Structure-Based Comparison of Biomolecules

Benedikt Christoph Wolters

Seminar Bioinformatics Algorithms

RWTH AACHEN

07/17/2015

Outline

Motivation

- *Previous topics in the seminar:*
  Similarities of molecules (RNA sequences) solely based on primary structure (Recall: Talks for Chapter 5)

Motivation

- *Previous topics in the seminar:*
  Similarities of molecules (RNA sequences) solely based on primary structure (Recall: Talks for Chapter 5)
- *However:*
  In order to derive the functions of molecules in living beings the spatial structure is of essential significance

Motivation

- *Previous topics in the seminar:*
  Similarities of molecules (RNA sequences) solely based on primary structure (Recall: Talks for Chapter 5)
- *However:*
  In order to derive the functions of molecules in living beings the spatial structure is of essential significance
- *Now:*
  Incorporate additional knowledge of spatial structure into the similarity comparison

Recapitulation: Protein Structure Hierarchy

Primary Structure: Sequence of nucleotides (Strings)

Secondary Structure: Folding of the RNA with itself (e.g., by hydrogen bounds)

Tertiary Structure: *real* spatial conformation: positions of single atoms in space, angle of bindings, etc.

Example

Primary Structure

AGGUCAGU...

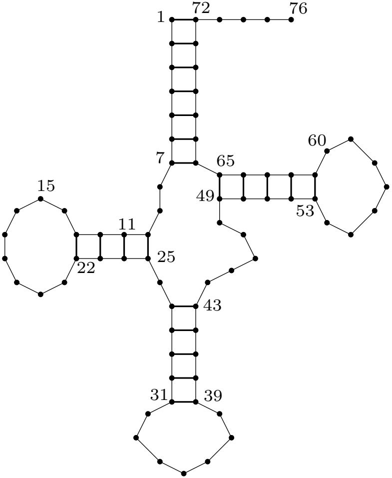
Images from Böckenhauer, Bongarts – Algorithmic Aspects of Bioinformatics (2007), p. 320

Example

Primary Structure

AGGUCAGU...

Secondary Structure



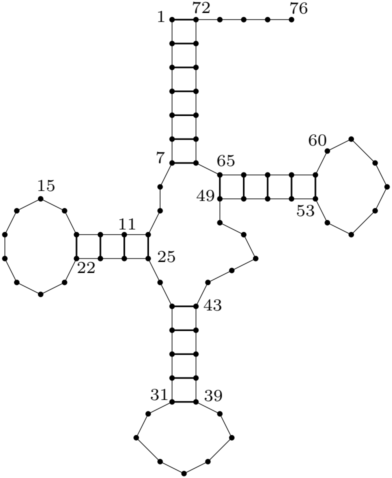
Images from Böckenhauer, Bongarts – Algorithmic Aspects of Bioinformatics (2007), p. 320

Example

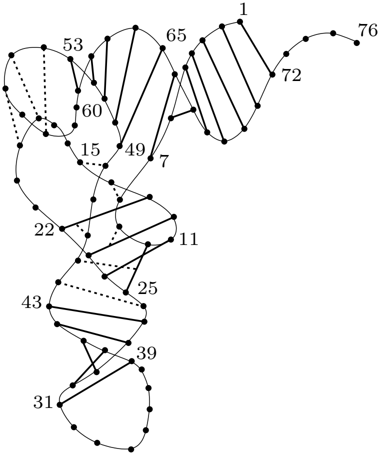
Primary Structure

`AGGUCAGU...`

Secondary Structure



Tertiary Structure



Images from Böckenhauer, Bongarts – Algorithmic Aspects of Bioinformatics (2007), p. 320

Protein Data Bases

There are several databases containing the higher-level structural information of biological molecules obtained by

- X-Ray crystallography, or
- NMR spectroscopy.

Examples:

Protein Data Bank (PDB)

http://www.rcsb.org/pdb/

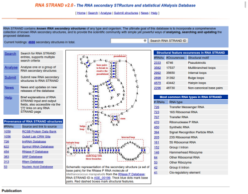
100.000 entries

RNA STRAND

http://www.rnasoft.ca/strand/

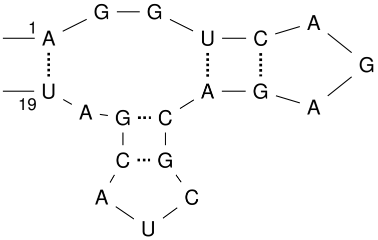
focused on RNA secondary structure

4.000 entries



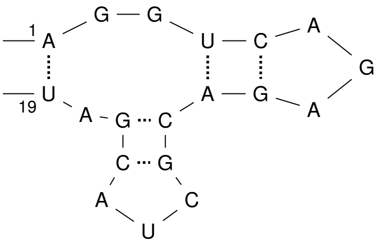
From Secondary Structures to Arc-Annotated Sequences

Goal: Find representation that enables processing/comparison of secondary structure.



From Secondary Structures to Arc-Annotated Sequences

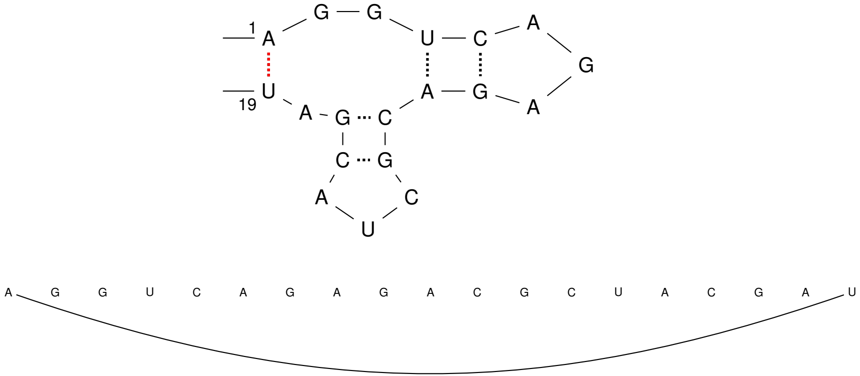
Goal: Find representation that enables processing/comparison of secondary structure.



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

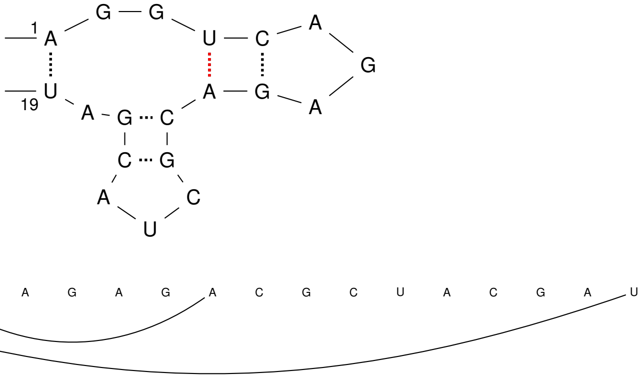
From Secondary Structures to Arc-Annotated Sequences

Goal: Find representation that enables processing/comparison of secondary structure.



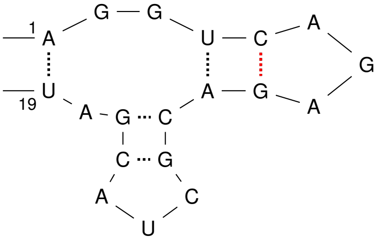
From Secondary Structures to Arc-Annotated Sequences

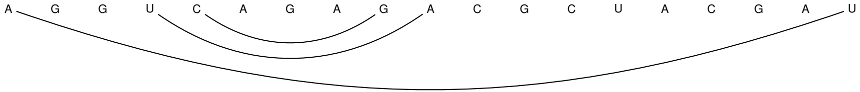
Goal: Find representation that enables processing/comparison of secondary structure.



From Secondary Structures to Arc-Annotated Sequences

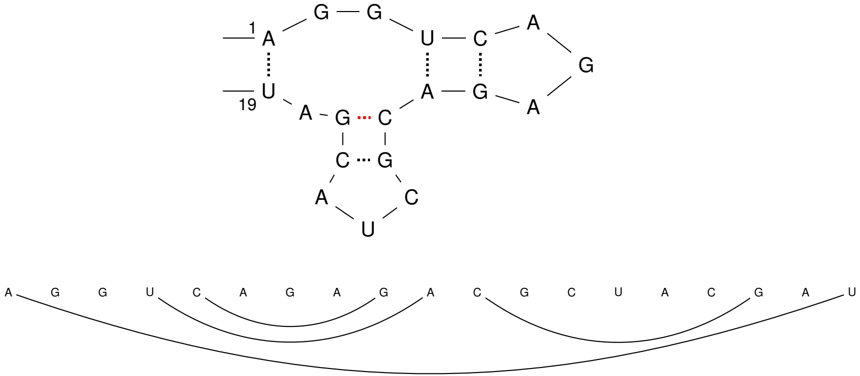
Goal: Find representation that enables processing/comparison of secondary structure.





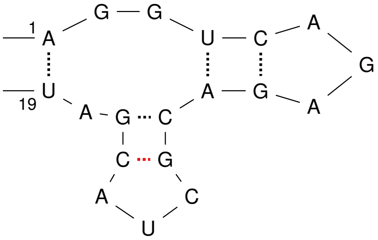
From Secondary Structures to Arc-Annotated Sequences

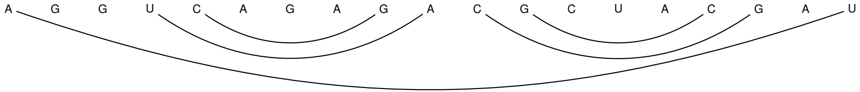
Goal: Find representation that enables processing/comparison of secondary structure.



From Secondary Structures to Arc-Annotated Sequences

Goal: Find representation that enables processing/comparison of secondary structure.





Arc-Annotated Sequence

Definition

Let $s = s_1 s_2 \dots s_n$ be a string over an alphabet Σ and let $P \subseteq \{(i,j) | 1 \leq i \leq j \leq n\}$ be an unordered set of position pairs in s.
We call $S = (s, P)$ an *arc-annotated string* with string s and arc set P. A pair from the arc set P is called an *arc*.

Classes of Arc-Annotated Sequences

C1 No two arcs share a common endpoint

C2 No two arcs cross each other

C3 No two arcs are nested

UNLIMITED	No restrictions
CROSSING	C1
NESTED	C1, C2
CHAIN	C1, C2, C3
PLAIN	No arcs at all

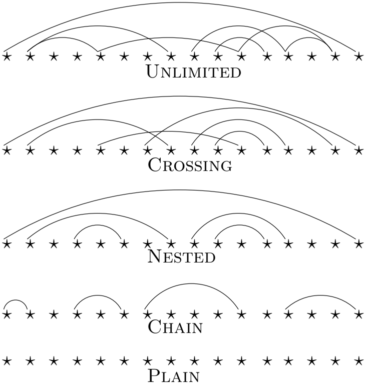


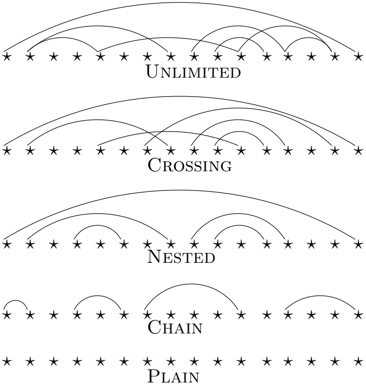
Figure from Böckenhauer, Bongarts – Algorithmic Aspects of Bioinformatics (2007), p. 341

Classes of Arc-Annotated Sequences

C1 No two arcs share a common endpoint

C2 No two arcs cross each other

C3 No two arcs are nested

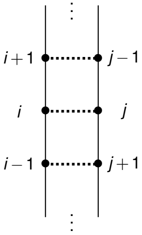


UNLIMITED	No restrictions
CROSSING	C1
NESTED	C1, C2
CHAIN	C1, C2, C3
PLAIN	No arcs at all

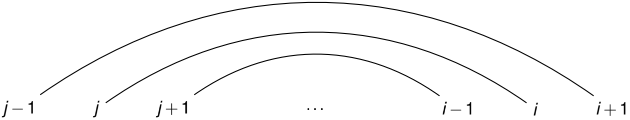
PLAIN $\subsetneq$ CHAIN $\subsetneq$ NESTED $\subsetneq$ CROSSING $\subsetneq$ UNLIMITED.

Figure from Böckenhauer, Bongarts – Algorithmic Aspects of Bioinformatics (2007), p. 341

Patterns and Substructures in RNA

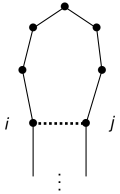


Stem

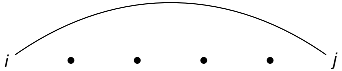


Corresponding arc-annontated string featuring a Stem
$\Rightarrow$ NESTED

Patterns and Substructures in RNA

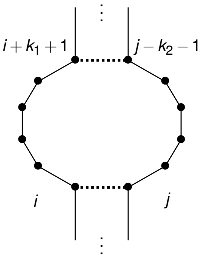


Hairpin Loop

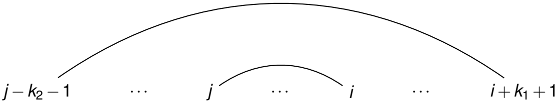


Arc-annontated string for a Hairpin Loop
$\Rightarrow$ CHAIN $\subseteq$ NESTED

Patterns and Substructures in RNA

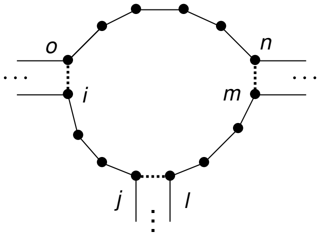


Interior Loop

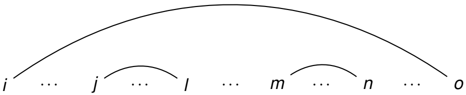


Corresponding arc-annontated string $\Rightarrow$ NESTED

Patterns and Substructures in RNA



Multiple Loop



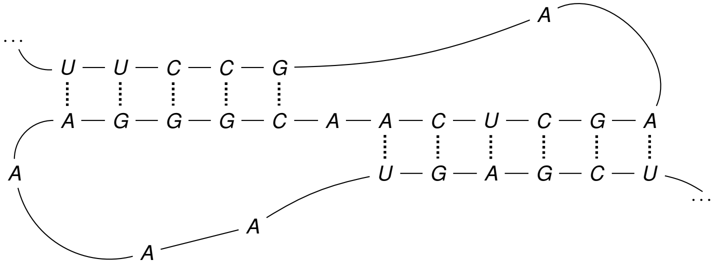
Corresponding arc-annontated string
⇒ NESTED

Excourse: Pseudoknots

Definition (Pseudoknot)

The secondary structure contains a pseudoknot if there exists two base pairs (i, j) and (k, l) such that $i < k < j < l$ holds.

Example

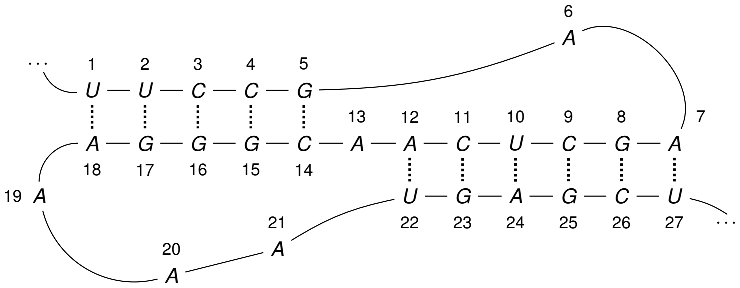


Excourse: Pseudoknots

Definition (Pseudoknot)

The secondary structure contains a pseudoknot if there exists two base pairs (i, j) and (k, l) such that $i < k < j < l$ holds.

Example

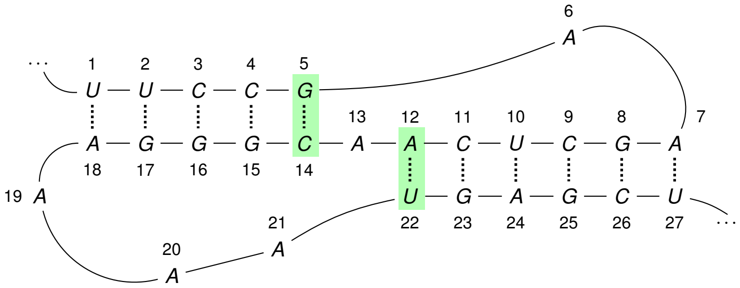


Excourse: Pseudoknots

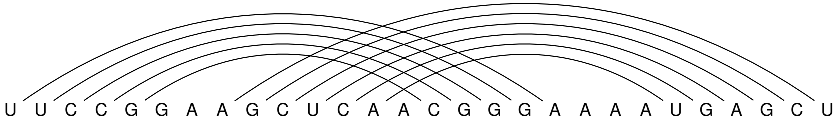
Definition (Pseudoknot)

The secondary structure contains a pseudoknot if there exists two base pairs (i, j) and (k, l) such that $i < k < j < l$ holds.

Example



Pseudoknots in Arc-Annotated Sequences



Secondary structures with a pseudonknot translated to arc-annotated sequences will be in CROSSING.

Consistent Mapping

Definition (Consistent Mapping)

Let $s = s_1 s_2 \ldots s_n$ and $t = t_1 t_2 \ldots t_m$ be two strings and let $w = w_1 w_2 \ldots w_k$ be a common subsequence of s and t.
Then a bijective mapping φ from a subset $M_s \subseteq \{1, \ldots, n\}$ onto a subset $M_t \subseteq \{1, \ldots, m\}$ is called *consistent with w* if it satisfies the following properties:

1. Mapping φ preserves the order of symbols along the strings s and t, i.e., for all $i_1, i_2 \in M_s$,

$$i_1 < i_2 \Leftrightarrow \varphi(i_1) < \varphi(i_2).$$

2. The symbols on positions assigned by φ are equal, i.e., for all $i \in M_s$,

$$s_i = t_{\varphi(i)}$$

In the following, we also write

$$\langle x, y \rangle \in \varphi \iff \varphi(x) = y$$

Arc-Preserving Common Subsequences

Definition (Arc-Preserving Common Subsequence)

Let $S = (s_1 s_2 \ldots s_m, P_s)$ and $T = (t_1 t_2 \ldots t_n, P_t)$ be two arc-annotated sequences over an alphabet Σ. A string is called an *arc-preserving common subsequence of S and T* if there exists a common subsequence w of s and t and a mapping φ consistent with w such that

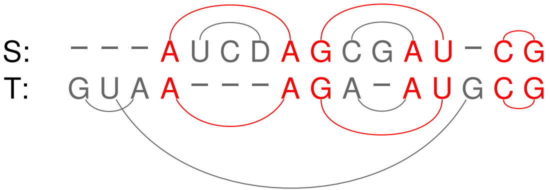
1. $s_i = t_j$ for all $\langle i, j \rangle \in \varphi$, and
2. for all pairs of elements $(\langle i_1, j_1 \rangle, \langle i_2, j_2 \rangle)$ from φ

$$(i_1, i_2) \in P_s \iff (j_1, j_2) \in P_t.$$

Example

$$\Sigma = \{A, G, U, C\}$$

$$\varphi = \{\langle 1,4\rangle, \langle 5,5\rangle, \langle 6,6\rangle, \langle 9,8\rangle, \langle 10,9\rangle, \langle 11,11\rangle, \langle 12,12\rangle\}$$



Definition (LAPCS(Level_1,Level_2))

By LAPCS(Level_1,Level_2 we denote the optimization problem for two arc-annotated strings $S \in \text{Level}_1$ and $T \in \text{Level}_2$ to find the longest common arc-annotated substring.

LAPCS(PLAIN,PLAIN)

Theorem

The optimization problem LAPCS(PLAIN,PLAIN) *is computable in* $\mathcal{O}(m \cdot n)$*, where m and n are the length of the input strings.*

Proof.

This problem is the same as the global alignment problem discussed in a previous talk. We can leverage dynamic programming and backtracking to solve this. □

NP-Hardness of LAPCS(CROSSING,CROSSING)

Theorem

LAPCS(CROSSING,CROSSING) *is an NP-hard optimization problem.*

Idea: Consider DECLAPCS, the corresponding decision problem of LAPCS. Reduce input instance of CLIQUE to DECLAPCS.

Recap: CLIQUE Problem

Definition

Let $G = (V, E)$ be an undirected graph. A subset $V' \subseteq V$ is called a *clique*, if every two vertices $v_i, v_j \in V'$, where $v_i \neq v_j$ are connected by an edge, i.e., $\{v_i, v_j\} \in E$.

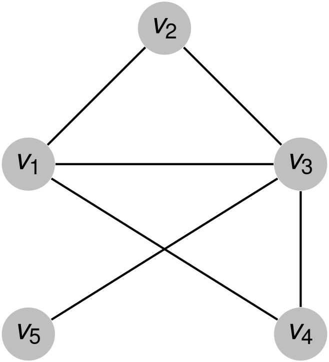
Definition (CLIQUE Decision Problem)

Input: An undirected graph $G = (V, E)$ and a positive integer k.
Output: YES if G contains a clique V' of size k, NO, otherwise.

Clique is a well-known NP-complete decision problem.

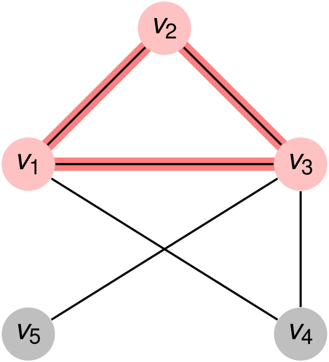
Example: CLIQUE

Is there a clique for $k = 3$?

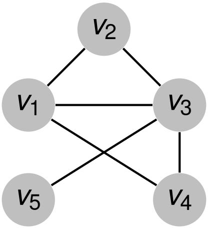


Example: CLIQUE

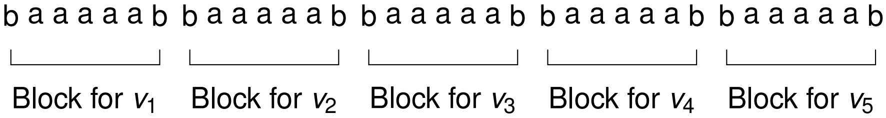
Is there a clique for $k = 3$?



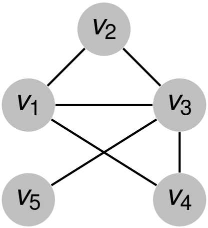
Arc-Annotated String Construction from Input-Graph



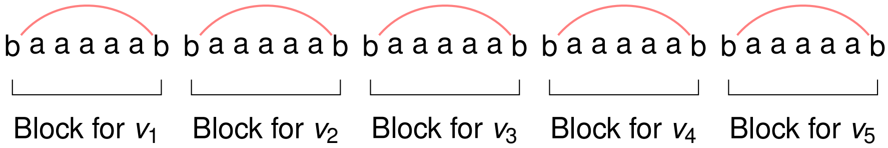
S :



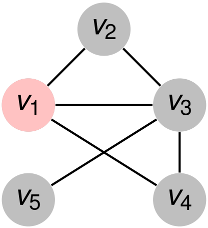
Arc-Annotated String Construction from Input-Graph



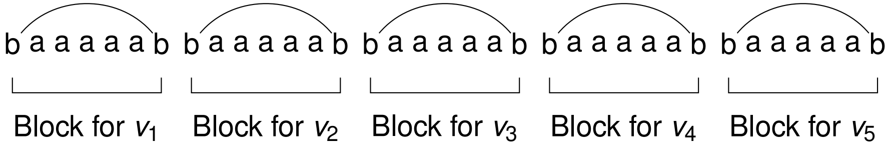
S :



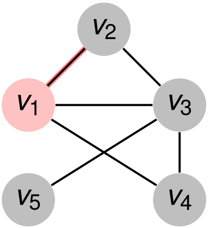
Arc-Annotated String Construction from Input-Graph



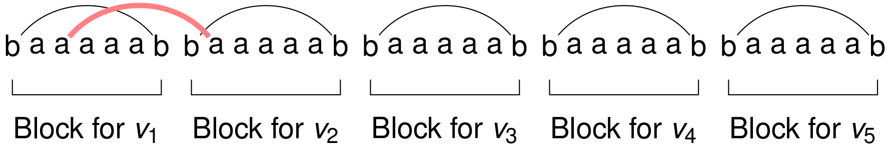
S :



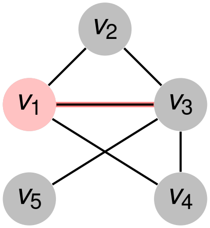
Arc-Annotated String Construction from Input-Graph



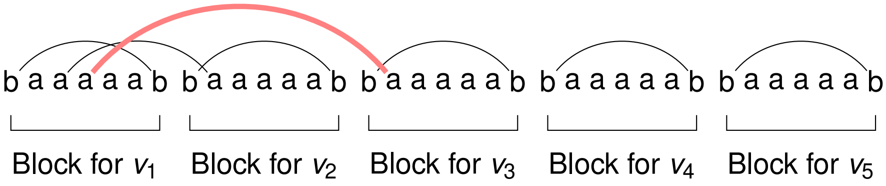
S :



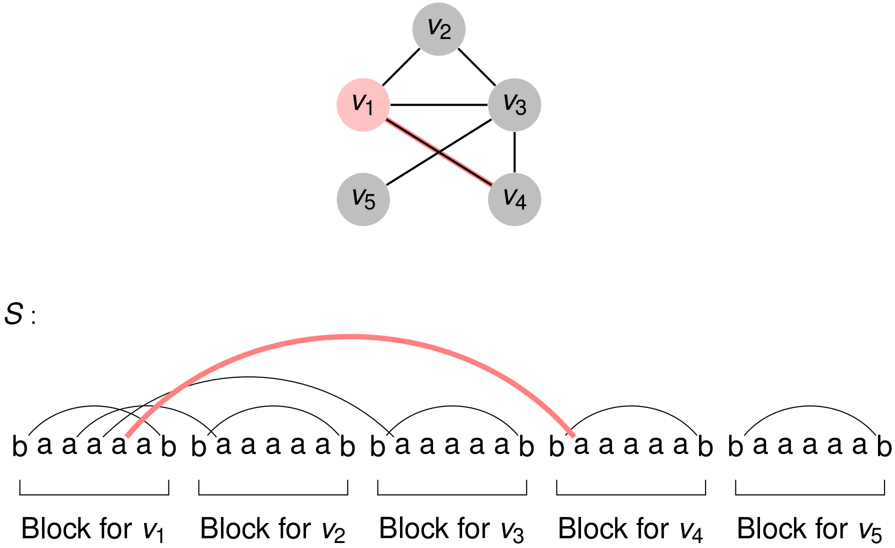
Arc-Annotated String Construction from Input-Graph



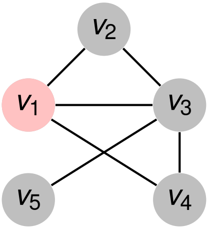
S :



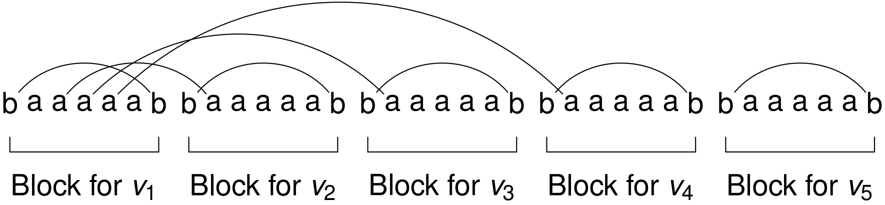
Arc-Annotated String Construction from Input-Graph



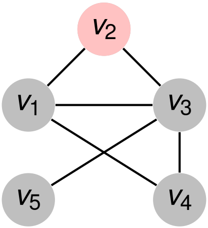
Arc-Annotated String Construction from Input-Graph



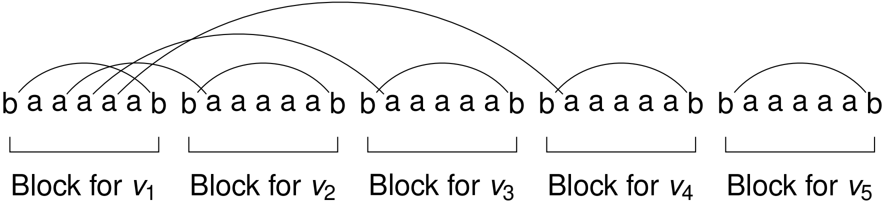
S :



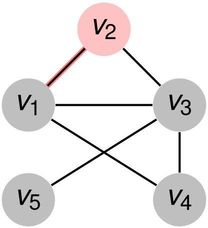
Arc-Annotated String Construction from Input-Graph



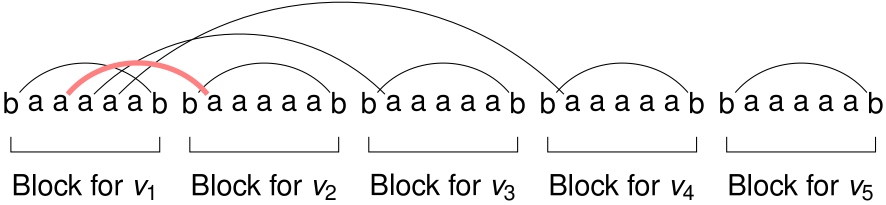
S :



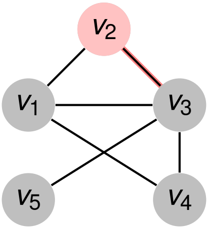
Arc-Annotated String Construction from Input-Graph

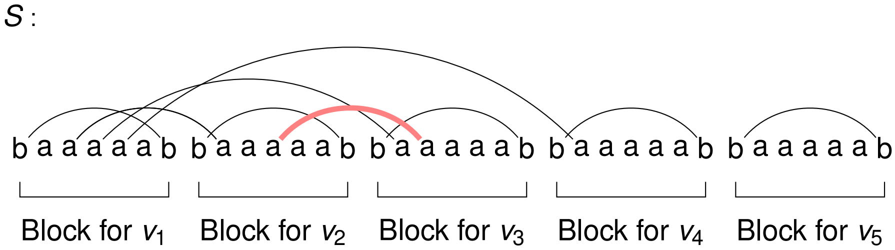


S :

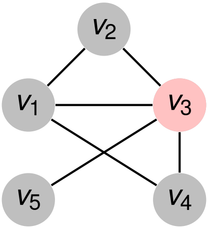


Arc-Annotated String Construction from Input-Graph

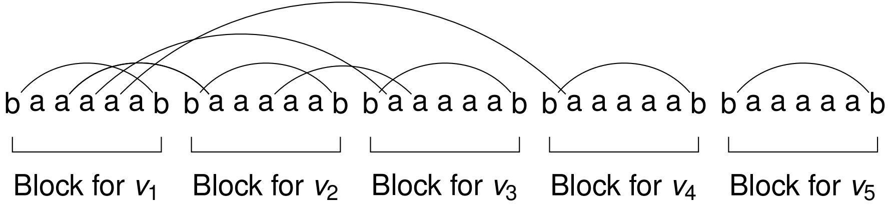




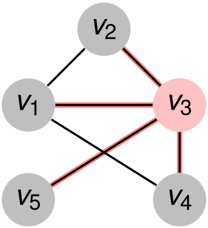
Arc-Annotated String Construction from Input-Graph



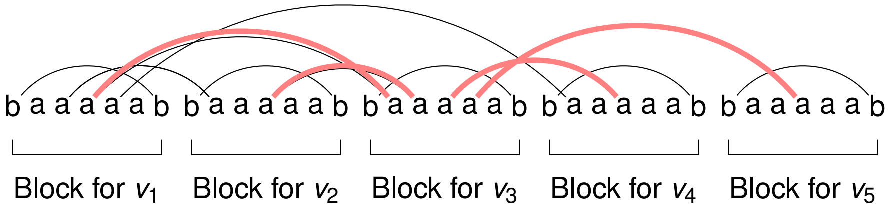
S :



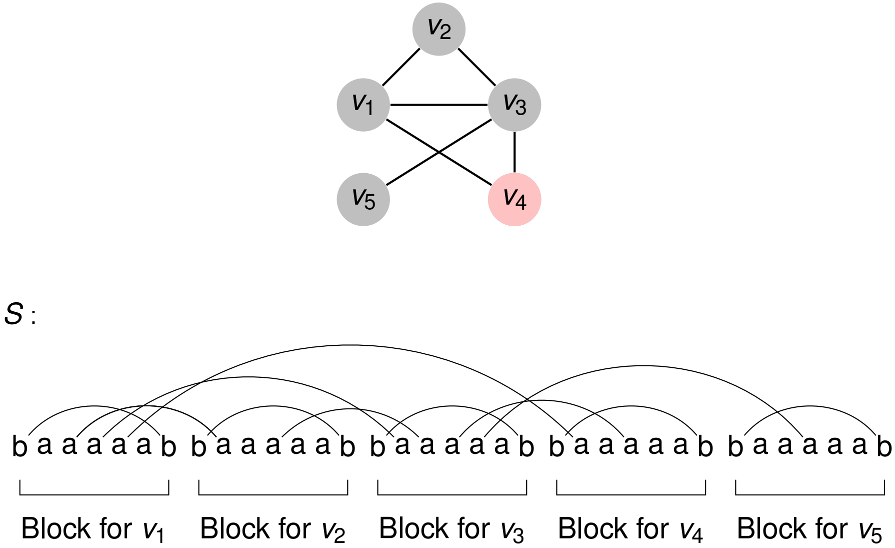
Arc-Annotated String Construction from Input-Graph



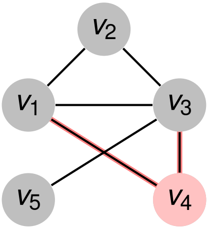
S :



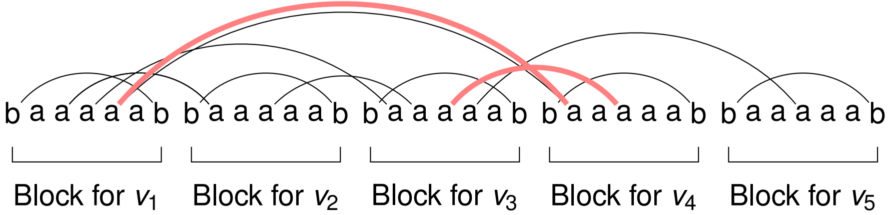
Arc-Annotated String Construction from Input-Graph



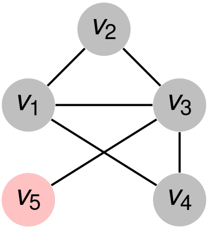
Arc-Annotated String Construction from Input-Graph



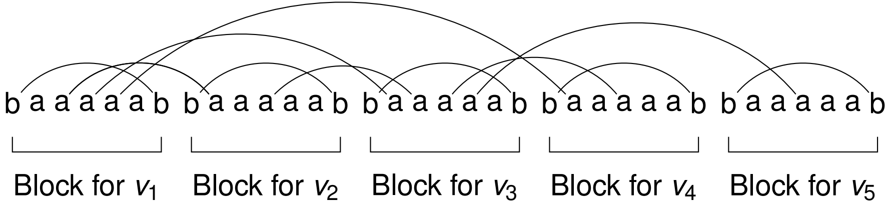
S :



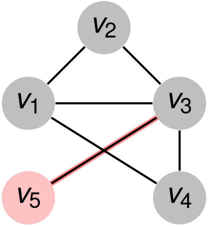
Arc-Annotated String Construction from Input-Graph



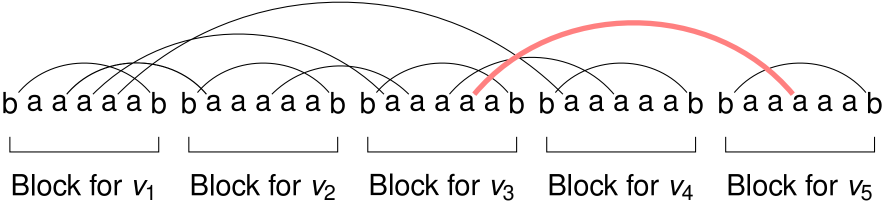
S :



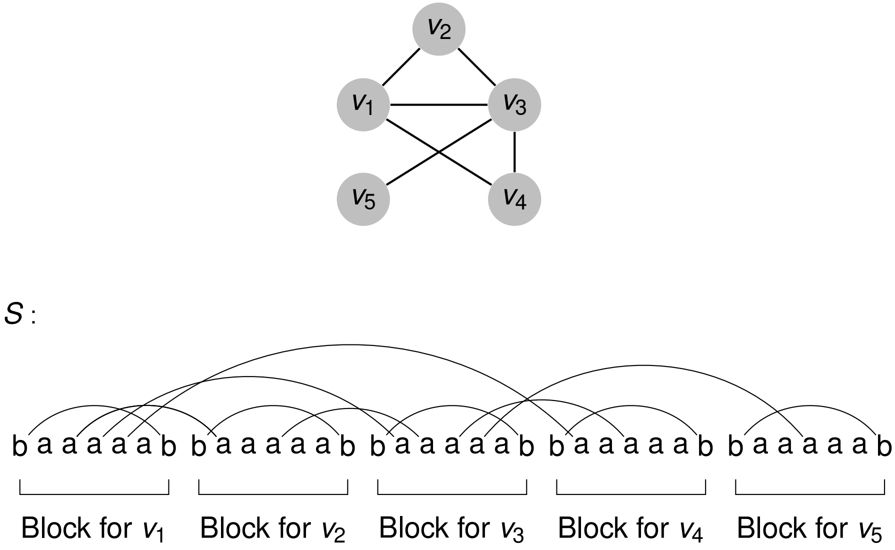
Arc-Annotated String Construction from Input-Graph



S :



Arc-Annotated String Construction from Input-Graph



Reduction construction formally

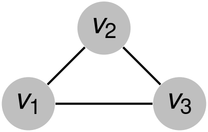
Definition

A undirected graph $G = (V, E)$, with $|V| = n$ can be encoded as an arc-annotated string $s = (s, P_s)$.

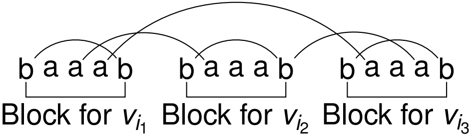
$$s = (ba^n b)^n$$

$$\begin{array}{rcl} P_s & = & \overbrace{\{((i-1)(n+2)+j+1,(j-1)(n+2)+i+1) | \{v_i, v_j\} \in E)\}}^{\text{arcs encoding edges}} \\ & \cup & \underbrace{\{((i-1)(n+2)+1, i(n+2)) | i \in \{1,\ldots,n\}\}}_{\text{arcs between two } b\text{'s of a block}} \end{array}$$

Analog: Construction of the Clique



T :

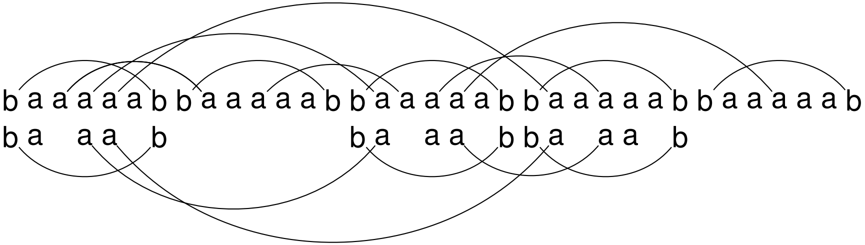


Note that $|T| = k \cdot (k+2)$, where k is the size of the clique.

Input for DECLAPCS(CROSSING,CROSSING)

Is there an arc-preserving common subsequence of size $|T|$?

S :



T :

Proof (I): Polynominal Time Reduction

Lemma

The input $(S, T, |T|)$ *to* DECLAPCS(CROSSING,CROSSING) *from* (G, k) *can be performed in polynomial time.*

- S can be directly constructed from G and has quadratic length in the number of vertices.
- A fully connected graph G_T of size k can be constructed in polynomial-time.
- Analogously to S, now also T and $|T|$ can be constructed in polynomial time by constructing a fully connected graph G_T. □

Proof (II): Correctness "$\Rightarrow$"

Lemma

Existence of a clique of size k in G implies existence of an arc-preserving common subsequence of S and T of size $|T|$.

- Let $\{v_{i_1}, \ldots, v_{i_k}\}$ be a clique of size k in the input graph.
- We can align k blocks of S to the k blocks of T.
- In each block again k symbols are matched to symbols at positions $i_1, \ldots, i_k$ in the block of S.
- Arcs between two b's are matched since we always map complete blocks to complete blocks
- $v_{i_1}, \ldots, v_{i_k}$ are vertices of a clique, thus their corresponding arcs between a 's are spanned by a arcs.

□

Proof (III): Correctness "$\Leftarrow$"

Lemma

Existence of an arc-preserving common subsequence of S and T of size $|T|$ implies a clique of size k in G.

- $|T| = k \cdot (k+2)$.
- Due to arcs over b framing a block only blocks can be mapped to blocks.
- T represents a clique of size k and blocks are constructed the same way as in S.
- Thus $i_1, \ldots, i_k$ blocks that are matched from T to S $\Rightarrow \{v_{i_1}, \ldots, v_{i_k}\}$ is a clique of size k.

□

NP-hardness of LAPCS(NESTED,NESTED)

Theorem

LAPCS(NESTED,NESTED) *is an NP-hard optimization problem.*

- Proof [Lin et al., 2002] not presented here due to many preliminaries.
- **Idea:** Reduction to variant of Maximum Independent Set (cubic planar graph) using several graph transformations with book embedding.

Complexity Results Overview for LAPCS Classes

	PLAIN	CHAIN	NESTED	CROSSING	UNLIMITED
UNLIMITED	NP-hard				
CROSSING	NP-hard				
NESTED	$\mathcal{O}(nm^3)$		NP-hard		
CHAIN	$\mathcal{O}(nm)$				
PLAIN	$\mathcal{O}(nm)$				

Table: Complexity Results for LAPCS(LEVEL1,LEVEL2)

Due to hardness results: LAPCS approximation algorithms.

2-Approximation Algorithm for LAPCS(CROSSING,CROSSING)

Idea: Use Longest Common Subsequence without arcs as a starting point and remove arc-conflicting parts successively.

2-Approximation Algorithm for LAPCS(CROSSING,CROSSING)

Input: Two arc-annotated strings $S = (s, P_s)$ and $T = (t, P_t)$ with $S, T \in$ CROSSING.

1. Determine longest common subsequence w of s and t. Let φ a mapping consistent to w.
2. Construct the conflict-graph G_φ from φ.
3. For each connected component in G_φ delete every second vertex.
4. From the resulting graph $G_{\varphi'}$ construct output string w'

Construction of the Conflict-Graph

Definition (Conflict-Graph)

Given a mapping φ that is consistent with by the longest common subsequence w of s and t.
$G_\varphi = (V, E)$

- $V = \{\langle i, j\rangle \,|\, \langle i, j\rangle \in \varphi\}$
- $E = \{\{\langle i_1, j_1\rangle, \langle i_2, j_2\rangle\}|$ **either** $(i_1, i_2) \in P_s$ or $(j_1, j_2) \in P_t\}$

Note: G_φ describes position pairs that are not arc-preserving.

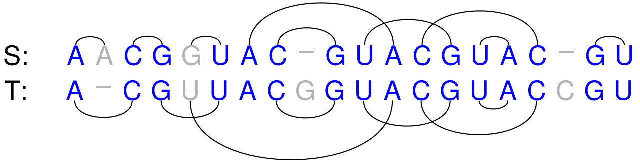
Conflict-Graph – Example

$$\varphi = \{\langle 1,1\rangle, \langle 3,2\rangle, \langle 4,3\rangle, \langle 6,5\rangle, \langle 7,6\rangle, \langle 8,7\rangle, \langle 9,9\rangle, \langle 10,10\rangle, \langle 11,11\rangle, \langle 12,12\rangle, \langle 13,13\rangle, \langle 14,14\rangle, \langle 15,15\rangle, \langle 16,16\rangle, \langle 17,18\rangle, \langle 18,19\rangle\}$$

S: A A C G G U A C – G U A C G U A C – G U
T: A – C G U U A C G G U A C G U A C C G U

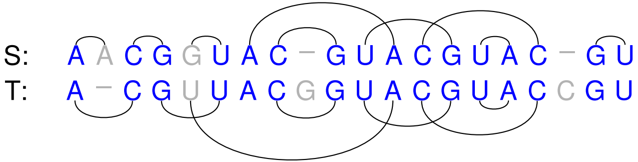
Conflict-Graph – Example

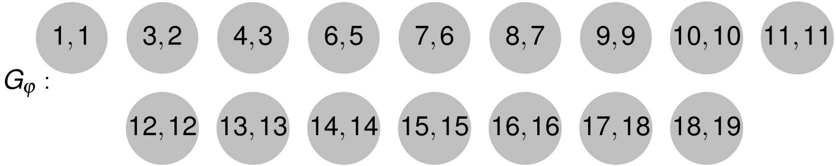
$$\varphi = \{\langle 1,1\rangle, \langle 3,2\rangle, \langle 4,3\rangle, \langle 6,5\rangle, \langle 7,6\rangle, \langle 8,7\rangle, \langle 9,9\rangle, \langle 10,10\rangle, \langle 11,11\rangle, \langle 12,12\rangle, \langle 13,13\rangle, \langle 14,14\rangle, \langle 15,15\rangle, \langle 16,16\rangle, \langle 17,18\rangle, \langle 18,19\rangle\}$$



Conflict-Graph – Example

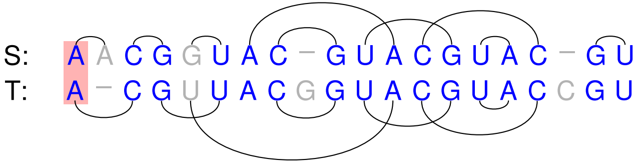
$$\varphi = \{\langle 1,1\rangle, \langle 3,2\rangle, \langle 4,3\rangle, \langle 6,5\rangle, \langle 7,6\rangle, \langle 8,7\rangle, \langle 9,9\rangle, \langle 10,10\rangle, \langle 11,11\rangle, \langle 12,12\rangle, \langle 13,13\rangle, \langle 14,14\rangle, \langle 15,15\rangle, \langle 16,16\rangle, \langle 17,18\rangle, \langle 18,19\rangle\}$$





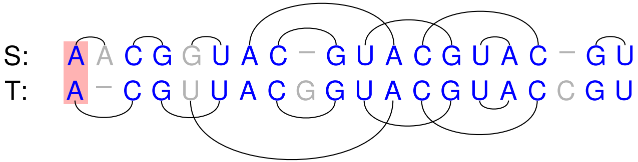
Conflict-Graph – Example

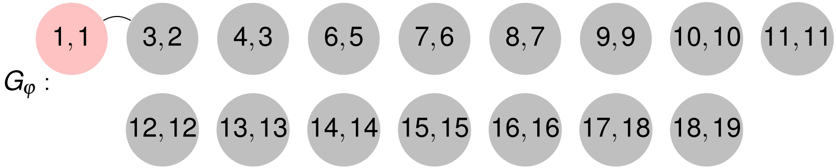
$$\varphi = \{\langle 1,1\rangle, \langle 3,2\rangle, \langle 4,3\rangle, \langle 6,5\rangle, \langle 7,6\rangle, \langle 8,7\rangle, \langle 9,9\rangle, \langle 10,10\rangle, \langle 11,11\rangle, \langle 12,12\rangle, \langle 13,13\rangle, \langle 14,14\rangle, \langle 15,15\rangle, \langle 16,16\rangle, \langle 17,18\rangle, \langle 18,19\rangle\}$$



Conflict-Graph – Example

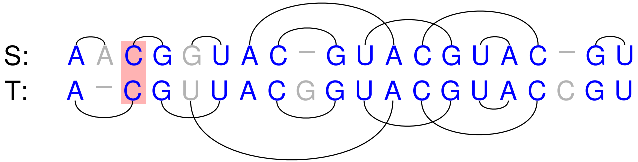
$$\varphi = \{\langle 1,1\rangle, \langle 3,2\rangle, \langle 4,3\rangle, \langle 6,5\rangle, \langle 7,6\rangle, \langle 8,7\rangle, \langle 9,9\rangle, \langle 10,10\rangle, \langle 11,11\rangle, \langle 12,12\rangle, \langle 13,13\rangle, \langle 14,14\rangle, \langle 15,15\rangle, \langle 16,16\rangle, \langle 17,18\rangle, \langle 18,19\rangle\}$$

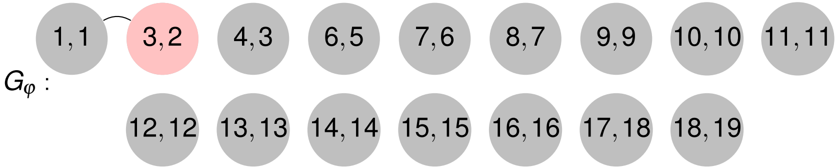




Conflict-Graph – Example

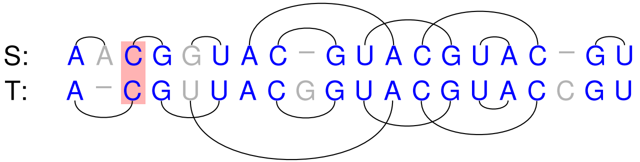
$$\varphi = \{\langle 1,1\rangle, \langle 3,2\rangle, \langle 4,3\rangle, \langle 6,5\rangle, \langle 7,6\rangle, \langle 8,7\rangle, \langle 9,9\rangle, \langle 10,10\rangle, \langle 11,11\rangle, \langle 12,12\rangle, \langle 13,13\rangle, \langle 14,14\rangle, \langle 15,15\rangle, \langle 16,16\rangle, \langle 17,18\rangle, \langle 18,19\rangle\}$$

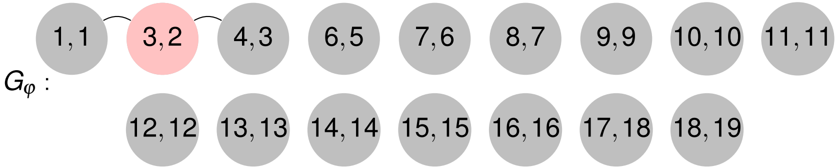




Conflict-Graph – Example

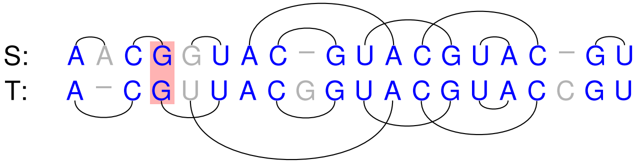
$$\varphi = \{\langle 1,1\rangle, \langle 3,2\rangle, \langle 4,3\rangle, \langle 6,5\rangle, \langle 7,6\rangle, \langle 8,7\rangle, \langle 9,9\rangle, \langle 10,10\rangle, \langle 11,11\rangle, \langle 12,12\rangle, \langle 13,13\rangle, \langle 14,14\rangle, \langle 15,15\rangle, \langle 16,16\rangle, \langle 17,18\rangle, \langle 18,19\rangle\}$$

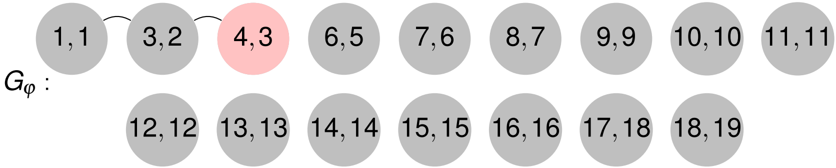




Conflict-Graph – Example

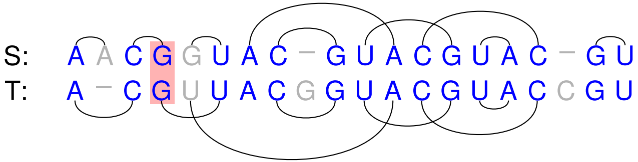
$$\varphi = \{\langle 1,1\rangle, \langle 3,2\rangle, \langle 4,3\rangle, \langle 6,5\rangle, \langle 7,6\rangle, \langle 8,7\rangle, \langle 9,9\rangle, \langle 10,10\rangle, \langle 11,11\rangle, \langle 12,12\rangle, \langle 13,13\rangle, \langle 14,14\rangle, \langle 15,15\rangle, \langle 16,16\rangle, \langle 17,18\rangle, \langle 18,19\rangle\}$$





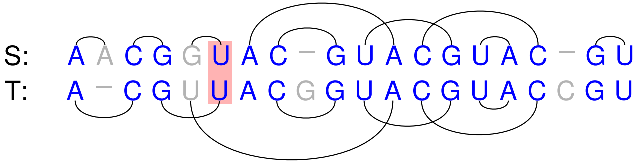
Conflict-Graph – Example

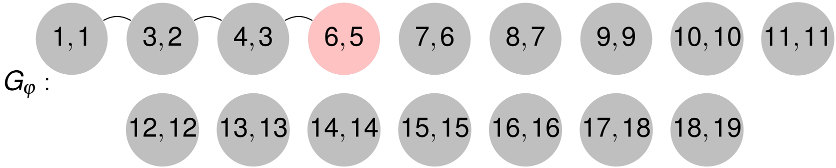
$$\varphi = \{\langle 1,1\rangle, \langle 3,2\rangle, \langle 4,3\rangle, \langle 6,5\rangle, \langle 7,6\rangle, \langle 8,7\rangle, \langle 9,9\rangle, \langle 10,10\rangle, \langle 11,11\rangle,$$
$$\langle 12,12\rangle, \langle 13,13\rangle, \langle 14,14\rangle, \langle 15,15\rangle, \langle 16,16\rangle, \langle 17,18\rangle, \langle 18,19\rangle\}$$



Conflict-Graph – Example

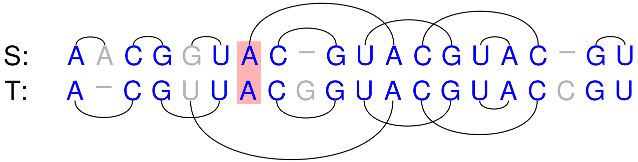
$$\varphi = \{\langle 1,1\rangle, \langle 3,2\rangle, \langle 4,3\rangle, \langle 6,5\rangle, \langle 7,6\rangle, \langle 8,7\rangle, \langle 9,9\rangle, \langle 10,10\rangle, \langle 11,11\rangle, \langle 12,12\rangle, \langle 13,13\rangle, \langle 14,14\rangle, \langle 15,15\rangle, \langle 16,16\rangle, \langle 17,18\rangle, \langle 18,19\rangle\}$$



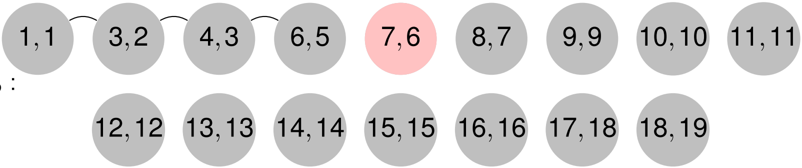


Conflict-Graph – Example

$$\varphi = \{\langle 1,1\rangle, \langle 3,2\rangle, \langle 4,3\rangle, \langle 6,5\rangle, \langle 7,6\rangle, \langle 8,7\rangle, \langle 9,9\rangle, \langle 10,10\rangle, \langle 11,11\rangle, \langle 12,12\rangle, \langle 13,13\rangle, \langle 14,14\rangle, \langle 15,15\rangle, \langle 16,16\rangle, \langle 17,18\rangle, \langle 18,19\rangle\}$$

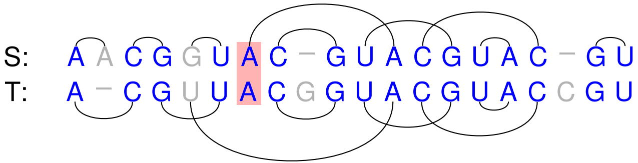


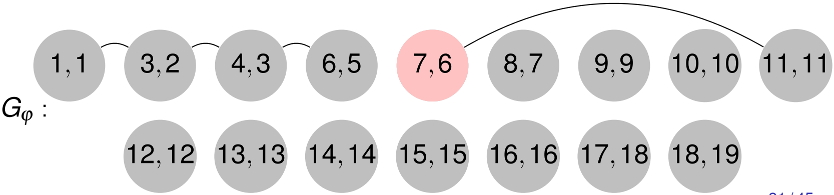
G_φ :



Conflict-Graph – Example

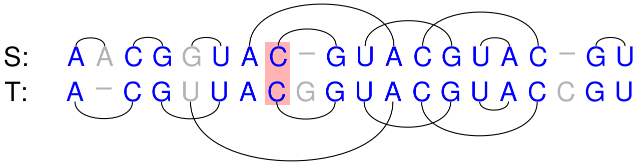
$$\varphi = \{\langle 1,1\rangle, \langle 3,2\rangle, \langle 4,3\rangle, \langle 6,5\rangle, \langle 7,6\rangle, \langle 8,7\rangle, \langle 9,9\rangle, \langle 10,10\rangle, \langle 11,11\rangle, \langle 12,12\rangle, \langle 13,13\rangle, \langle 14,14\rangle, \langle 15,15\rangle, \langle 16,16\rangle, \langle 17,18\rangle, \langle 18,19\rangle\}$$

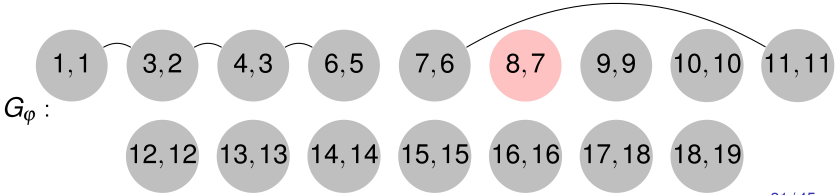




Conflict-Graph – Example

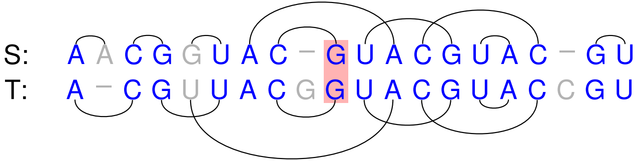
$$\varphi = \{\langle 1,1\rangle, \langle 3,2\rangle, \langle 4,3\rangle, \langle 6,5\rangle, \langle 7,6\rangle, \langle 8,7\rangle, \langle 9,9\rangle, \langle 10,10\rangle, \langle 11,11\rangle, \langle 12,12\rangle, \langle 13,13\rangle, \langle 14,14\rangle, \langle 15,15\rangle, \langle 16,16\rangle, \langle 17,18\rangle, \langle 18,19\rangle\}$$

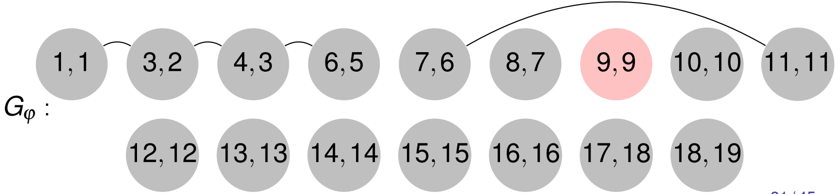




Conflict-Graph – Example

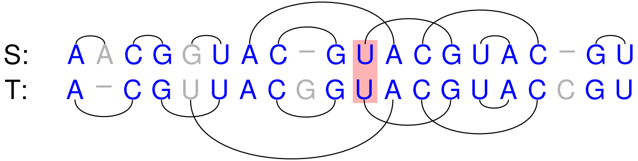
$$\varphi = \{\langle 1,1\rangle, \langle 3,2\rangle, \langle 4,3\rangle, \langle 6,5\rangle, \langle 7,6\rangle, \langle 8,7\rangle, \langle 9,9\rangle, \langle 10,10\rangle, \langle 11,11\rangle, \langle 12,12\rangle, \langle 13,13\rangle, \langle 14,14\rangle, \langle 15,15\rangle, \langle 16,16\rangle, \langle 17,18\rangle, \langle 18,19\rangle\}$$

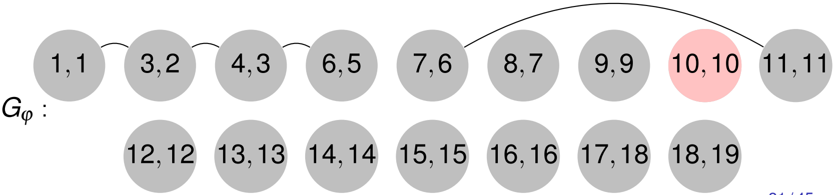




Conflict-Graph – Example

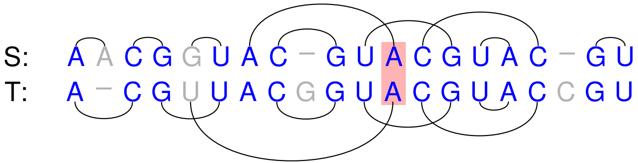
$$\varphi = \{\langle 1,1\rangle, \langle 3,2\rangle, \langle 4,3\rangle, \langle 6,5\rangle, \langle 7,6\rangle, \langle 8,7\rangle, \langle 9,9\rangle, \langle 10,10\rangle, \langle 11,11\rangle, \langle 12,12\rangle, \langle 13,13\rangle, \langle 14,14\rangle, \langle 15,15\rangle, \langle 16,16\rangle, \langle 17,18\rangle, \langle 18,19\rangle\}$$

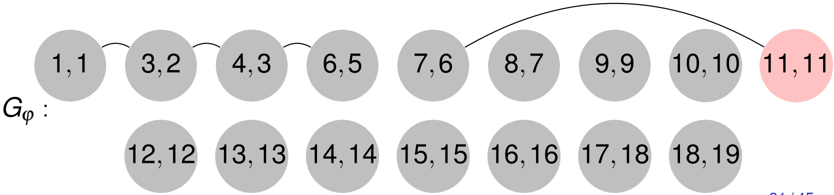




Conflict-Graph – Example

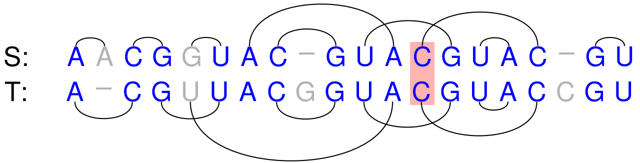
$$\varphi = \{\langle 1,1\rangle, \langle 3,2\rangle, \langle 4,3\rangle, \langle 6,5\rangle, \langle 7,6\rangle, \langle 8,7\rangle, \langle 9,9\rangle, \langle 10,10\rangle, \langle 11,11\rangle, \langle 12,12\rangle, \langle 13,13\rangle, \langle 14,14\rangle, \langle 15,15\rangle, \langle 16,16\rangle, \langle 17,18\rangle, \langle 18,19\rangle\}$$

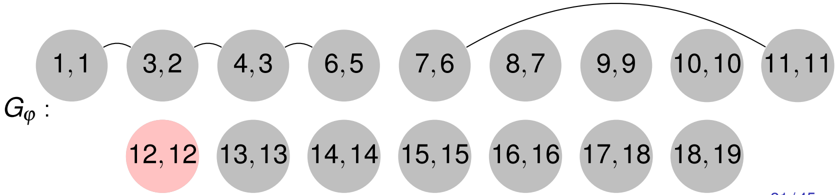




Conflict-Graph – Example

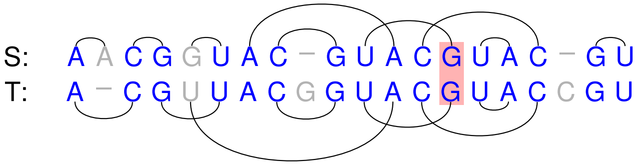
$$\varphi = \{\langle 1,1\rangle, \langle 3,2\rangle, \langle 4,3\rangle, \langle 6,5\rangle, \langle 7,6\rangle, \langle 8,7\rangle, \langle 9,9\rangle, \langle 10,10\rangle, \langle 11,11\rangle, \langle 12,12\rangle, \langle 13,13\rangle, \langle 14,14\rangle, \langle 15,15\rangle, \langle 16,16\rangle, \langle 17,18\rangle, \langle 18,19\rangle\}$$

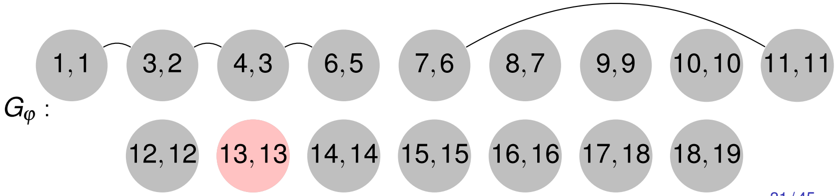




Conflict-Graph – Example

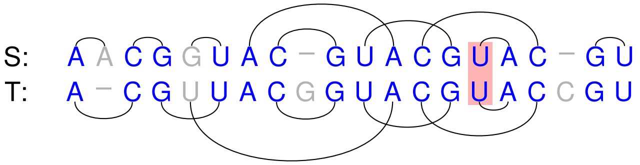
$$\varphi = \{\langle 1,1\rangle, \langle 3,2\rangle, \langle 4,3\rangle, \langle 6,5\rangle, \langle 7,6\rangle, \langle 8,7\rangle, \langle 9,9\rangle, \langle 10,10\rangle, \langle 11,11\rangle, \langle 12,12\rangle, \langle 13,13\rangle, \langle 14,14\rangle, \langle 15,15\rangle, \langle 16,16\rangle, \langle 17,18\rangle, \langle 18,19\rangle\}$$



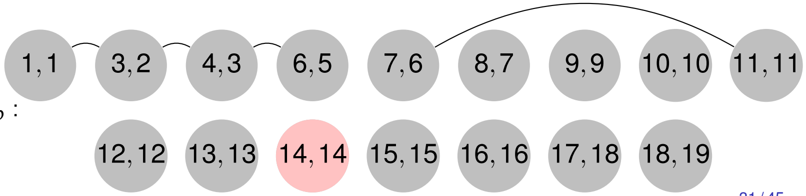


Conflict-Graph – Example

$$\varphi = \{\langle 1,1\rangle, \langle 3,2\rangle, \langle 4,3\rangle, \langle 6,5\rangle, \langle 7,6\rangle, \langle 8,7\rangle, \langle 9,9\rangle, \langle 10,10\rangle, \langle 11,11\rangle,$$
$$\langle 12,12\rangle, \langle 13,13\rangle, \langle 14,14\rangle, \langle 15,15\rangle, \langle 16,16\rangle, \langle 17,18\rangle, \langle 18,19\rangle\}$$

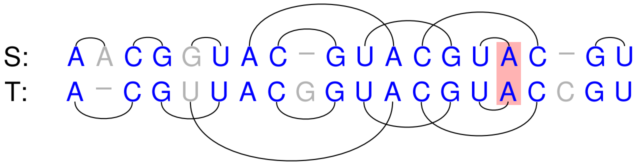


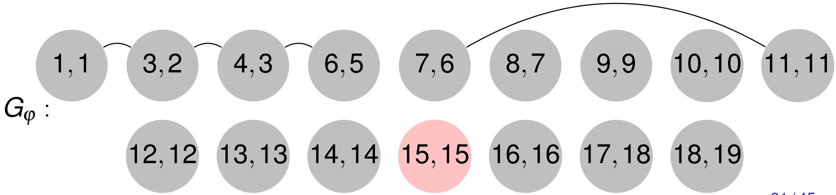
G_φ :



Conflict-Graph – Example

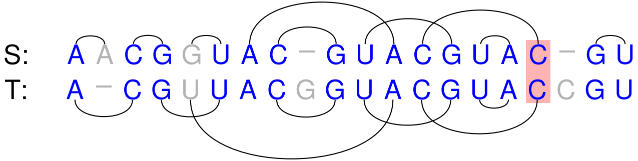
$$\varphi = \{\langle 1,1\rangle, \langle 3,2\rangle, \langle 4,3\rangle, \langle 6,5\rangle, \langle 7,6\rangle, \langle 8,7\rangle, \langle 9,9\rangle, \langle 10,10\rangle, \langle 11,11\rangle, \langle 12,12\rangle, \langle 13,13\rangle, \langle 14,14\rangle, \langle 15,15\rangle, \langle 16,16\rangle, \langle 17,18\rangle, \langle 18,19\rangle\}$$

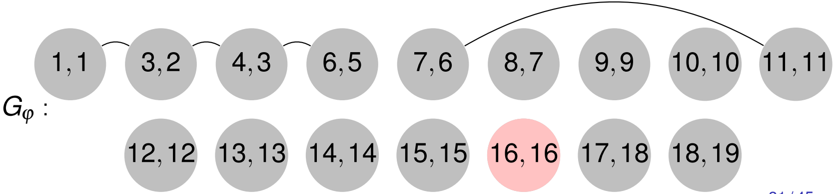




Conflict-Graph – Example

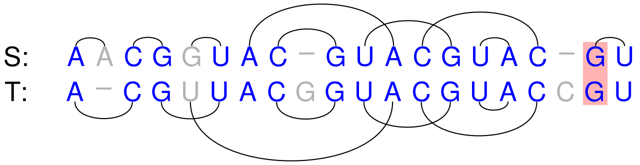
$$\varphi = \{\langle 1,1\rangle, \langle 3,2\rangle, \langle 4,3\rangle, \langle 6,5\rangle, \langle 7,6\rangle, \langle 8,7\rangle, \langle 9,9\rangle, \langle 10,10\rangle, \langle 11,11\rangle, \langle 12,12\rangle, \langle 13,13\rangle, \langle 14,14\rangle, \langle 15,15\rangle, \langle 16,16\rangle, \langle 17,18\rangle, \langle 18,19\rangle\}$$

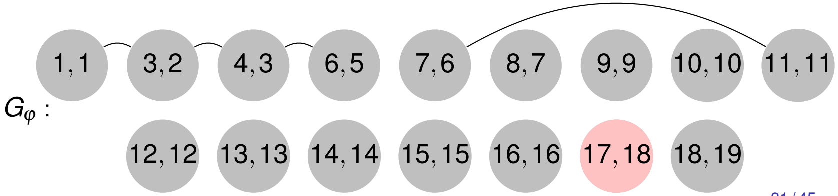




Conflict-Graph – Example

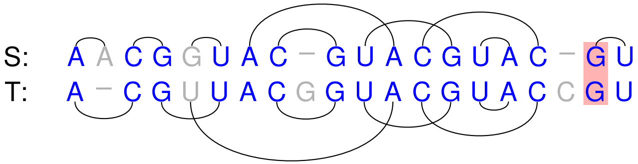
$$\varphi = \{\langle 1,1\rangle, \langle 3,2\rangle, \langle 4,3\rangle, \langle 6,5\rangle, \langle 7,6\rangle, \langle 8,7\rangle, \langle 9,9\rangle, \langle 10,10\rangle, \langle 11,11\rangle, \langle 12,12\rangle, \langle 13,13\rangle, \langle 14,14\rangle, \langle 15,15\rangle, \langle 16,16\rangle, \langle 17,18\rangle, \langle 18,19\rangle\}$$

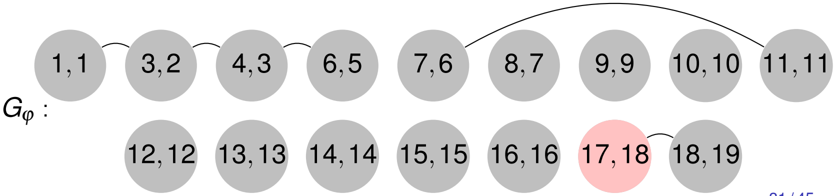




Conflict-Graph – Example

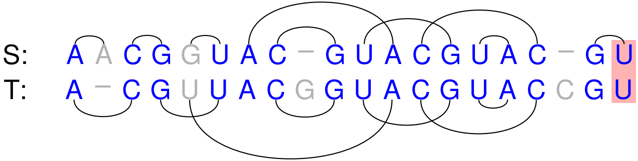
$$\varphi = \{\langle 1,1\rangle, \langle 3,2\rangle, \langle 4,3\rangle, \langle 6,5\rangle, \langle 7,6\rangle, \langle 8,7\rangle, \langle 9,9\rangle, \langle 10,10\rangle, \langle 11,11\rangle,$$
$$\langle 12,12\rangle, \langle 13,13\rangle, \langle 14,14\rangle, \langle 15,15\rangle, \langle 16,16\rangle, \langle 17,18\rangle, \langle 18,19\rangle\}$$

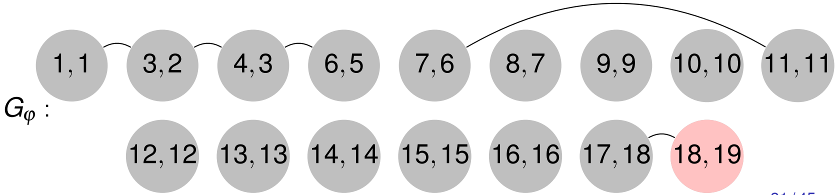




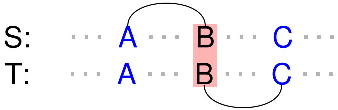
Conflict-Graph – Example

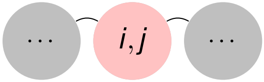
$$\varphi = \{\langle 1,1\rangle, \langle 3,2\rangle, \langle 4,3\rangle, \langle 6,5\rangle, \langle 7,6\rangle, \langle 8,7\rangle, \langle 9,9\rangle, \langle 10,10\rangle, \langle 11,11\rangle, \langle 12,12\rangle, \langle 13,13\rangle, \langle 14,14\rangle, \langle 15,15\rangle, \langle 16,16\rangle, \langle 17,18\rangle, \langle 18,19\rangle\}$$





Conflict-Graph Observation





Lemma

G_φ has at most node degree two for two arc-annotated strings $T, S \in$ CROSSING.

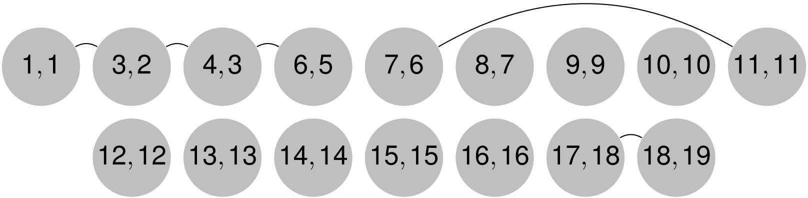
Proof.

- Since $T, S \in$ CROSSING no two arcs share a common start/endpoint.
- Incoming edge: w.l.o.g. at most one arc-mismatch for incoming edges
- Outgoing edge: analogous.

□

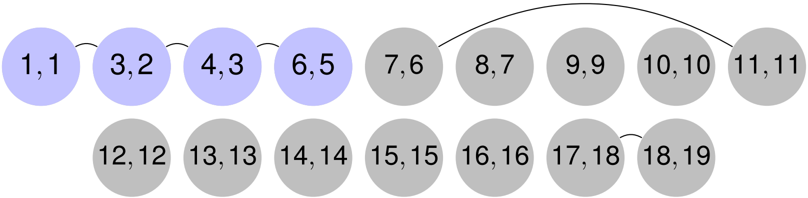
Approximation Algorithm – Step 3

For each connected component in G_φ delete every second vertex.



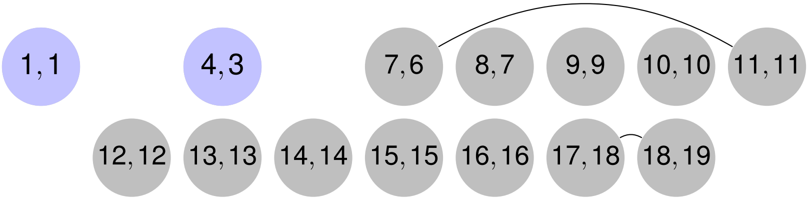
Approximation Algorithm – Step 3

For each connected component in G_φ delete every second vertex.



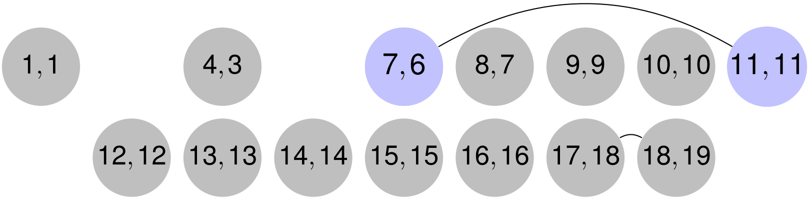
Approximation Algorithm – Step 3

For each connected component in G_φ delete every second vertex.



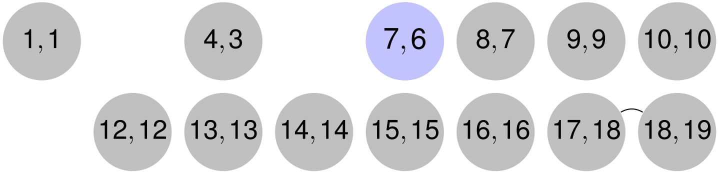
Approximation Algorithm – Step 3

For each connected component in G_φ delete every second vertex.



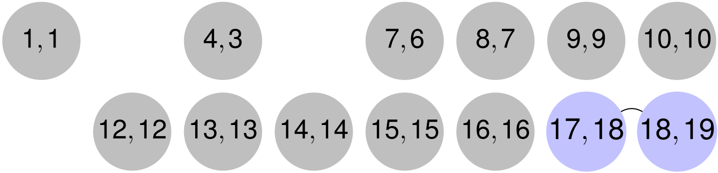
Approximation Algorithm – Step 3

For each connected component in G_φ delete every second vertex.



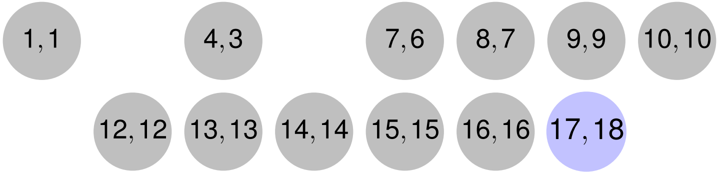
Approximation Algorithm – Step 3

For each connected component in G_φ delete every second vertex.



Approximation Algorithm – Step 3

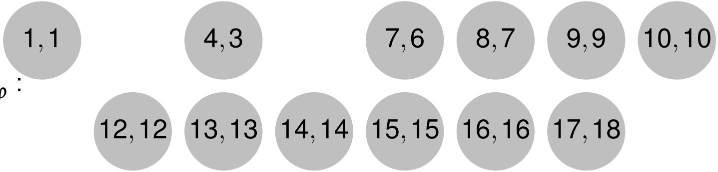
For each connected component in G_φ delete every second vertex.



Approximation Algorithm – Step 3

For each connected component in G_φ delete every second vertex.

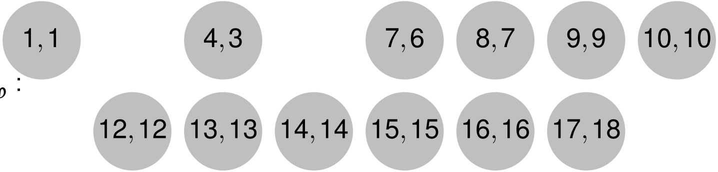
G'_φ :



Approximation Algorithm – Final Step

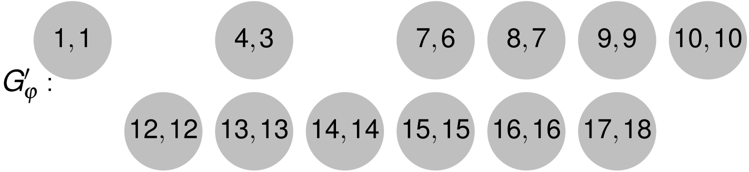
Reconstruct corresponding arc-preserving common subsequence w'.

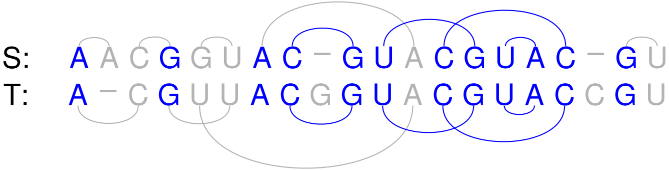
G'_φ :



Approximation Algorithm – Final Step

Reconstruct corresponding arc-preserving common subsequence w'.





Correctness Proof (I)

Theorem

The Approximation algorithm computes a feasible solution for LAPCS(CROSSING,CROSSING)*.*

Proof.

- The string w' results from removing some symbols in w and thus is still a common subsequence.
- Also, w' is arc-preserving:
  - Connected vertices in the conflict-graph G_φ denoted violating position pairs.
  - The algorithm removes all edges from the conflict graph.

□

Correctness Proof (II)

Theorem

The algorithm computes 2-approximation for LAPCS(CROSSING,CROSSING).

Proof.

Let $S = (s, P_s)$ and $T = (t, P_t)$ be two arc-annotated strings and w_{opt} be a longest arc-preserving of S and T. Let w' be the output of the approximation algorithm.

- Let w be the longest common subsequence of s and t. $|w| \geq |w_{\text{opt}}|$.
- Because we delete at most every second vertex in a path in the conflict-graph it holds that $|w'| \geq \frac{|w|}{2}$.
- Combining both inequalities leads to $|w'| \geq \frac{|w_{\text{opt}}|}{2}$.

□

Complexity Proof (I)

Theorem

The approximation algorithm requires a running time in $\mathcal{O}(n \cdot m)$, where n and m denote the length of the input strings.

Proof.

- Computation of Longest Common Subsequence: $\mathcal{O}(n \cdot m)$.
- Construction of the conflict-graph:
  - For two position pairs $\langle i_1, j_1 \rangle, \langle i_2, j_2 \rangle \in \varphi$ we need to check whether $(i_1, i_2) \in P_s$ and $(j_1, j_2) \in P_t$.
  - $|w| \leq \min(n, m)$, Thus φ contains at most $\min(n, m)$ position pairs, hence construction takes $\mathcal{O}\left(\min(n, m)^2\right) \subseteq \mathcal{O}(n \cdot m)$.

Complexity Proof (II)

Proof (Cont.)

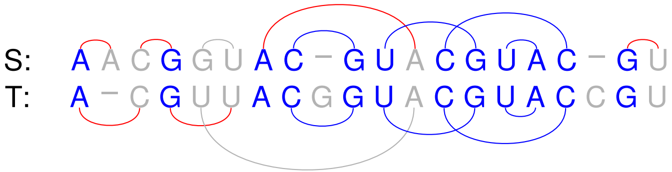
Traversal and deletion of nodes in the conflict-graph $G_\varphi = (V, E)$:

- For each node $v \in V$, we need to determine whether v is an isolated vertex, or part of a path.
  - Traverse edges starting from v.
  - Euler's handshaking lemma gives $\sum_{v \in V} \deg(v) = 2|E|$.
  - G_φ has at most node degree 2.
  - This yields: $|E| \leq \min(n, m)$.
  - The procedure requires $\mathcal{O}\left(\min(n, m)^2\right) \subseteq \mathcal{O}(n \cdot m)$.
- For each path we need to delete every second vertex: Same reasoning as above: $\mathcal{O}(n \cdot m)$.

Reconstruction of w' from $G_{\varphi'}$: $\mathcal{O}(\min(n, m)) \subseteq \mathcal{O}(n \cdot m)$. □

Discussion

- Algorithm is adjustable, other variants than global alignment can be used in the initial step.
- 2-approximation is a worst-case approximation.
- However, algorithm conflict graph ignores arcs of non-matched characters:



Exact Solution with Parametrized Complexity

Concept: "*Extract*" parameter responsible for the exponential running time. [Alber et al., 2002]

Parameters: Number of deletions k_1 and k_2 in the strings T and S, respectively.

Idea: Use recursive search tree, investigate smaller substrings, decrement k_1 and k_2 in the recursion.

Complexity: $\mathcal{O}\left(3,31^{k_1+k_2} \cdot \min(m,n)\right)$
Proof by branching-vector analysis over size the search tree.

Parametrized Complexity: Cutwidth

Concept: Again, "*Extract*" parameter responsible for the exponential running time, here: Cutwidth [Evans, 1999]

Parameters: Cutwidth, i.e. the maximum number of arcs that cross or end at any arbitrary position of the sequence.
Complexity: $\mathcal{O}(f(k) \cdot m \cdot n)$

Conclusion

- RNA secondary structures can be represented in terms of arc-annotated strings
- Distinguish between different classes of arc-annotated strings
- Similarity comparison motivates the LAPCS problem.
- Unfortunately, LAPCS is NP-hard for relevant cases.
- The LAPCS can be approximated by a 2-approximation algorithm.

Conclusion

- RNA secondary structures can be represented in terms of arc-annotated strings
- Distinguish between different classes of arc-annotated strings
- Similarity comparison motivates the LAPCS problem.
- Unfortunately, LAPCS is NP-hard for relevant cases.
- The LAPCS can be approximated by a 2-approximation algorithm.

Thank you for your attention.

References I

Alber, J., Gramm, J., Guo, J., and Niedermeier, R. (2002).
Towards optimally solving the longest common subsequenceproblem for sequences with nested arc annotations in linear time.
In Apostolico, A. and Takeda, M., editors, *Combinatorial Pattern Matching*, volume 2373 of *Lecture Notes in Computer Science*, pages 99–114. Springer Berlin Heidelberg.

Böckenhauer, H.-J. and Bongartz, D. (2007).
Algorithmic Aspects of Bioinformatics.
Springer.

References II

Evans, P. A. (1999).
Algorithms and Complexity for Annotated Sequence Analysis.
PhD thesis, Victoria, B.C., Canada, Canada.
AAINQ41369.

Jiang, T., Lin, G.-H., Ma, B., and Zhang, K. (2000).
The longest common subsequence problem for arc-annotated sequences.
In *Combinatorial Pattern Matching*, pages 154–165.
Springer.

References III

Lin, G., Chen, Z.-Z., Jiang, T., and Wen, J. (2002).
The longest common subsequence problem for sequences with nested arc annotations.
Journal of Computer and System Sciences, 65(3):465 – 480.
Special Issue on Computational Biology 2002.