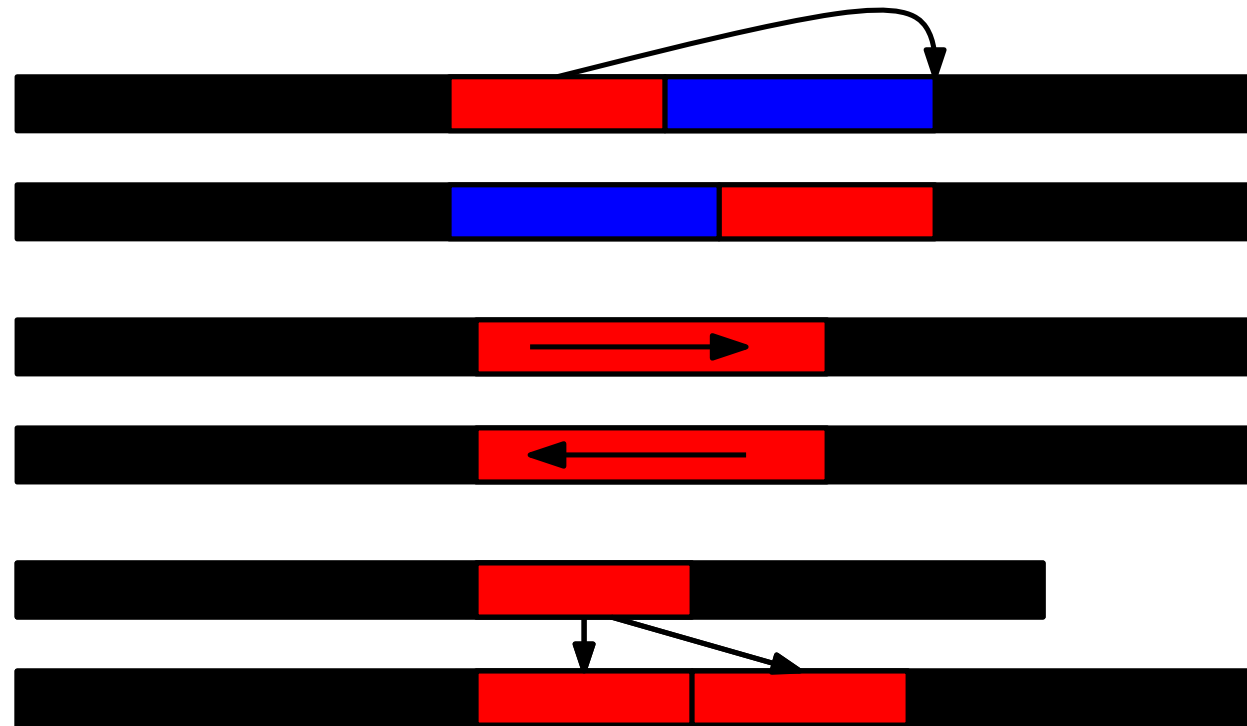
Edit Operations:

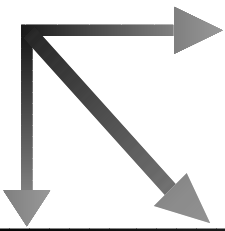
{Insertion, Deletion, Substitution}



Motivation

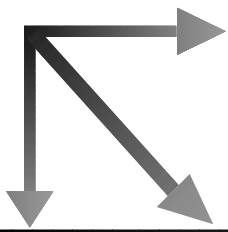
What about moves, inversions and tandem duplications?





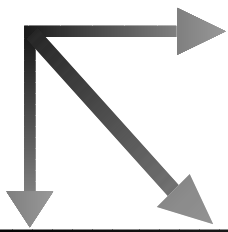
Outline

- Previous Work
- Definition of the Problem
- Outline of the Algorithm
- Results
- Example



General move operations are NP-complete
[Shapira and Storer, 2002]

- $O(\log n \log^* n)$ factor approximation in $O(n \log n)$ time. [Cormode and Muthukrishnan, 2002]
- Heuristic [Fliess et al., 2002]
- Non-overlapping inversions in $O(n^3)$ time due to work of
[Schoeninger and Waterman, 1992, Vellozo et al., 2006]

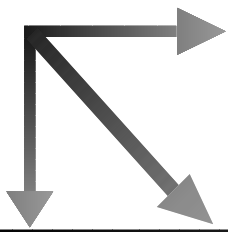


Definition of δ_c

- Let $\delta(S, T)$ denote the standard alignment score.
- Cyclic string comparison score is

$$\delta_c(S, T) = \max\{\delta(S_1, T_2) + \delta(S_2, T_1) \mid S_1S_2 = S, T_1T_2 = T\}$$

$$\delta_c\left(\begin{array}{cc} S_1 & S_2 \\ \text{---} & \text{---} \\ T_1 & T_2 \\ \text{---} & \text{---} \end{array}\right) = \delta\left(\begin{array}{c} S_1 \\ \text{---} \\ T_2 \\ \text{---} \end{array}\right) + \delta\left(\begin{array}{c} S_2 \\ \text{---} \\ T_1 \\ \text{---} \end{array}\right)$$



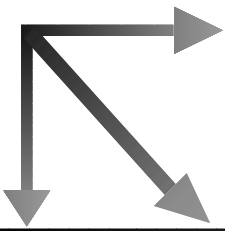
Definition of δ_m

- String comparison with non-overlapping moves:
find $S_1 \dots S_d = S, T_1 \dots T_d = T$ such that

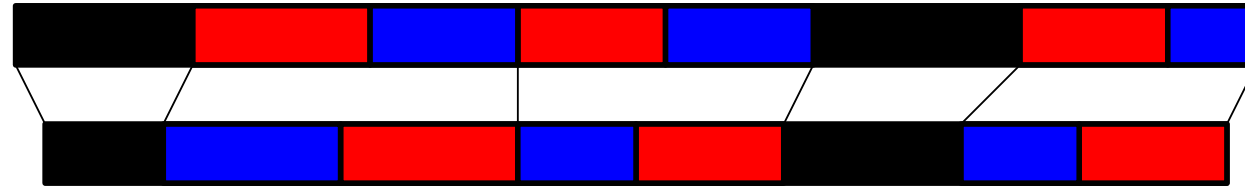
$$\delta_m(S, T) = \sum_{i=1}^d \max\{\delta(S_i, T_i), \delta_c(S_i, T_i) + \sigma_c\} \text{ is maximum}$$

$$\delta_m \left(\begin{array}{c} \overline{S_1 \quad S_2 \quad S_3} \\ \overline{T_1 \quad T_2 \quad T_3} \end{array} \right) =$$

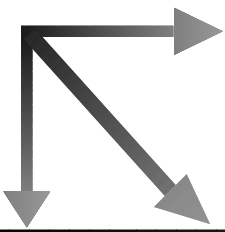
$$\delta \left(\begin{array}{c} \overline{S_1} \\ \overline{T_1} \end{array} \right) + \delta_c \left(\begin{array}{c} \overline{S_2} \\ \overline{T_2} \end{array} \right) + \sigma_c + \delta \left(\begin{array}{c} \overline{S_3} \\ \overline{T_3} \end{array} \right)$$



Example



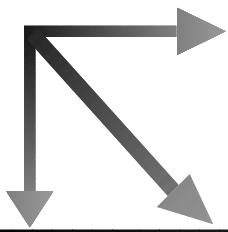
Non-Overlapping



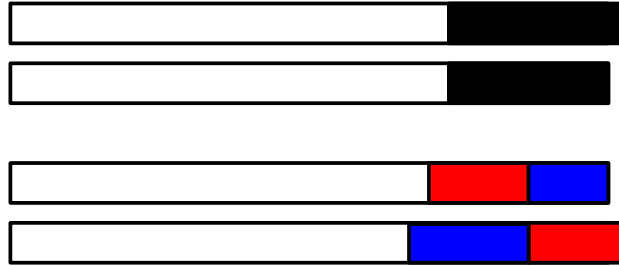
Needleman-Wunsch Recurrence

ABD.....IGOPRGADBZCEAGEHKKADAADR	A
DFEGVEACEHHJPALE...AEWQAAEGCLAKJEAGH	_
ABD.....IGOPRGADBZCEAGEHKKADAADR	F
DFEGVEACEHHJPALE...AEWQAAEGCLAKJEAGH	H
ABD.....IGOPRGADBZCEAGEHKKADAADR	_
DFEGVEACEHHJPALE...AEWQAAEGCLAKJEAGH	K

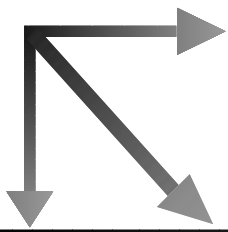
$$SCO[i, j] = \max \begin{cases} SCO[i, j - 1] + \sigma_I \\ SCO[i - 1, j - 1] + \sigma(S[i], T[j]) \\ SCO[i - 1, j] + \sigma_I \end{cases}$$



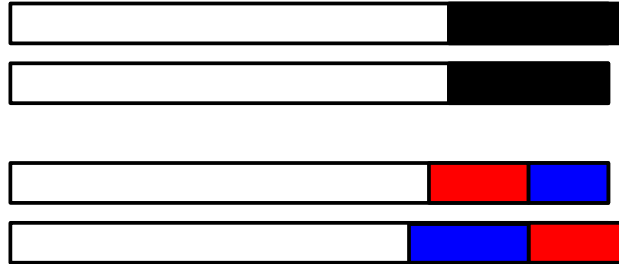
Non-overlapping Moves Recurrence



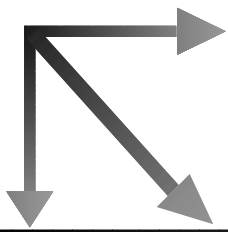
$$SCO[i, j] = \max \begin{cases} NO_MOVE \\ MOVE + \sigma_c \end{cases}$$



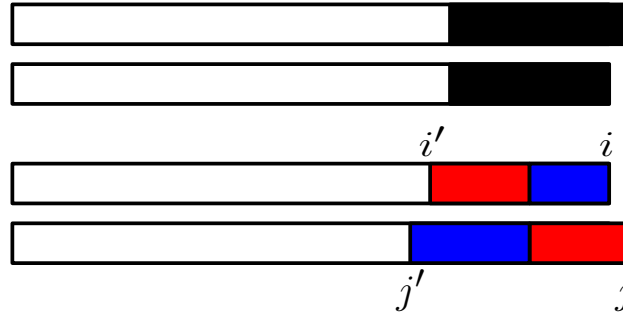
Non-overlapping Moves Recurrence



$$SCO[i, j] = \max \begin{cases} SCO[i, j - 1] + \sigma_I \\ SCO[i - 1, j - 1] + \sigma(S[i], T[j]) \\ SCO[i - 1, j] + \sigma_I \\ MOVE + \sigma_c \end{cases}$$

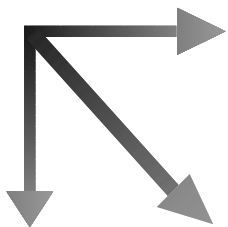


Non-overlapping Moves Recurrence

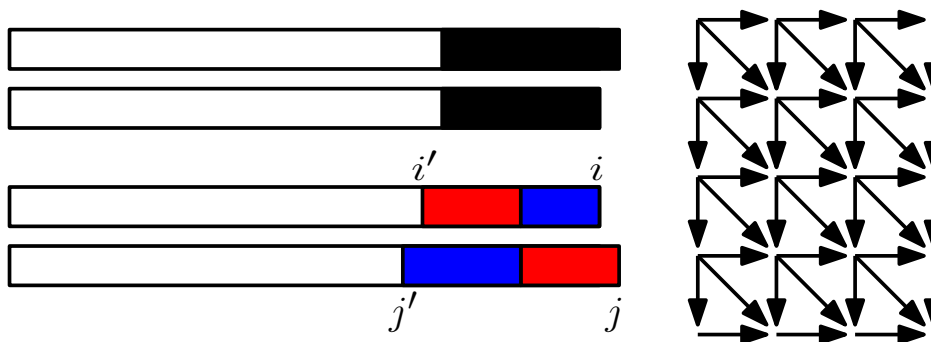


$$SCO[i, j] = \max \begin{cases} SCO[i, j - 1] + \sigma_I \\ SCO[i - 1, j - 1] + \sigma(S[i], T[j]) \\ SCO[i - 1, j] + \sigma_I \\ MOVE + \sigma_c \end{cases}$$

$$MOVE = \max_{0 \leq i' < i, 0 \leq j' < j} \{SCO[i', j'] + \delta_c(S[i'..i], T[j'..j])\}$$

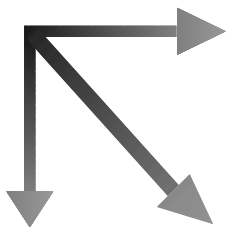


Non-overlapping Moves Recurrence

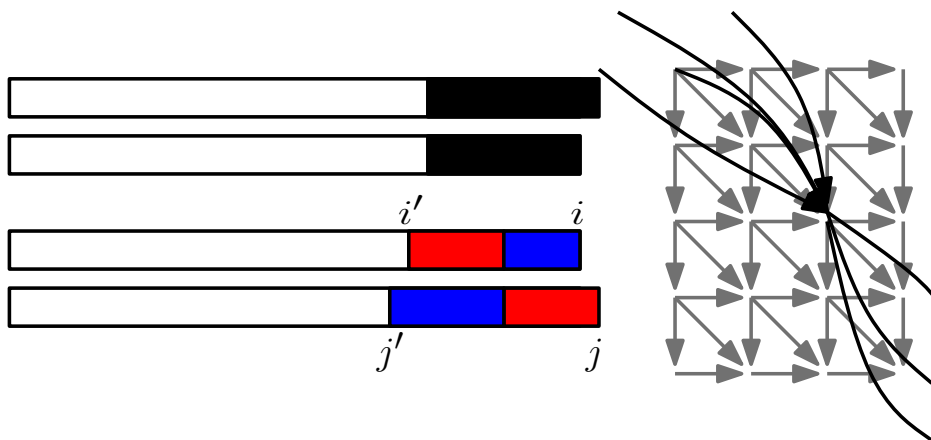


$$SCO[i, j] = \max \begin{cases} SCO[i, j - 1] + \sigma_I \\ SCO[i - 1, j - 1] + \sigma(S[i], T[j]) \\ SCO[i - 1, j] + \sigma_I \\ MOVE + \sigma_c \end{cases}$$

$$MOVE = \max_{0 \leq i' < i, 0 \leq j' < j} \{SCO[i', j'] + \delta_c(S[i'..i], T[j'..j])\}$$

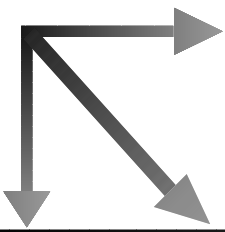


Non-overlapping Moves Recurrence



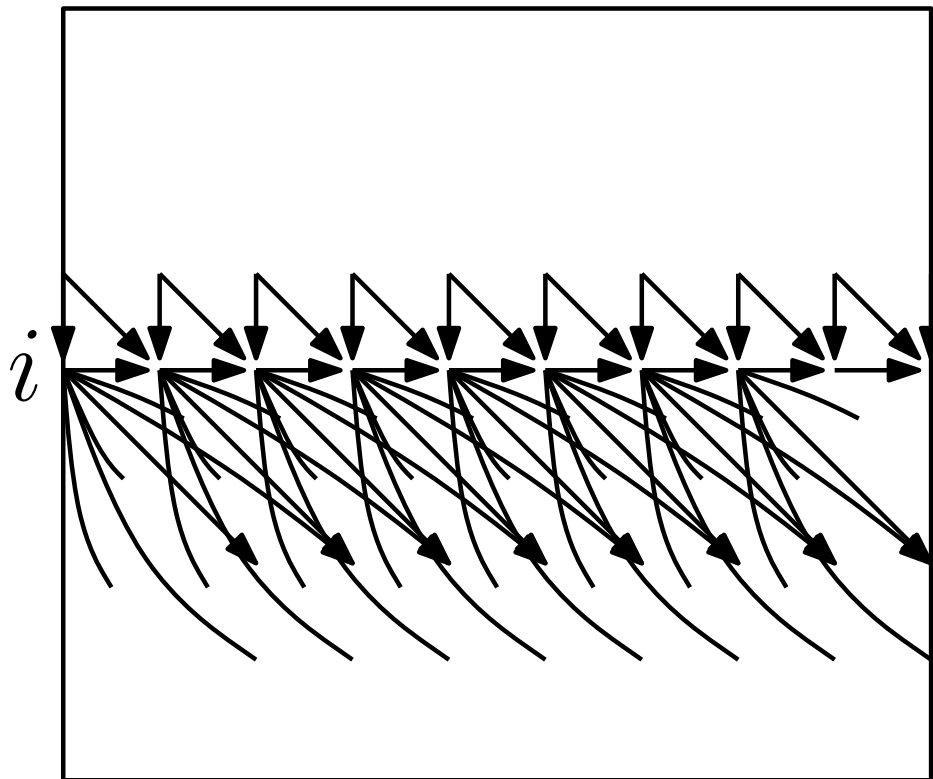
$$SCO[i, j] = \max \begin{cases} SCO[i, j - 1] + \sigma_I \\ SCO[i - 1, j - 1] + \sigma(S[i], T[j]) \\ SCO[i - 1, j] + \sigma_I \\ MOVE + \sigma_c \end{cases}$$

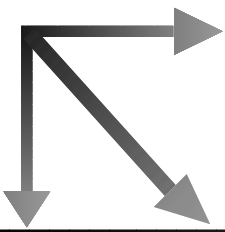
$$MOVE = \max_{0 \leq i' < i, 0 \leq j' < j} \{SCO[i', j'] + \delta_c(S[i'..i], T[j'..j])\}$$



Processing the Edges row-wise

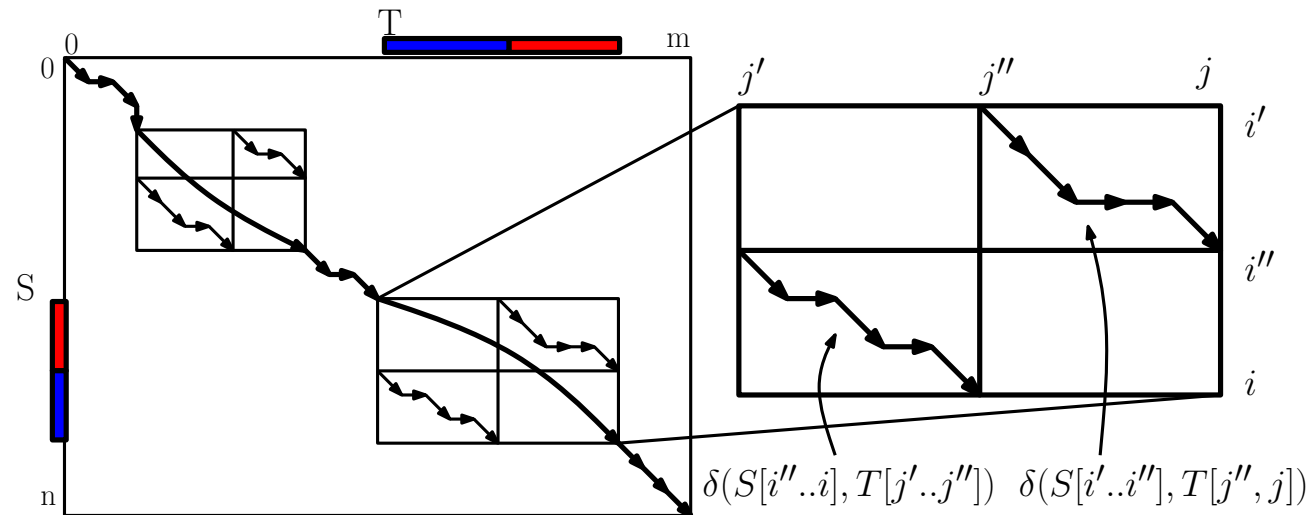
- Initialize with $-\infty$

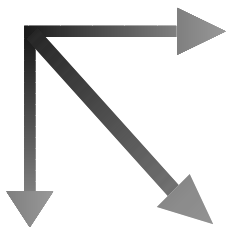




Computing Move

$$MOVE = \max_{0 \leq i' < i, 0 \leq j' < j} \{SCO[i', j'] + \delta_c(S[i'..i], T[j'..j])\}$$

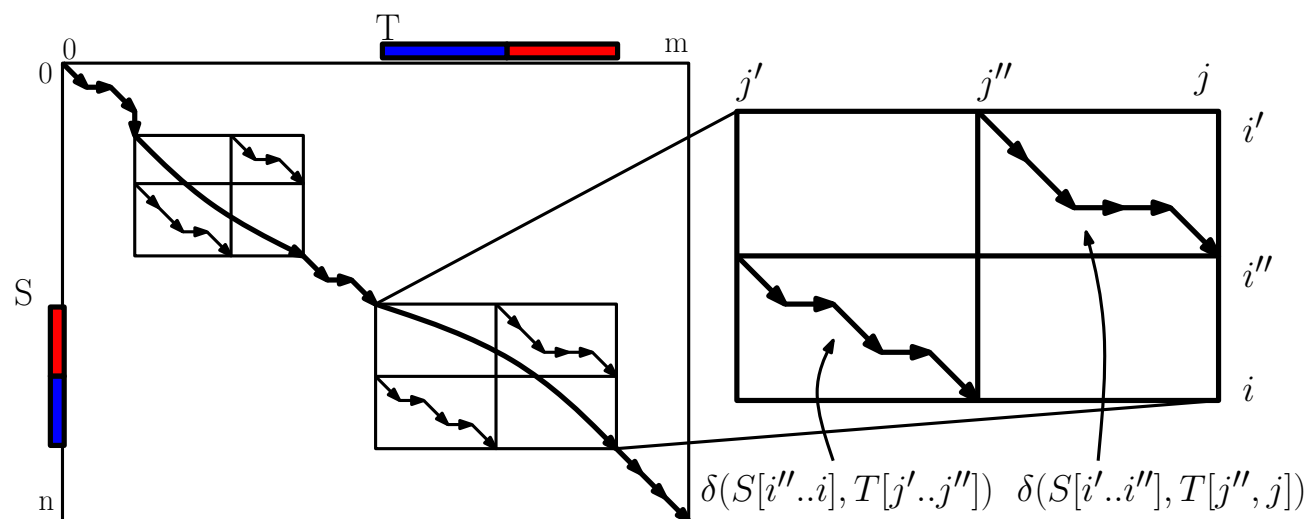


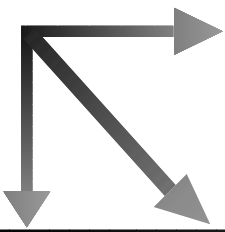


Computing Move

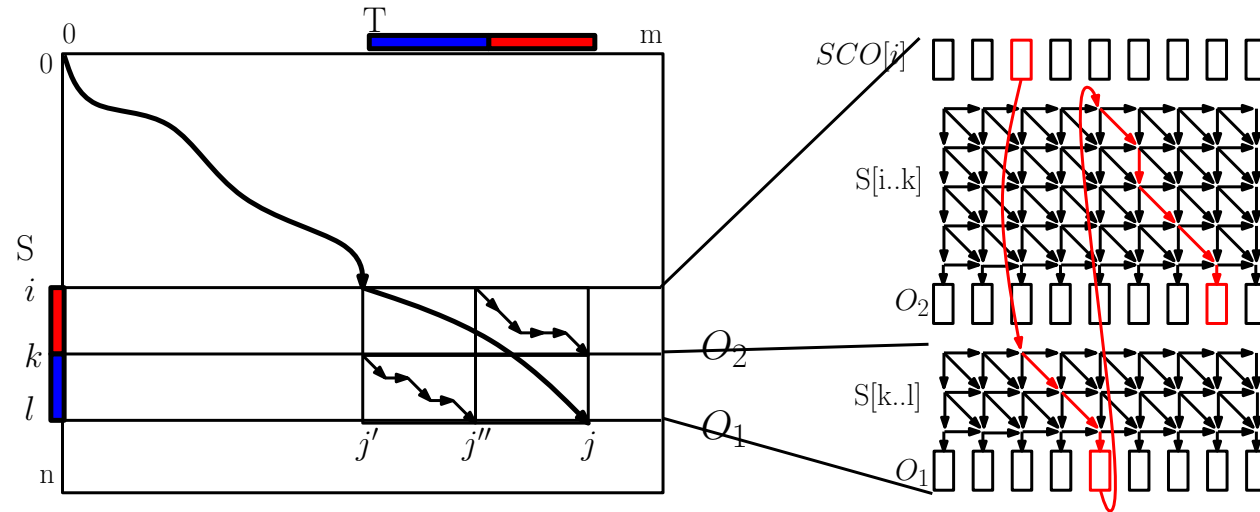
$MOVE =$

$$\max_{\substack{0 \leq i' < i'' < i \\ 0 \leq j' < j'' < j}} \{SCO_{S,T}[i', j'] + \delta(S[i''..i], T[j'..j'']) + \delta(S[i'..i''], T[j''..j])\}.$$



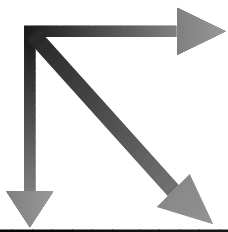


Computing Move



$$O_1[j''] = \max_{j'} \{SCO[i', j'] + \delta(S[i''..i], T[j'..j''])\}$$

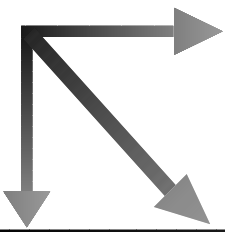
$$O_2[j] = \max_{j''} \{O_1[j''] + \delta(S[i'..i''], T[j''..j])\}$$



DIST Arrays

[Landau and Ziv-Ukelson, 2001, Schmidt, 1998]

Scoring Scheme	Compute O given $I, DIST$
General	$O(n^2)$
Constant Gap	$O(n \log n)$
Restricted	$O(n)$

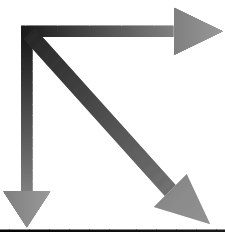


Scoring schemes

Scoring scheme	Indel	Move $\sigma_c(l)$	Scoring Matrix
General	Affine	Arbitrary	Arbitrary
Concave	Constant	Concave	Arbitrary
Restricted	Constant	Constant	Finite Precision

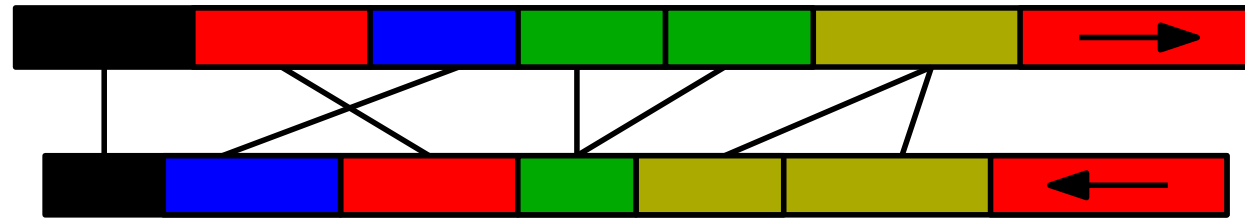
Non-overlapping moves can be solved in:

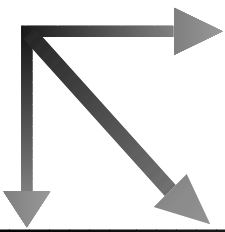
Scoring scheme	Time	Space
General	$O(n^5)$	$O(n^2)$
Concave	$O(n^4 \log n)$	$O(n^2)$
Restricted	$O(n^4)$	$O(n^2)$



Extension

All results can be extended to include inversions and tandem duplications in the *same time*.





Example

NEEDLEMAN_WUNSCH

Seq1: RPSTVPLPNTQALAMAGPKPKAHQFSIKSFPSPTQCSHCTSLMVGLIRQGYACEVCAFSCHVSCKDSAPQVCPPIPEQSKRPLGVQVQRGIG
Seq2: _____LSSADNDPEDSQHS__SLLSLTQ_____D

Seq1: TAYKGYVKVPKPTGVKK__GWQRAYAVVCDCKLFLY__DLPEGKSTQPGVIASQVLDLRDDEFVSSVLASDVIHATRRDIPCFRVT_____
Seq2: SVFEGWLSVFNKQNRRRGHGWKRQYVIVSSRKIIFYNSDIDKHNTTD____AVLIDL_SKVYHVRSVTQGDVIRADAKEIPRIFQLLYAGE

Seq1: _____TASLLGS
Seq2: GASHRPDEQSQLDVSVLHGNCNEERP GTIVHKGHEFVHI TYHMPTACEVCPKPLWHMFKPPAAYECKRCRNKIHKEHV D KHDPLAPCKLNHD

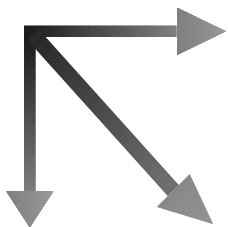
Seq1: PSKTSSLLILTENENEKRK WVGIL_____EGLQAILHKNRLRSQVVHVAQEAYDSSLPLI
Seq2: PRSARDMLLLAATPEDQSLWVARLLKRIQKSGYKAASYN NN_____STDGSKISPSQSTR

DP_MOVE

Seq1: RPSTVPLP_NTQ__A_LAMA_ [GTAYKGYVKVP_KPTGVK_KGWQRAYAVVCDCKLFLYDLPEGK_STQPGVIASQVLDLRDDEFVSSVLA
Seq2: LSSADNDPEDSQHSLLSLTQ [DSVFEGWLSVFNKQNRRRGHGWKRQYVIVSSRKIIFYNSDIDKHNTTD AVL____IDL_SKVYHVRSVTQ

Seq1: SDVIHATRRDIPCFRVT_ASLLG_S__PSKTSSL_L_ILTENENEKRK | GP_KPKAHQF_SIKSFPSPTQCSHCTSLMVGLIR__QGYACE
Seq2: GDVIRADAKEIPRIFQLLYAGE_GASHRPDEQSQLDVSVLHGNCNEERP | GTIVHKGHEFVHI_TYHMPTACEVCPKPLWHMFKPPAAYECK

Seq1: VCAFSCHVSCKDS_APQV_CPIPE_QSKRP____LGVDVQ_RGI] WVGILEGLQAILHKNRLRSQVV_HVAQEAYD_S_SLPLI
Seq2: RCRN KIHKEHV D KHDPLAPCKLNHDPRSARDMLLLAATPEDQSL] WVARL__LKRI_QKSGYKAASYN NNSTDGSKISPSQSTR



Example

NEEDLEMAN_WUNSCH

Seq1: RPSTVPLPNTQALAMAGPKPKAHQFSIKSFPSPTQCSHCTSLMVGLIRQGYACEVCAF SCHV SCKDSAPQV C PIPPEQSKRPLGVDVQRGIG
Seq2: _____LSSADNDPEDSQHS__SLLSLTQ_____D

Seq1: TAYKGYVKVPKPTGVKK__GWQRAYAVVCDCKLFLY__DLPEGKSTQPGVIASQVLDLRDDEFVSSVLASDVIHATRRDIPCFRV_____
Seq2: SVFEGWLSVPNKQNRRRGHGWKRQYVIVSSRKIIFYNSDIDKHNTTD____AVLIDL_SKVYHVRSVTQGDVIRADAKEIPRIFQLLYAGE

Seq1: _____TASLLGS
Seq2: GASHRPDEQSQLDVSVLHGNCNEERP GTIVHKGHEFVHI_TYHMPTACEVCPKPLWHMFKPPAAYECKRCRNKIHKEHVVDKHDPLAPCKLNHD

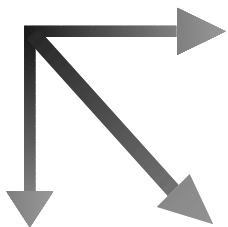
Seq1: PSKTSSLLILTENENEKRKVGIL_____EGLQAILHKNRLRSQVVHVAQEAYDSSLPLI
Seq2: PRSARDMLLLAATPEDQSLWVARLLKRIQKSGYKAASYNNTD_____STDGSKISPSQSTR

DP_MOVE

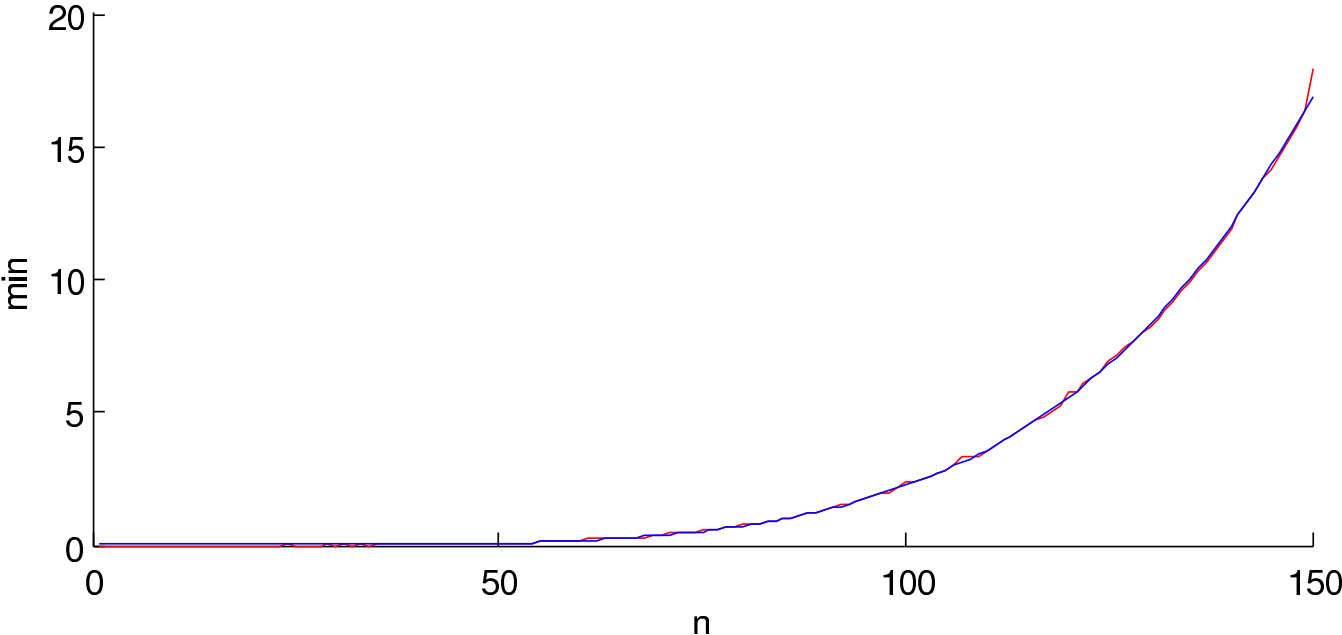
Seq1: RPSTVPLP_NTQ__A_LAMA_ [GTAYKGYVKVP_KPTGVK_KGWQRAYAVVCDCKLFLYDLPEGK_STQPGVIASQVLDLRDDEFVSSVLA
Seq2: LSSADNDPEDSQHSLLSLTQ [DSVFEGWLSVPNKQNRRRGHGWKRQYVIVSSRKIIFYNSDIDKHNTTD AVL____IDL_SKVYHVRSVTQ

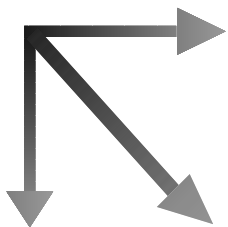
Seq1: SDVIHATRRDIPCFRVT_ASLLG_S__PSKTSSL_L_ILTENENEKRK | GP_KPKAHQF_SIKSFPSPTQCSHCTSLMVGLIR__QGYACE
Seq2: GDVIRADAKEIPRIFQLLYAGE_GASHRPDEQSQLDVSVLHGNCNEERP | GTIVHKGHEFVHI_TYHMPTACEVCPKPLWHMFKPPAAYECK

Seq1: VCAF SCHV SCKDS_APQV_C PIPPE_QSKRP____LGVDVQ_RGI] WVGILEGLQAILHKNRLRSQVV_HVAQEAYD_S_SLPLI
Seq2: RCRNKIHKEHVVDKHDPLAPCKLNHDPRSARDMLLLAATPEDQSL] WVARL__LKRI_QKSGYKAASYNNTDSTDGSKISPSQSTR

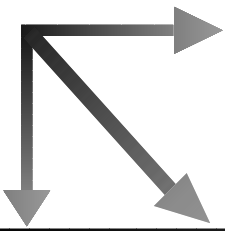


Example Runtime





END



Announcement

Will complete Bachelor in September

Project / Internship from September - February?

References

- [Cormode and Muthukrishnan, 2002] Cormode, G. and Muthukrishnan, S. (2002). The string edit distance matching problem with moves. In *SODA '02: Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 667–676, Philadelphia, PA, USA. Society for Industrial and Applied Mathematics.
- [Fliess et al., 2002] Fliess, A., Motro, B., and Unger, R. (2002). Swaps in protein sequences. *Proteins.*, 48(2):377–387.
- [Landau and Ziv-Ukelson, 2001] Landau, G. M. and Ziv-Ukelson, M. (2001). On the common substring alignment problem. *J. Algorithms*, 41(2):338–354.
- [Schmidt, 1998] Schmidt, J. P. (1998). 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.
- [Schoeninger and Waterman, 1992] Schoeninger, M. and Waterman, M. S. (1992). A local algorithm for dna sequence alignment with inversions. *Bull Math Biol*, 54(4):521–536.
- [Shapira and Storer, 2002] Shapira, D. and Storer, J. A. (2002). Edit distance with move operations. In *CPM '02:*

Proceedings of the 13th Annual Symposium on Combinatorial Pattern Matching, pages 85–98, London, UK. Springer-Verlag.

[Vellozo et al., 2006] Vellozo, A. F., Alves, C. E. R., and do Lago, A. P. (2006). Alignment with non-overlapping inversions in $o(n^3)$ -time. In *WABI*, volume 4175, pages 186–196. LNCS Springer-Verlag.