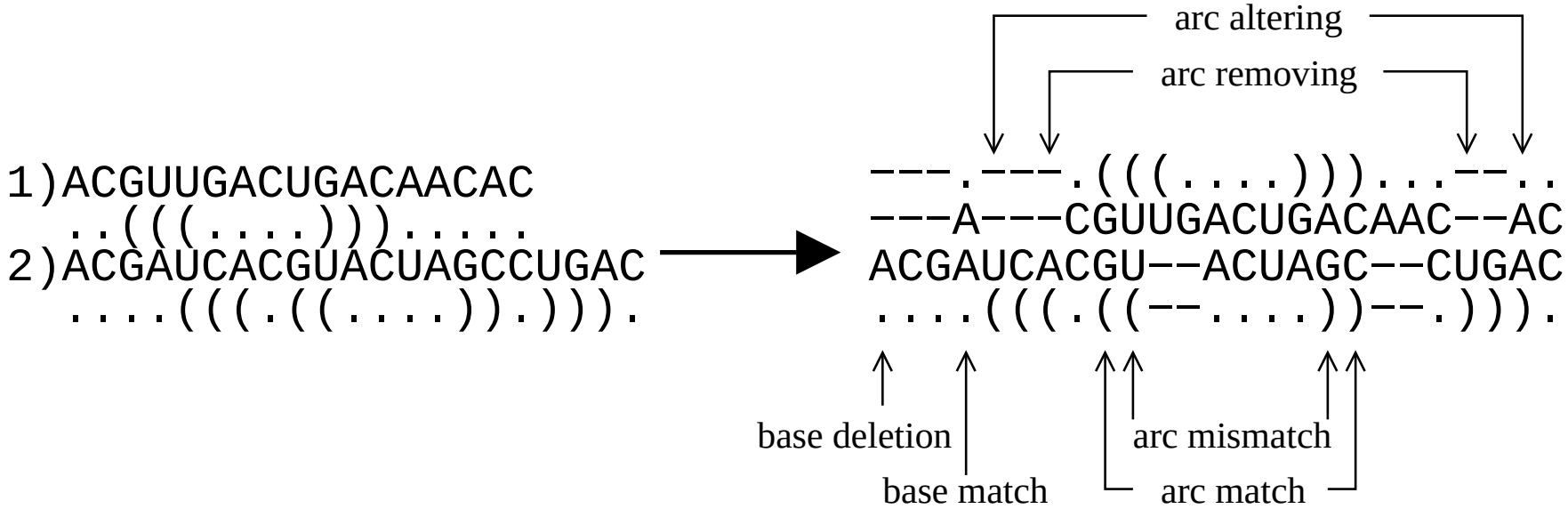
Pairwise RNA Edit Distance

- In the following:
  - Sequences S_1 and S_2
  - associated structures P_1 and P_2

- scoring of alignment: different edit operations

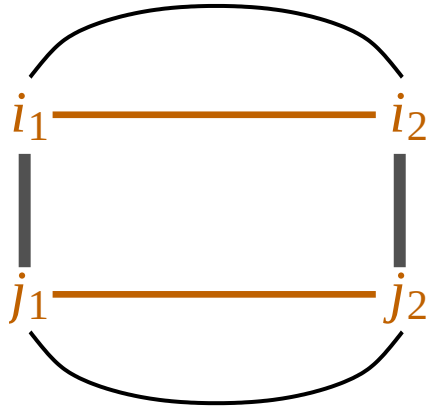


- **Notation:**
  - $S_k[i]$: position i in sequence k (for $k = 1, 2$).
  - $S_k[i]$ is *free* if there is no arc incident in P_k to i

Jiang et al., 2002:
- above scoring scheme
- complexity of different problem classes
- algorithms

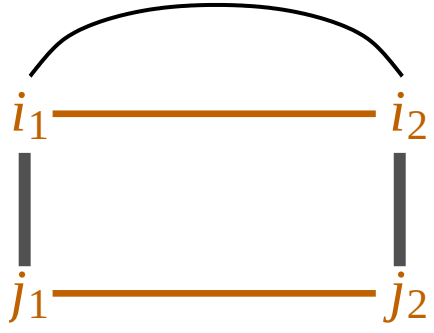
Edit Distance – Scores

- base scoring: *base mismatch* w_m, *base indel* w_d.
- **case 1:** *arc match* and *arc mismatch*



  - *arc match* (cost 0): $S_1[i_1] = S_2[j_1]$ and $S_1[i_2] = S_2[j_2]$
  - *arc mismatch*: $S_1[i_1] \neq S_2[j_1]$ or $S_1[i_2] \neq S_2[j_2]$
  - cost for mismatch:
    - if both ends differ: w_{am}
    - if only one differs: $\frac{w_{am}}{2}$

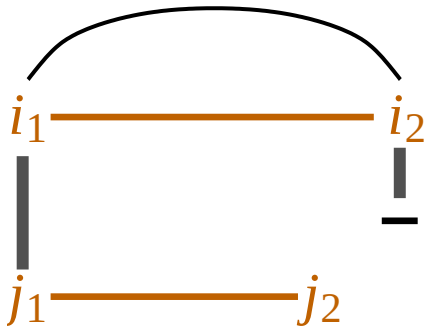
- in the following: different ways of deleting arcs
  cost: cost for deleting arc + cost for base operations
- **case 2:** *arc breaking*



  - (i_1, i_2) in P_1, but (j_1, j_2) is **not in** P_2
  - cost: w_b + possibly $2 \cdot w_m$.

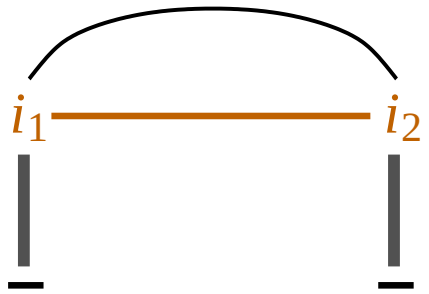
Edit Distance – Scores (Cont.)

- **case 3:** *arc altering*



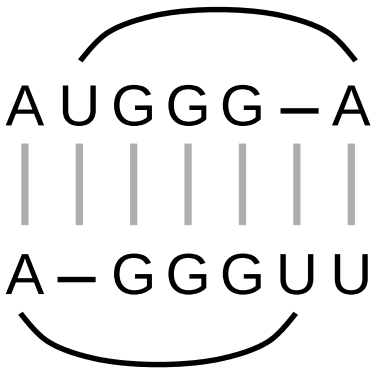
  - cost: w_a + possibly w_m.

- **case 4:** *arc removing*



  - cost: w_r

- **remark:** arc breaking/altering/removal can overlap



Edit Distance – Scores Summary

- operations on single bases:
  - base insertion/deletion (w_d)
  - base mismatch (w_m)
- operations that act on both ends of an arc:
  1. arc mismatch (w_{am})
  2. arc breaking (w_b)
  3. arc altering (w_a)
  4. arc removing (w_r)

Example:

```
1234567890123456
(..)((.(.)))(..)
CCGGAGGCCGCUCCCG
CCG-ACCC-CGU-CC-
(.).((....)).....
```

Plan

1. Jiang algorithm solves the edit problem given the following restrictions:
   - non-crossing (aka nested aka pseudoknot-free) input structures[1]
   - pairwise alignment only
   - scoring restricted by $w_a = \frac{w_r + w_b}{2}$.

2. show MAX-SNP-hardness without the restriction $w_a = \frac{w_r + w_b}{2}$.

[1] actually, we will see that crossing in at most one structure is OK

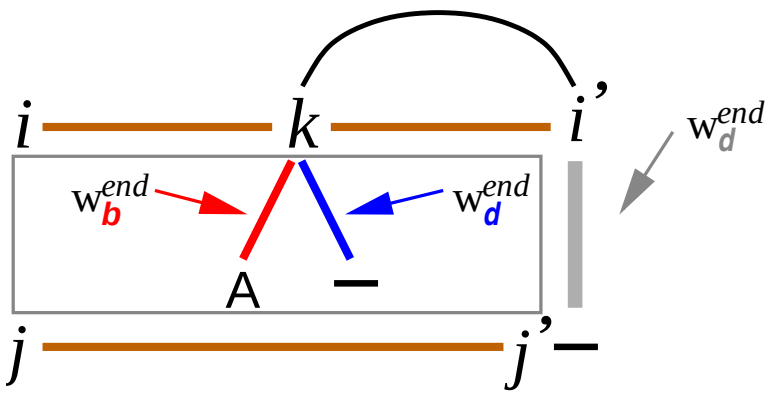
Restriction $w_a = \frac{w_r + w_b}{2}$

- Arc altering is at one end like arc removing and at the other end arc breaking
- Restriction $w_a = \frac{w_r + w_b}{2}$ captures that
  $\Rightarrow$ left and right ends of arcs can be scored independently if they are broken, deleted or altered.
  $\Rightarrow$ cost for arc end *deletion* w_d^{end} and *breaking* w_b^{end} instead of w_r,w_b, and w_a:

$$w_b = 2 \cdot w_b^{end}$$

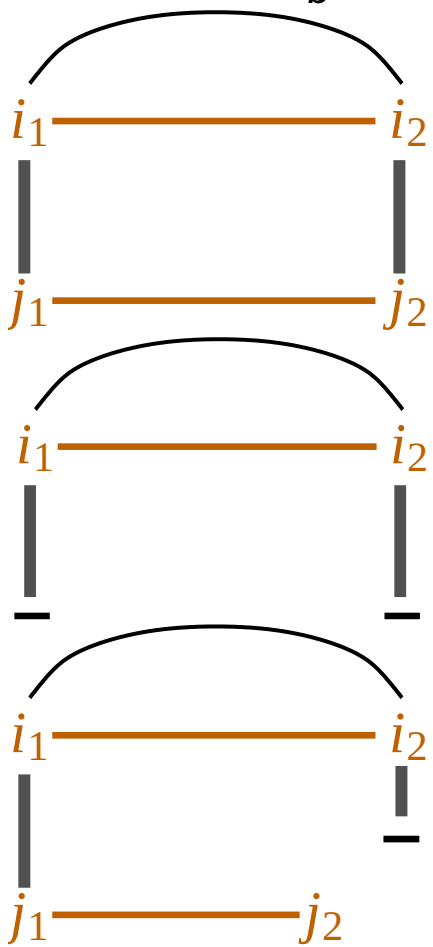
$$w_r = 2 \cdot w_d^{end}$$

$$w_a = \frac{w_r + w_b}{2} = w_b^{end} + w_d^{end}$$



Independent Arc Scoring

- cost for arc end *deletion* w_d^{end} and *breaking* w_b^{end} Hence: Cost
- arc breaking: $w_b = 2 \cdot w_b^{end}$
- arc removing: $w_r = 2 \cdot w_d^{end}$
- arc altering: $w_a = w_b^{end} + w_d^{end}$



of breaking or removing one end of the arc is independent of whether the other end is broken/removed or not. Only the cost of matching one end of an arc is dependent on whether the other end is matched, too.

Example

- cost for arc end *deletion* w_d^{end} and *breaking* w_b^{end}
- arc breaking: $w_b = 2 \cdot w_b^{end}$
- arc removing: $w_r = 2 \cdot w_d^{end}$
- arc altering: $w_a = w_b^{end} + w_d^{end}$

```
1234567890123456
(..)((.(.)))(..)
CCGGAGGCCGCUCCCG
CCG-ACCC-CGU-CC-
(.).((....)).....
```

How to make a DP algorithm for alignment?

dynamic programming $\Rightarrow$ compute optimal alignment recursively from optimal alignments of "fragments"

questions to answer:

- what kind of "fragments" do we consider?
  ($\Rightarrow$ semantics of a matrix entry)
- how to compute the solutions for all these fragments?
  ($\Rightarrow$ recursion equation)
- complexity
- details (evaluation order, implementation details,...)

Semantics of DP entry $D(i, i', j, j')$

$D(i, i', j, j')$ is the minimum cost of aligning the fragment $[i, i']$ of the first sequence to the fragment $[j, j']$ of the second sequence given that no arcs are matched that have one end inside these fragments and one end outside.

Remarks

- The additional restriction makes the alignment of the fragments independent of the alignment of the remaining parts.
- We will see later, why it is not sufficient to look at (alignments of) prefixes, as done for plain sequence alignment.

Recursion for $D(i, i', j, j')$

$$D(i,i',j,j') =$$

$$\min \begin{cases} D(i,i'-1,j,j') + w_d + \psi_1(i')(w_d^{end} - w_d) \\ D(i,i',j,j'-1) + w_d + \psi_2(j')(w_d^{end} - w_d) \\ D(i,i'-1,j,j'-1) + \chi(i',j')w_m + (\psi_1(i') + \psi_2(j'))w_b^{end} \\ \text{if } \exists (a_1,a_2) = ((i_1,i'),(j_1,j')) \in P_1 \times P_2 \text{ for some } i_1, j_1 \\ \quad D(i,i_1-1,j,j_1-1) + D(i_1+1,i'-1,j_1+1,j'-1) \\ \quad +(\chi(i_1,j_1) + \chi(i',j'))\frac{w_{am}}{2} \end{cases}$$

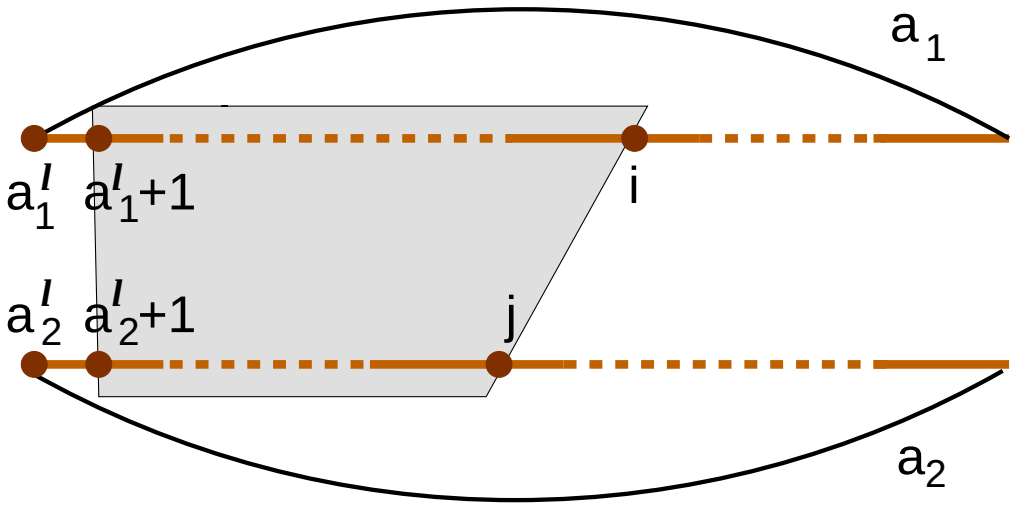
Notation

- $\psi_1(i) = 1$ if i is paired in structure 1, 0 otherwise. ($\psi_2(i)$ analogous)
- $\chi(i,j) = 1$ if $S_1[i] \neq S_2[j]$, 0 otherwise.

An optimized version: Jiang Algorithm

- $D(i, i', j, j')$ alignment of subsequences
- in principle: all regions $[i..i']$ and $[j..j']$.
  $\Rightarrow O(n^2 m^2)$ space

- **But:** not all entries are considered

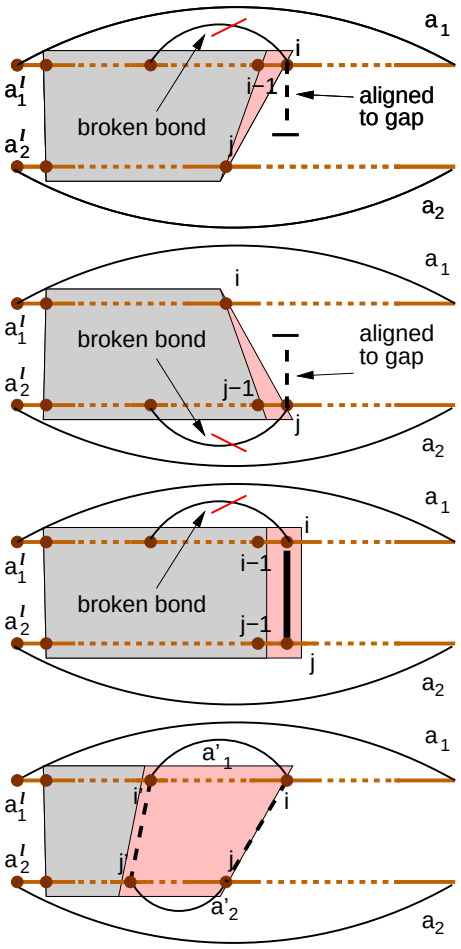


- **Hence:** $O(nm)$-matrices $M_{a_2}^{a_1}$ for each pair of arcs a_1, a_2.
  Each matrix: $O(nm)$ entries $M_{a_2}^{a_1}(i, j)$

Jiang Recursion

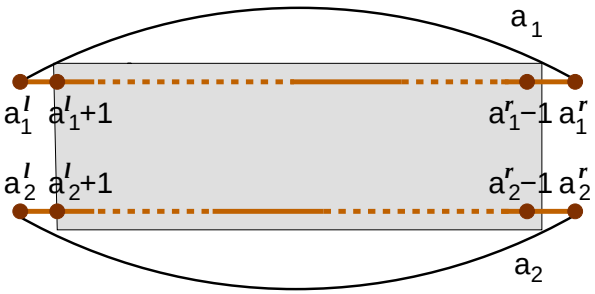
- reformulated recursion:

$$M_{a_2}^{a_1}(i,j) = \min \begin{cases} M_{a_2}^{a_1}(i-1,j) + w_d \\ \quad +\psi_1(i)(w_d^{end} - w_d) \\ M_{a_2}^{a_1}(i,j-1) + w_d \\ \quad +\psi_2(j)(w_d^{end} - w_d) \\ M_{a_2}^{a_1}(i-1,j-1) + \chi(i,j) w_m \\ \quad +(\psi_1(i) + \psi_2(j)) w_b^{end} \\ M_{a_2}^{a_1}(i'-1,j'-1) \\ \quad +M_{a_2'}^{a_1'}(i-1,j-1) \\ \quad +(\chi(i',j') + \chi(i,j)) \frac{w_{am}}{2} \end{cases}$$



Complexity

- time complexity:
  $O(nm)$ arc pairs $\times$ $O(nm)$ alignment below arcs $=$ $O(n^2m^2)$ time
- remaining question: space complexity:
- each entry of some $M_{a_2}^{a_1}$ only depends on
  - other entries of the same matrix $M_{a_2}^{a_1}$
  - and final entries of arc pairs of smaller arcs:



  $\Rightarrow$ store final values in separate $O(nm)$ matrix F
  (in recursion, replace lookup $M_{a_2'}^{a_1'}(i-1,j-1)$ by $F(a_1', a_2')$)
- $\Rightarrow$ it suffices to keep only F and one $M_{a_2}^{a_1}$ in memory simultaneously.

- compute all $M_{a_2}^{a_1}$ ordered (increasing) according to size of a_1 and a_2

Complexity

- Matrix F: $O(nm)$ space
- only one Matrix $M_{a_2}^{a_1}$ at a time: $O(nm)$ space
  argument: for computing one entry $M_{a_2}^{a_1}(i,j)$,
  recurse only to $F(a_1', a_2')$ for "smaller" a_1', a_2' or entries of the same matrix $M_{a_2}^{a_1}$
  consequence: reuse space for $M_{a_2}^{a_1}$
- **TOTAL:** $O(nm + nm) = O(nm)$ space

drawback: traceback requires recomputation
but only $O(\min(n,m))$ many matrices $M_{a_2}^{a_1}$ need to be recomputed.

What about Pseudoknots?

- Why doesn't the algorithm work for pseudoknots?
  $\Rightarrow$ last recursion case does not cover cases where matched arcs cross (compare Nussinov)

- only matching of crossing arcs is a problem
  $\Rightarrow$ pseudoknots in only one of the structures are OK.

The alignment hierarchy

- Alignment approaches have different limitations concerning
  - the two input structures
  - the common superstructure (e.g. for tree alignment $\Rightarrow$ nested)
  - the set of edit operations
- alignment hierarchy classifies alignment problems as
  input1$\times$ input2$\rightarrow$ superstructure
  with input1,input2,superstructure being one of
  - PLAIN: only plain sequence (no basepairs at all)
  - NEST: only nested structures (no pseudoknots)
  - CROSS: crossing structures (pseudoknots)
  - UNLIM: unlimited, also several base pairs per base possible.
- Examples:
  - CROSS$\times$NEST$\rightarrow$UNLIM: Jiang algorithm
  - NEST$\times$NEST$\rightarrow$NEST: tree alignment

The alignment hierarchy

- besides the limitations of input and superstructure, the scoring scheme (set of edit operations) is an important difference between the various alignment problems / algorithms.
- Overview: alignment hierarchy (Blin&Touzet, SPIRE 2006)

structures	scoring schemes		
	no altering+removing	no arc altering	all operations
NEST×NEST→NEST	$O(n^4)$	$O(n^4)$	$O(n^4)$
NEST×NEST→CROSS	$O(n^3 \log(n))$	NP-complete	
NEST×NEST→UNLIM	$O(n^3 \log(n))$	NP-complete	NP-complete
CROSS×NEST→CROSS	$O(n^3 \log(n))$	NP-complete	
CROSS×NEST→UNLIM	$O(n^3 \log(n))$	NP-complete	Max SNP-hard
CROSS×CROSS→CROSS	NP-complete	NP-complete	
CROSS×CROSS→UNLIM	NP-complete	NP-complete	Max SNP-hard
UNLIM×NEST→UNLIM	$O(n^3 \log(n))$	NP-complete	Max SNP-hard
UNLIM×CROSS→UNLIM	NP-complete	NP-complete	Max SNP-hard
UNLIM×UNLIM→UNLIM	NP-complete	NP-complete	Max SNP-hard

- $O(n^3 log(n))$: P.Klein, ESA 1998
  $O(n^3)$: E.Demaine *et al.*, ICALP 2007