13 Comparative RNA analysis

Sources for this lecture:

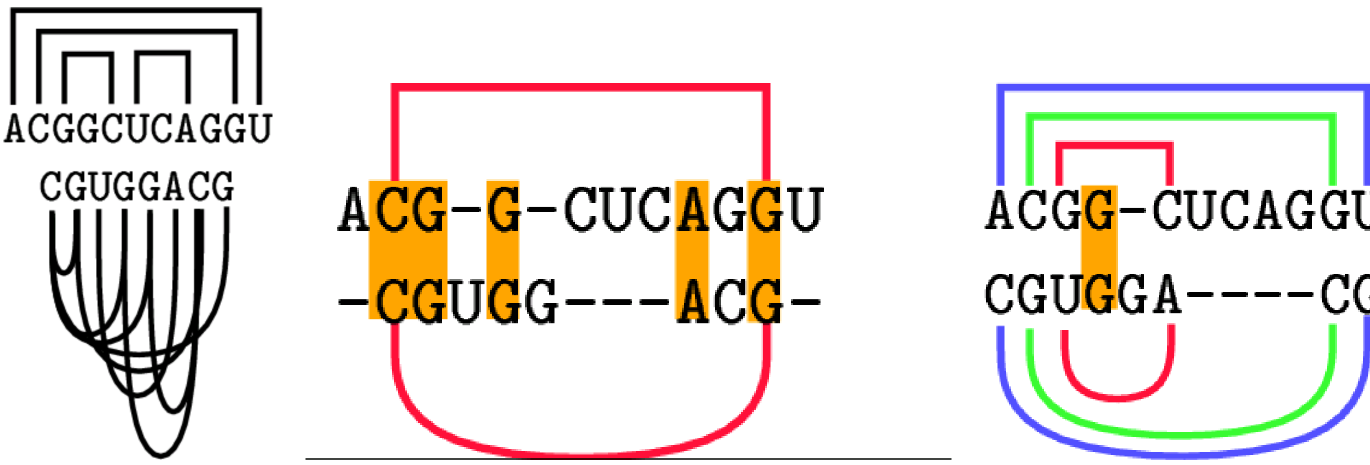
- R. Durbin, S. Eddy, A. Krogh und G. Mitchison, Biological sequence analysis, Cambridge, 1998
- D.W. Mount. Bioinformatics: Sequences and Genome analysis, 2001.
- V. Bafna, S. Muthukrishnan, R. Ravi, Computing similarity between RNA strings.
- D. Sankoff, Simultaneous solution of the RNA Folding , Alignment and Protosequence Problems, SIAM Journal of Appl. Math., 45,5,1985
- J. Gorodkin, L.J. Heyer, S. Brunak, G.D. Stormo, Displaying the information contents of structural RNA alignments: the structure logos.

The algorithms of Nussinov and Zuker try to compute the best RNA folding by optimizing certain objective functions like the Gibbs free energy or the number of base pairs to compute the best fold.

Another successful approach is based on the comparison of sequence *structure* of RNA. While this is different to (primary) sequence analysis, the main tasks are the same.

In the below figure you see on the right two RNA alignments. One has a rather high sequence conservation but does not preserve the secondary structure, while the second preserves the structure and not sequence.

The problem in RNA comparison is to capture the structure conservation and possibly also the sequence conservation.



interactions

alignment with high sequence score

alignment with high interaction score

13.1 RNA folding via comparative analysis

Although energy minimization techniques are attractive, almost all trusted RNA secondary structures to date were determined using comparative analysis. However, comparative methods require many diverse sequences and highly accurate multiple alignments to work well.

The key idea is to identify the interactions (that is the Watson-Crick correlated positions) in a multiple alignment, e.g.:

seq1	G**C**CUUCGG**G**C
seq2	G**A**CUUCGG**U**C
seq3	G**G**CUUCGG**C**C

Assume for now that we are given a multiple alignment. The amount of correlation of two positions can then be computed as the *mutual information* content measure: *if you tell me the identity of position i, how much do I learn about the identity of position j?*

13.2 Mutual information content

A method used to locate covariant positions in a multiple sequence alignment is the mutual information content of two columns.

First, for each column i of the alignment, the frequency $f_i(x)$ of each base $x \in \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{U}\}$ is calculated.

Second, the 16 joint frequencies $f_{ij}(x, y)$ of two nucleotides, x in column i and y in column j, are calculated.

If the base frequencies of any two columns i and j are *independent* of each other, then the ratio of $\frac{f_{ij}(x,y)}{f_i(x) \times f_j(y)} \approx 1$.

If these frequencies are *correlated*, then this ratio will be greater than 1.

To calculate the *mutual information content* $H(i, j)$ in bits between the two columns i and j, the logarithm of this ratio is calculated and summed over all possible 16 base-pair combinations:

$$H_{ij} = \sum_{xy} f_{ij}(x, y) \log_2 \frac{f_{ij}(x, y)}{f_i(x) f_j(y)}.$$

This measure is maximum at 2 bits, representing perfect correlation.

If either site is conserved, there is less mutual information: for example, if all bases at site i are A, then the mutual information is 0, even if site j is always U, because there is no covariance.

The main problem with the comparative approach is that we need an accurate multiple alignment to get good structures and we need accurate structures to get a good alignment!

Examples of how to compute the mutual information content:

1	2	3	4	5	6	7	8
C	G	C	G	A	U	A	A
C	G	G	C	C	G	C	C
C	G	C	G	G	C	G	G
C	G	G	C	U	A	U	U

Compute:
H_{12} = ________
H_{34} = ________
H_{56} = ________
H_{78} = ________

The above example illustrates, that the mutual information a given in the above formula is a general concept that not only captures the correlation between Watson crick pairs (columns 5 and 6), but also others (columns 7 and 8).

If we are only interested in the correlation of Watson crick pairs, Gorodkin suggested a slight modification to the above measure. In addition he proposed a method to display this information as a sequence-structure logo.

Recall, that for primary sequences over the alphabet $A = \{A, C, G, U, -\}$ the information content of position i of an alignment is:

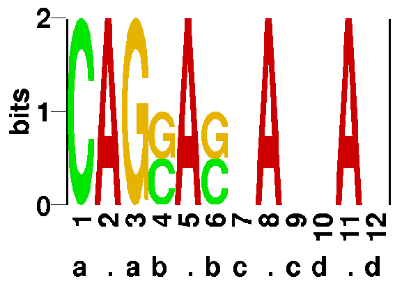
$$I_i = \sum_{k \in A} I_{ik} = \sum_{k \in A} q_{ik} \log_2 \frac{q_{ik}}{p_k}$$

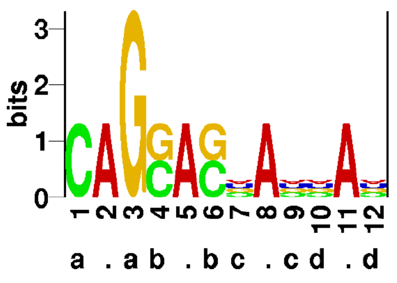
where q_{ik} is the fraction of 'base' k at position i. For $k \neq' -'$ we interpret p_k as the *a priori* distribution of the bases for that genome. (This is an extension of the Schneider logos proposed by Hertz and Stormo).

Having determined I_i for a position, there are two common ways to display the height d_{ik} for each letter:

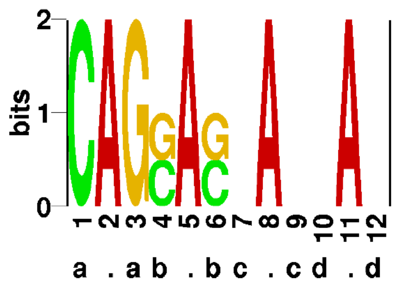
1. $d_{ik} = q_{ik} \cdot I_i$ (type 1 logo), the height is proportional to the frequency.
2. $d_{ik} = \frac{q_{ik}/p_k}{\sum_l q_{il}/p_l} \cdot I_i$ (type 2 logo), the height is in proportion to the their frequencies relative to the expected frequencies.

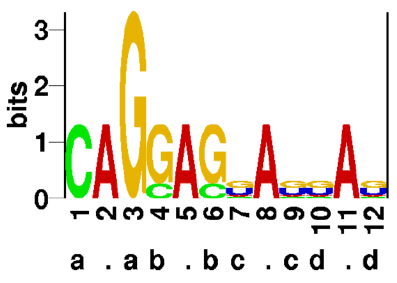
$(0.25, 0.25, 0.25, 0.25)$ $(0.4, 0.4, 0.1, 0.1)$





t1





t2

Gorodkin proposed now the following: Define $\tilde{q}_{ij}$ to be the fraction of sequences that have complementary base pairs at positions i and j. Then the expected value of $\tilde{q}_{ij}$ is $E(\tilde{q}_{ij}) = \sum_{(k,l) \in B} C_{kl} \cdot q_{ik} \cdot q_{jl}$, where $B = (A \setminus \{-\})^2$, and C_{kl} is a symmetrical matrix with $C_{kl} = 1$ if the base pairs k and l are complementary and 0 otherwise.

Then he defines the mutual information as the log likelihood ratio of the observed to expected frequency of complementary bases (Kullback-Leibler distance between two distributions), namely:

$$M_{ij} = \tilde{q}_{ij} \log_2 \frac{\tilde{q}_{ij}}{E(\tilde{q}_{ij})} + (1 - \tilde{q}_{ij}) \log_2 \frac{1 - \tilde{q}_{ij}}{1 - E(\tilde{q}_{ij})}.$$

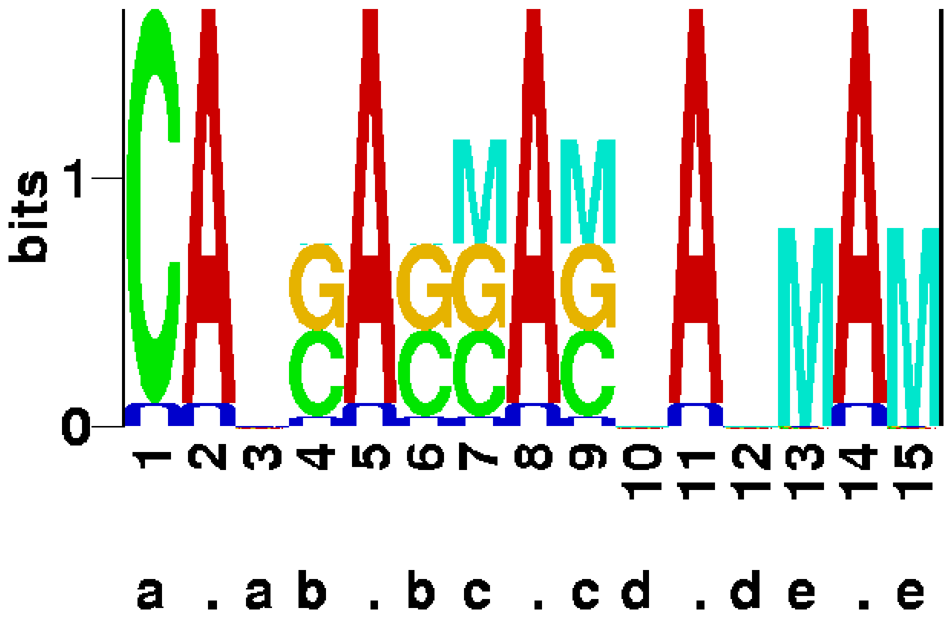
The mutual information between two positions is symmetrical. Hence we define $M_i = M_j = M_{ij}/2$ and display the amount in the sequence logo as a 'M'.

Consider the following alignment:

```
1   2   3   4   5   6   7   8   9   10   11   12   13   14   15

C   A   A   C   A   G   C   A   G   A    A    G    A    A    U
C   A   C   G   A   C   G   A   C   C    A    A    C    A    G
C   A   G   C   A   C   C   A   G   G    A    C    G    A    C
C   A   U   G   A   G   G   A   C   U    A    U    U    A    A
```

This results in the following sequence logo (ignore the 'U's they are necessary because of a bug in the program):

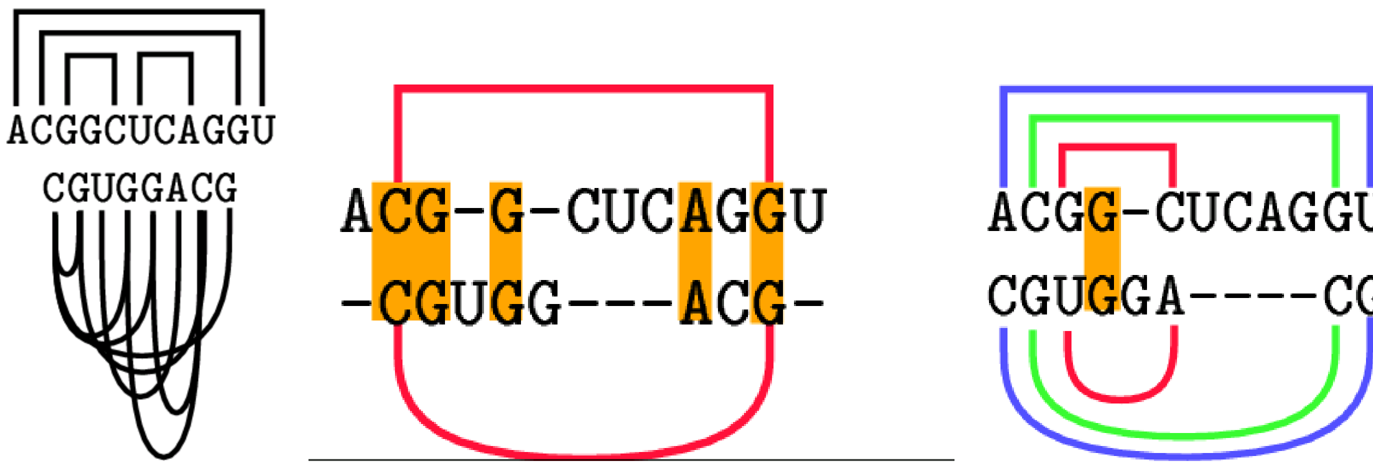


13.3 RNA sequence structure alignment

Denote with $\hat{\Sigma}$ the alphabet extended with a gap character.

Definition 1. Let S be a sequence over $\hat{\Sigma}$ with length m. A pair (i, j) with $1 \leq i < j \leq m$ is called *interaction* if $s_i \neq$ '-' and $s_j \neq$ '-'. A set P of interactions is called *annotation* of S. Two interactions (i, j), (k, l) in an annotation are in *conflict* if $i = k$ or $i = l$ or $j = k$ or $j = l$. A *(secondary) structure* is an annotation with no two interactions in conflict. If S is a sequence and P an annotation we call the pair (S, P) an *annotated sequence*. If P is a (secondary) structure we call it a *structured sequence*.

According to the definition the below figures shows one structured sequence and one annotated sequence together with two possible structural alignments



Definition 2. Let Σ be a finite alphabet without the blank character '-' and let $\hat{\Sigma} = \Sigma \cup \{\text{'-'}\}$. If $(S_1, P_1), \dots, (S_k, P_k)$ are k annotated sequences $\in \Sigma^*$ with lengths $n_1, \dots, n_k$ then a *multiple structural alignment* A of $(S_1, P_1), \dots, (S_k, P_k)$ is a $k \times n$–dimensional matrix consisting of k structured sequences $(\hat{S}_1, \hat{P}_1), \dots, (\hat{S}_k, \hat{P}_k)$ with the following properties:
1) $a_{i,j} \in \hat{\Sigma} \quad \forall\, 1 \le i \le k,\ 1 \le j \le n$ and sequence $\hat{S}_i$ gives sequence S_i if the blanks are removed.

2) There is no column consisting only of blank characters implying $\max\{n_1, \dots, n_k\} \le n \le \sum_{i=1}^{k} n_i$.

3) $\forall (l, m) \in \hat{P}_i$ the following holds $(l - gaps(i, l), m - gaps(i, m)) \in P_i$.

where

$$gaps(i, j) = |\ \{l < j \mid a_{i,l} = \text{'-'}\}\ |\ .$$

13.4 Scoring structural alignments

Denote a structural alignment of k annotated sequences $(S_1, P_1), \dots, (S_k, P_k)$ with $A_s((S_1, P_1), \dots, (S_k, P_k))$, the set of all structural alignments of $(S_1, P_1), \dots, (S_k, P_k)$ with $\mathcal{A}_s^k((S_1, P_1), \dots, (S_k, P_k))$, and the set of all alignments between any k annotated sequences with $\mathcal{A}_s^k$. Define a *structural alignment score function* and the *optimal structural alignment score* as follows.

Definition 3 (structural alignment score)**.** A function $sc : \mathcal{A}_s^k \to \mathbb{R}$ is called *structural alignment score function*. If A_s is a structural alignment of k annotated sequences, then $sc(A_s)$ is called *structural alignment score* of A_s. The *optimal structural alignment score* of k annotated sequences $(S_1, P_1), \dots, (S_k, P_k)$ is defined as $sc^{opt}((S_1, P_1), \dots, (S_k, P_k)) := \max_{A_s \in \mathcal{A}_s^k((S_1,P_1),\dots,(S_k,P_k))} sc(A_s)$.

The following structural alignment score function $rna : \mathcal{A}_s^2 \to \mathbb{R}$ could be used to identify alignments that have both, high sequence and high structure conservation:

$$rna(A_s) = \sum_{i=1}^{n} sim(a_{1,i}, a_{2,i}) + \sum_{\substack{(j,l)\in\hat{P}_1,(q,r)\in\hat{P}_2 \\ (q,r)=(j,l)}} isim(a_{1,j}, a_{1,l}, a_{2,q}, a_{2,r}).$$

The above function assigns a high score not only to alignments with many matching characters in the sequence but also to alignments that show few such matches but many *interaction matches*. If we score a character match with 1 and an interaction match with 3, the second alignment above would be scored higher (score 10) than the first (score 8). It is clear that such a score could be used to define a score for a multiple alignment.

13.5 Computing structural alignments

Assume that we are given two annotated sequences (S_1, P_1) and (S_2, P_2) together with the structural alignment score function *rna*.

The goal is to compute an optimal structural alignment with respect to *rna*.

We will consider the case in which no pseudoknots are allowed in the secondary structures of structured sequences.

We present an general algorithm for annotated sequences. The running time will depend on whether the annotations are structures or not.

13.6 Computing structural alignments

We use a four-dimensional array A to record the value of solutions of subproblems. The entry $A[i_1, j_1, i_2, j_2]$ contains the score of an optimal structural alignment between the annotated sequences $(S_{1,[i_1:j_1]}, P_{1,[i_1:j_1]})$ and $(S_{2,[i_2:j_2]}, P_{2,[i_2:j_2]})$ where $P_{i,[j:k]}$ is the set of all interaction $(x, y) \in P$ with $j \le x < y \le k$. Hence $A[1, n_1, 1, n_2]$ contains the score of the optimal structural alignment between (S_1, P_1) and (S_2, P_2).

We fill the fourdimensional matrix for intervals of increasing widths.

1. The score of aligning an empty infix of (S_1, P_1) with an empty infix of (S_2, P_2) is zero. Hence $A[i_1, j_1, i_2, j_2] = 0$ for all quadruples (i_1, j_1, i_2, j_2) with $j_1 < i_1$ and $j_2 < i_2$.
2. Aligning an empty infix of (S_1, P_1) with an nonempty infix of (S_2, P_2) yields as score the sum of all insertion scores for the infix of the second sequence, i. e., $A[i_1, j_1, i_2, j_2] = \sum_{i_2 \le k \le j_2} sim('\text{-}', s_{2,k})$ for all quadruples (i_1, j_1, i_2, j_2) with $j_1 < i_1$ and $j_2 \ge i_2$.
3. Aligning an empty infix of (S_2, P_2) with an nonempty infix of (S_1, P_1) yields as score the sum of all deletion scores for the infix of the second sequence, i. e., $A[i_1, j_1, i_2, j_2] = \sum_{i_1 \le k \le j_1} sim(s_{1,k}, '\text{-}')$ for all quadruples (i_1, j_1, i_2, j_2) with $j_2 < i_2$ and $j_1 \ge i_1$.

Assume now that we do not allow any pseudoknots, then the following recurrences hold:

$$\begin{aligned}\hat{A}[i_1, j_1, i_2, j_2] = \max\Big\{ & A[i_1, j_1 - 1, i_2, j_2] + sim(s_{1,j_1}, '\text{-}'),\\ & A[i_1, j_1, i_2, j_2 - 1] + sim('\text{-}', s_{2,j_2}),\\ & A[i_1, j_1 - 1, i_2, j_2 - 1] + sim(s_{1,j_1}, s_{2,j_2})\Big\}\end{aligned} \tag{13.1}$$

and, for all $k_1 \ge i_1, k_2 \ge i_2$ such that $(k_1, j_1) \in P_1$ and $(k_2, j_2) \in P_2$

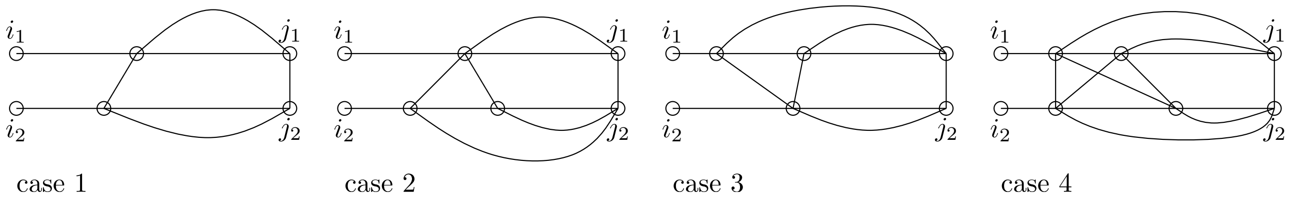
$$\begin{aligned}A[i_1, j_1, i_2, j_2] = \max\Big\{ & \hat{A}[i_1, j_1, i_2, j_2],\\ & A[i_1, k_1 - 1, i_2, k_2 - 1]\\ & + A[k_1 + 1, j_1 - 1, k_2 + 1, j_2 - 1]\\ & + sim(s_{1,k_1}, s_{2,k_2}) + sim(s_{1,j_1}, s_{2,j_2})\\ & + isim(s_{1,k_1}, s_{1,j_1}, s_{2,k_2}, s_{2,j_2})\Big\}\end{aligned} \tag{13.2}$$

The intervals must be computed in increasing order of width. For example in this lexicographical order $(i_1, j_1, i_2, j_2) = (1, 2, 1, 2)$,$(1, 2, 2, 3)$,. . .,$(1, 2, n_2 - 1, n_2)$,$(2, 3, 1, 2)$,. . ..

Part (**??**) of the above recurrence computes similarly to sequence alignment all possibilities of extending the primary sequence to the right.

Part (2) is new. Basically, whenever we encounter a potential right end (j_1, j_2) of an interaction match, we check for all possibilities within $[i_1, j_1]$ and $[i_2, j_2]$ to be the left end of the interaction match.

How many such possibilities exist depends on whether P_1 and P_2 are an annotation or a structure.



1. P_1 is a structure and P_2 is a structure.
   That means there is only one possible interaction match.
2. P_1 is a structure and P_2 is not a structure.
   That means there are $O(n_2)$ possible interaction matches.
3. P_1 is not a structure and P_2 is a structure.
   That means there are $O(n_1)$ possible interaction matches.
4. P_1 is a not structure and P_2 is not a structure.
   That means there are $O(n_1 \cdot n_2)$ possible interaction matches.

Theorem 4. *Assume that we are given two annotated sequences (S_1, P_1) and (S_2, P_2) together with the structural alignment score function rna. The above recurrences and the initializations compute an optimal structural alignment not containing pseudoknots with respect to rna.*

Proof: exercise. Hint: use structural induction.

Theorem 5. *The described algorithm needs space $O(n_1^2 \cdot n_2^2)$ and time $O(n_1^2 \cdot n_2^2)$, $O(n_1^2 \cdot n_2^3)$ (or $O(n_1^3 \cdot n_2^2)$), and $O(n_1^3 \cdot n_2^3)$ for the case of P_1 and P_2 being structures, exactly one of P_1 or P_2 being a structure, and P_1 and P_2 being an annotation respectively.*

Proof: In part (2) of the recurrence we have to consider all possible cases with j_1,j_2 being the right match of an interaction match, and the left match of the interaction match lying in the current interval. For P_1 and P_2 being a structure, only one such left match is possible. Hence the step takes constant time. For the other cases the step takes $O(n_1)$ (or $O(n_2)$) time and $O(n_1 \cdot n_2)$ time. Since this is executed for each cell of the four-dimensional array, the time bounds follow.

13.7 Improving the run time

Intuitively the described algorithm is not optimal. Given the right match of an interaction match, we check for its left match for *each* interval that ends with the right match. for the structure–structure case this does not hurt so much, since it is only a constant overhead. However, when one of P_1 and P_2 is an annotation we pay dearly for this.

We will now consider the case where one annotation (w.l.o.g P_1) is a structure and the other an annotation. Then we will make some modifications to the recurrences that will reduce the running time from $O(n_1^2 \cdot n_2^3)$ to $O(n_1^2 \cdot n_2^2) + O(n_1 \cdot n_2^3)$.

The first step to make is to *binarize* the tree representation of P_1. That means we add to P_1 a set of artificial interactions resulting in a set P' as follows:

```
(1)  Binarize($i, j$);
(2)  $(i, j) \in P_1$ has $k$ children $\{(i_1, j_1), \dots, (i_k, j_k)\}$;
(3)  for $1 \le u \le k$ do
(4)      $Binarize(i_u, j_u)$;
(5)      $P' = P' \cup \{(i_1, j_u)\}$;
(6)      if $u > 1$ then
(7)                  $parent((i_1, j_{u-1})) = (i_1, j_u)$;
(8)                  $parent((i_u, j_u)) = (i_1, j_u)$;
(9)      fi
(10) od
(11) $parent((i_1, j_k)) = (i, j)$;
```

The for-loop of the above algorithm is called for the list of all the visible intervals.

Consider the following example in mountain notation:

```
      =======
   =============  =======  =======
G  A  C  A  G  U  U  C  A  C  G  G  C
1  2  3  4  5  6  7  8  9 10 11 12 13

after binarize:

      =======
   =============  =======  =======
   ----------------------
   -------------------------------
G  A  C  A  G  U  U  C  A  C  G  G  C
1  2  3  4  5  6  7  8  9 10 11 12 13
```

'children' $(2, 6), (7, 9), (10, 12)$.

- $u = 1$:
  Binarize(2,6);
  $P' = P' \cup \{(2, 6)\}$;
- $u = 2$:
  Binarize(7,9);
  $P' = P' \cup \{(2, 9)\}$;
  parent((2,6))=parent((7,9))=(2,9);
- $u = 3$:
  Binarize(10,12);
  $P' = P' \cup \{(2, 12)\}$;
  parent((2,9))=parent((10,12))=(2,12);

Binarize(2,6), children $(3, 5)$.

- $u = 1$:
  Binarize(3,5);
  $P' = P' \cup \{(3, 5)\}$;

parent((3,5))=(2,6);

So in the end $P' = \{(3,5),(2,6),(2,9),(2,12)\}$.

Then the new recurrence reads as follows with identical initializations:

$$\begin{aligned}\hat{A}[i_1,j_1,i_2,j_2] = \max\Big\{ & A[i_1+1,j_1,i_2,j_2] + sim(s_{1,i_1},'\text{-}'),\\ & A[i_1,j_1,i_2+1,j_2] + sim('\text{-}',s_{2,i_2}),\\ & A[i_1+1,j_1,i_2+1,j_2] + sim(s_{1,i_1},s_{2,i_2}),\\ & A[i_1,j_1-1,i_2,j_2] + sim(s_{1,j_1},'\text{-}'),\\ & A[i_1,j_1,i_2,j_2-1] + sim('\text{-}',s_{2,j_2}),\\ & A[i_1,j_1-1,i_2,j_2-1] + sim(s_{1,j_1},s_{2,j_2})\Big\}\end{aligned} \tag{13.3}$$

and, if $(i_1,j_1) \in P_1$ and $(i_2,j_2) \in P_2$

$$\begin{aligned}A[i_1,j_1,i_2,j_2] = \max\Big\{ & \hat{A}[i_1,j_1,i_2,j_2],\\ & A[i_1+1,j_1-1,i_2+1,j_2-1]\\ & \;+ sim(s_{1,i_1},s_{2,i_2}) + sim(s_{1,j_1},s_{2,j_2})\\ & \;+ isim(s_{1,i_1},s_{1,j_1},s_{2,i_2},s_{2,j_2})\Big\}\end{aligned} \tag{13.4}$$

and, if $(i_1,j_1) \in P' \setminus P_1$ and $(k,j_1) = rightchild(i_1,j_1)$

$$\begin{aligned}A[i_1,j_1,i_2,j_2] = \max\Big\{ & \hat{A}[i_1,j_1,i_2,j_2],\\ & \max_{i_2<l<j_2}\{A[i_1,k-1,i_2,l-1] + A[k,j_1,l,j_2]\}\Big\}\end{aligned} \tag{13.5}$$

Theorem 6. *Assuming that P_1 is a secondary structure, the new recurrence needs space $O(n_1^2 \cdot n_2^2)$ and time $O(n_1^2 \cdot n_2^2 + n_1 \cdot n_2^3)$.*

Proof: Parts (**??**) and (**??**) need constant time at each step of filling the four-dimensional table. Part (**??**) of the recurrence needs time $O(n_2)$. But it is called only $O(|P'| \cdot n_2^2) = O(n_1 \cdot n_2^2)$ times. From this the claimed bound follows.

What does the new recurrence do differently?

1. The old recurrence triggered the expensive part for each interval where j_1, j_2 was the right end of a possible interaction match.
2. The new recurrence triggers the expensive part *only* for intervals that define an artificial interaction, which in turn adds multiloops together.
3. This also warrants the extension to the left and *right* and the additional case of a real interaction match.

Theorem 7. *Assuming that P_1 is a secondary structure, the new algorithm is correct.*

Proof: exercise (hard).

13.8 Summary

- Similar to sequence logos, we can visualize the mutual information contained in RNA sequence-structure alignments.
- RNA sequence-structure alignments without pseudoknots can be computed using dynamic programming.

- Te time requirement for the DP algorithm depends on the fact whether the annotations of the sequences are a structure or not.
- If one annotation is a structure, then the running time can be improved.