

RNA sequence-structure alignment

- RNA alignments:
 - sequence-sequence alignment.
 - sequence-structure alignment.
 - It's most useful for ncRNA finding.
 - structure-structure alignment.

What is an alignment?

- Alignment of two sequences

```
ACCCG-UUAAU
|  |||  ||*||
A-CCGUUUCAU
```

- Sequence-structure alignment:
 - One RNA sequence with its known secondary structure.
 - One just sequence
 - Take into account both structure similarity and sequence similarity.

```
Stru1: >>>> <<<<<
Seq1:  GGGGCAACCCC
      +++++*||+++++
Seq2:  AUCCGAAGGAU
      | | | | |
      | | | | |
      | | | | |
      | | | | |
      | | | | |
```

RNA sequence-structure alignment

- Given two RNA sequence $s[1 \dots n]$ and $t[1 \dots m]$, and s has a known secondary structure S , where (i,j) in S implies $s[i]$ is basepaired with $s[j]$.
- Score for the alignment is the sum of scores γ of each column plus the sum of scores δ of each basepair in S .
- Scores for each column:
 - $\gamma(a,b)$ for all a, b in $\{A, C, G, U, -\}$.
- Scores for two basepair columns:
 - $\delta(a, b, c, d)$ for all a, b, c , and d in $\{A, C, G, U\}$.
 - For example $\delta(a, b, c, d) = 1$ if (a,b) in S and c and d form basepair, otherwise $-\infty$.

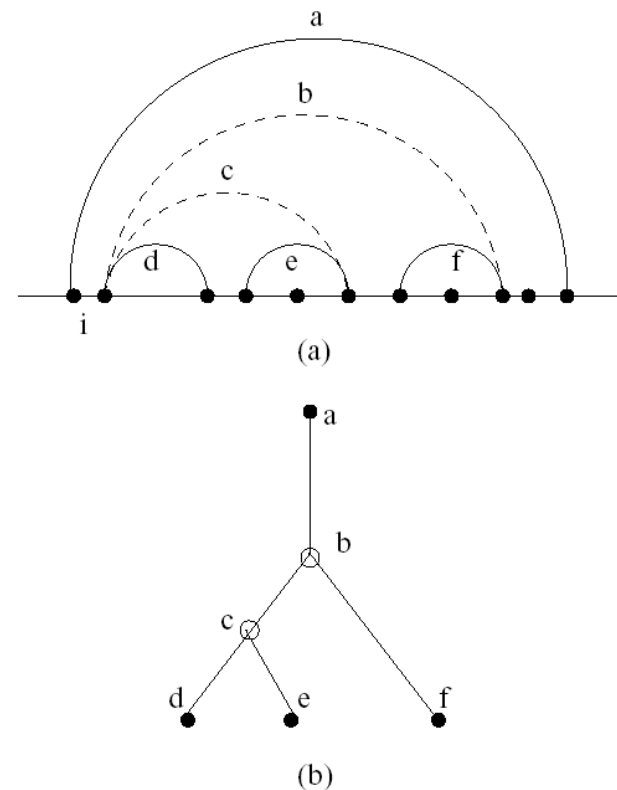
Problem 3: [RNA sequence-structure alignment problem] Given two RNA sequence $s[1..n]$ and $t[1..m]$, where s has a known secondary structure without pseudoknots, compute the highest scoring alignment.

Analysis: alignment solutions

- Similar to Nussinov's base pair maximization problem's solution, the obvious dynamic programming solution is in $O(n^6)$ time.
- Bafna et al. (1995) improved the dynamic programming algorithm to a running time of $O(n^4)$ by using binarized tree for structure S .
- We further improve the algorithm by building a binarized tree for whole sequence $s[1 \dots n]$ with its structure, and reduce the computation time into $O(n^3 m_1)$, where m_1 is the number of the branches in the tree (typically very small constant).

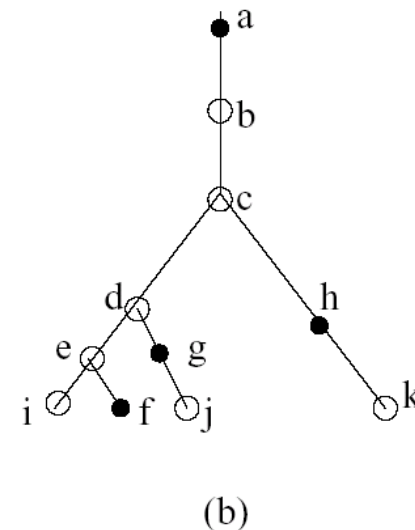
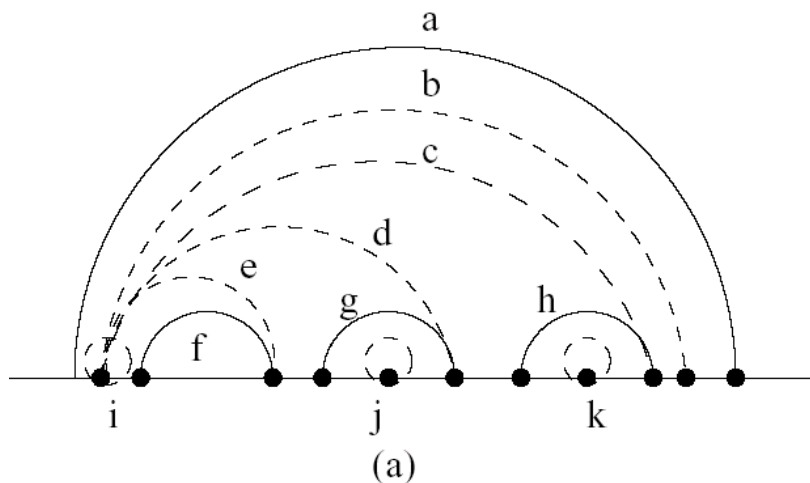
The method of Bafna et al. (1995)

- Adding spurious (dashed) edges to S , so that each node in S' has at most 2 children.
- We use the dashed edges (void nodes) to fix certain interval doing recursions, when we need find the branch points.
- It is in $O(n^2m^2+nm^3)$ time.



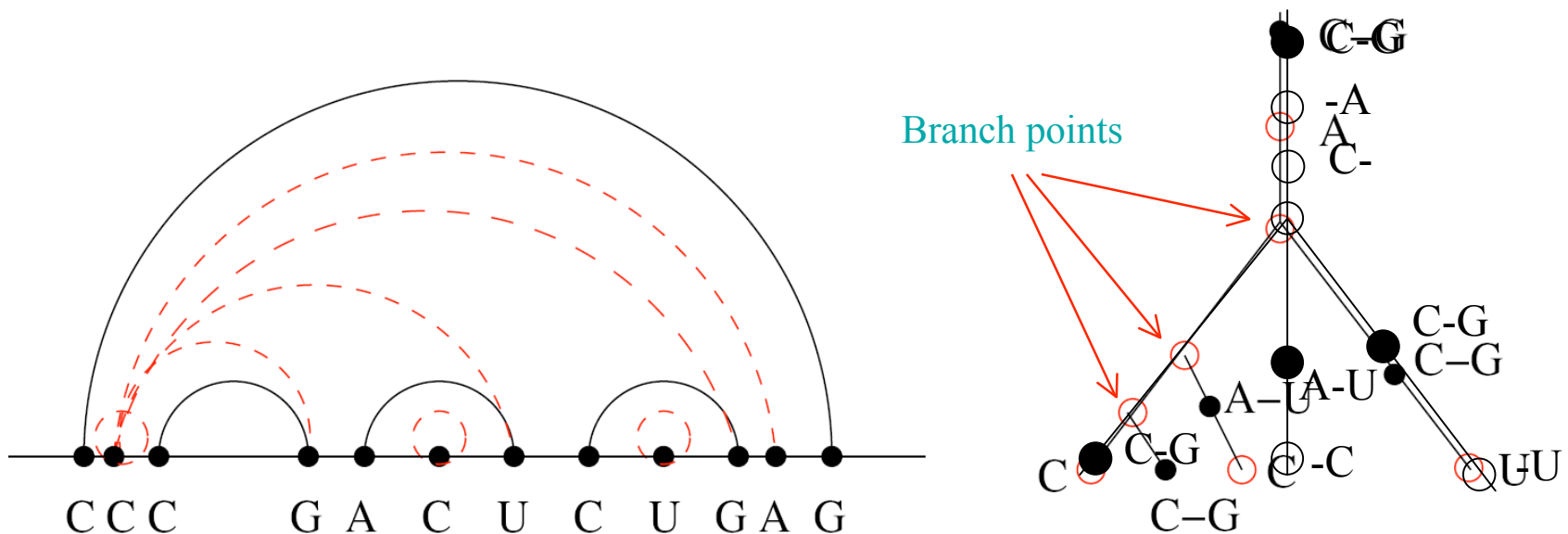
Extend the binary tree into the whole sequence $s[1\dots n]$

- Solid nodes represent the basepairs.
- Dotted nodes represent either branch point or unpaired bases.
- Now the $s[1\dots n]$ with its structure is changed into a binary tree.
- The dynamic programming compares each nodes against an interval.
- It is in $O(n^3m_1)$ time. (m_1 the number of branch points)



Using a binary tree to represent an RNA

- Solid nodes represent the basepairs.
- Void nodes represent either branch point or unpaired bases.



The dynamic programming aligns each subtree rooted at node v against an subinterval (i, j)

If root v is a solid node
(first and last bases of the subsequence represented by the subtree are paired to each other):

If v is a void node
(the subsequence's last base is unpaired):

If v is a void node
(it is a branch point for multi-loop.):

procedure alignRNA

(* S is the set of base-pairs in RNA structure of s . S' is the augmented set. *)

for all intervals (i, j) , $1 \leq i < j \leq n$, all nodes $v \in S'$

$A[i, j, v] = \max \left\{ \begin{array}{l} A[i + 1, j - 1, \text{child}(v)] + \gamma(s[l_v], t[i]) + \gamma(s[r_v], t[j]) \\ \quad + \delta(t[i], t[j], s[l_v], s[r_v]), \\ A[i, j - 1, v] + \gamma(' - ', t[j]), \\ A[i + 1, j, v] + \gamma(' - ', t[i]), \\ A[i + 1, j, \text{child}[v]] + \gamma(s[l_v], t[i]) + \gamma(s[r_v], ' - '), \\ A[i, j - 1, \text{child}[v]] + \gamma(s[l_v], ' - ') + \gamma(s[r_v], t[j]), \\ A[i, j, \text{child}[v]] + \gamma(s[l_v], ' - ') + \gamma(s[r_v], ' - '), \end{array} \right.$	$A[i, j, v] = \max \left\{ \begin{array}{l} A[i, j - 1, \text{child}[v]] + \gamma(s[r_v], t[j]), \\ A[i, j, \text{child}[v]] + \gamma(s[r_v], ' - '), \\ A[i, j - 1, v] + \gamma(' - ', t[j]), \\ A[i + 1, j, v] + \gamma(' - ', t[i]), \end{array} \right.$
$A[i, j, v] = \max_{i \leq k \leq j} \{ A[i, k - 1, \text{left_child}[v]] + A[k, j, \text{right_child}[v]] \}$	

end if

Modifications of the formulation

- Banded alignment. [$O(n^2\delta_n)$]
- Complicated scoring functions:
 - Affine gap penalties in both stacks and loops.
 - Learning scoring matrix from handmade alignments.