Pattern matching and common structure inference in RNA (secondary) structures

Stéphane Vialette
Stephane.Vialette@lri.fr
Laboratoire de Recherche en Informatique (LRI)
bât.490, Univ. Paris-Sud XI, 91405 Orsay cedex, France
http://www.lri.fr/~vialette

September 19, 2007, Wuhan, China
Outline
RNA secondary structures

**Definition**

RNA molecules fold back on themselves via Watson-Crick base paring between the bases (A with U and G with C or U) leading to double-stranded helices interrupted by single-stranded regions in internal loops or hairpin loops.
RNA secondary structures

Possible representations

- Linear representation
RNA secondary structures

Possible representations

- Bracket representation

```
((((((......))))(((......))))][(...........))][(...........)]...
```

Stéphane Vialette

Pattern matching and common structure inference in RNA (secondary) structures
RNA secondary structures

Possible representations

- Tree representation
RNA secondary structures

Possible representations

- Circle representation
RNA secondary structures

Possible representations

- Mountain representation

![Mountain representation of RNA secondary structure](image)
Definition

In the next level of organization, the tertiary structure, the secondary structure elements are associated through numerous contacts, specific hydrogen bonds via the formation of a small number of additional Watson-Crick pairs and/or unusual pairs involving hairpin loops or internal bulges.
RNA tertiary structure

Stéphane Vialette

Pattern matching and common structure inference in RNA (secondary) structures

David W. Staple et Samuel E. Butcher,
*Pseudoknots: RNA Structures with Diverse Functions*,
Fact

Most problems cannot be solved to optimality in reasonable \textit{(polynomial)} running time.

Most problems are \textbf{NP}-complete.
The class $\text{NP}$ (Non-deterministic Polynomial)

The class $\text{NP}$ is composed of all decision problems for which answers can be checked by an algorithm whose running time is polynomial in the size of the input.

Note that this doesn’t require or imply that an answer can be found quickly, only that any claimed solution can be verified quickly.
A problem $\Pi$ is $\textbf{NP}$-hard if an algorithm for solving it can be translated into one for solving any problem in $\textbf{NP}$ (non-deterministic polynomial time).

$\textbf{NP}$-hard therefore means ”at least as hard as any problem in $\textbf{NP}$”, although $\Pi$ might, in fact, be harder.
A problem $\Pi$ is $\text{NP}$-complete if

- $\Pi$ is in $\text{NP}$ (verifiable in non-deterministic polynomial time), and
- $\Pi$ is $\text{NP}$-hard (any problem in $\text{NP}$ can be translated into this problem).
A crash course in algorithmic complexity theory

Stéphane Vialette

Pattern matching and common structure inference in RNA (secondary) structures
Proving a problem $\Pi$ to be NP-complete

1. Prove that problem $\Pi$ is in $\text{NP}$.

2. Choose any known $\text{NP}$-complete problem $\Pi'$ and prove that $\Pi'$ reduces to $\Pi$. 

Stéphane Vialette

Pattern matching and common structure inference in RNA (secondary) structures
Coping with hardness

OK. So what is the next step?

- Approximation algorithms.
- Parameterized algorithms.
- Heuristic algorithms.
- …

The choice of the direction to follow is application-dependent.
Approximation algorithms

**definition**

An algorithm to solve an optimization problem that runs in polynomial-time in the length of the input and outputs a solution that is guaranteed to be close to the optimal solution.

"Close" has some well-defined sense called the performance guarantee.
Parameterized algorithms

**definition**

An algorithm to solve an optimization problem that runs in polynomial-time in the length of the input but in exponential-time in a parameter, and outputs a solution that is guaranteed to be the optimal solution.

The choice of a parameter makes parameterized algorithms well-suited for practical problems.
Heuristic algorithms

**definition**

An algorithm that usually, but not always, works or that gives nearly the right answer.

The running time of the algorithm might be prohibitive . . . but not always.
More or less a fact

Most RNA structure problems cannot be solved to optimality in reasonable (polynomial) running time for crossing structures, i.e., pseudo-knotted structures.

- Dynamic programming.
- Dynamic programming can deal with reasonable pseudo-knotted structures.
- Approximation algorithms.
- Parameterized algorithms.
New (not so simple) RNA representations

- Sets of 2-intervals

- Linear graphs

- Arc-annotated sequences
Outline
**General problem**

**Definition**

Given two (secondary) structures $S$ and $T$, decide whether or not $S$ “occurs” in $T$.

- Parsing RNA structure databases.
- Comparing RNA structures.
- The exact problem depends on
  - the structure of $S$ and $T$, and
  - what does it mean for a structure to occur in another one?
The **Arc-Preserving Subsequence** problem

**Definition**

Given two arc-annotated sequences $S$ and $T$, decide whether or not $S$ occurs in $T$ as an arc-preserving subsequence.

**Example**

```
 a a a g g t t c a a c u
  a g t c c
```

Stéphane Vialette

Pattern matching and common structure inference in RNA (secondary) structures
The **Arc-Preserving Subsequence** problem

**Definition**

Given two arc-annotated sequences $S$ and $T$, decide whether or not $S$ occurs in $T$ as an arc-preserving subsequence.

**Example**

![Diagram illustrating an example of the Arc-Preserving Subsequence problem]
The **Arc-Preserving Subsequence** problem

Unlimited

Crossing

Nested

Chain

Plain
## The Arc-Preserving Subsequence problem

### Complexity issues

<table>
<thead>
<tr>
<th></th>
<th>APS</th>
<th>CROSSING</th>
<th>NESTED</th>
<th>CHAIN</th>
<th>PLAIN</th>
</tr>
</thead>
<tbody>
<tr>
<td>CROSSING</td>
<td>NP-complete</td>
<td>NP-complete</td>
<td>NP-complete</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NESTED</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$O(nm)$</td>
</tr>
<tr>
<td>CHAIN</td>
<td></td>
<td></td>
<td></td>
<td>$O(nm)$</td>
<td>$O(n + m)$</td>
</tr>
</tbody>
</table>
Outline
General problem

Definition

Given $n$ (secondary) structures $S_1, S_2, \ldots, S_n$, find the largest (secondary) structure $T$ that occurs in each input structure.

- Parsing RNA structure databases.
- Comparing RNA structures.
- The exact problem depends on
  - $n$,
  - the input structures and the structure of $T$, and
  - what does it mean for a structure to occur in another one?
Common structure inference

Remarks

Variants of the problem exist for
- 2-interval sets,
- linear graphs, and
- arc-annotated sequences.

The choice of the structure to focus in here is (mostly) algorithmic-dependent: The simpler the structure, the simpler the algorithmic problem.
The Longest Arc-Preserving Common Subsequence problem

**Definition**

Given $n$ arc-annotated sequences $S_1, S_2, \ldots, S_n$, find the largest arc-annotated sequence that occurs in each $S_i$, $1 \leq i \leq n$, as an arc-preserving subsequence.

**Remarks**

- The complexity of the problem depends on the structure of each input arc-annotated sequence.
- Bad news: the problem is hard to solve to optimality even the structures are crossing-free.
The **LONGEST ARC-PRESERVING COMMON SUBSEQUENCE** problem

**Example**

\[
S_1 \quad a \ a \ a \ g \ g \ t \ t \ c \ a \ c \ g \ u \\
S_2 \quad a \ a \ g \ t \ c \ c \ g \\
S_3 \quad a \ g \ a \ a \ a \ c \ t \ c \ g \ c \ g
\]
The **LONGEST ARC-PRESERVING COMMON SUBSEQUENCE** problem

**Example**

\[ S_1 \quad \text{aaaggtttcaca} \quad \text{g} \quad \text{u} \]

\[ S_2 \quad \text{aagttccg} \]

\[ S_3 \quad \text{agaatcgcg} \]
The **LONGEST ARC-PRESERVING COMMON SUBSEQUENCE** problem

<table>
<thead>
<tr>
<th></th>
<th>Chain</th>
<th>Nest</th>
<th>Cross</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chain</td>
<td>Chain</td>
<td>Nest</td>
<td>Chain</td>
</tr>
<tr>
<td>Nest</td>
<td>Nest</td>
<td>Cross</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Chain</th>
<th>Nest</th>
<th>Cross</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>EDIT</strong></td>
<td>$O(nm)$</td>
<td>$O(nm^3)$</td>
<td>NPC</td>
</tr>
<tr>
<td>LAPCS</td>
<td>$O(nm)$</td>
<td>$O(nm^3)$</td>
<td>NPC</td>
</tr>
<tr>
<td>MLG</td>
<td>$O(nm)$</td>
<td>$O(n^2m)$</td>
<td>$O(n^2m^2)$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Unlim</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chain</td>
<td>Nest</td>
</tr>
<tr>
<td>Cross</td>
<td>Unlim</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Unlim</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>EDIT</strong></td>
<td>APX-hard</td>
</tr>
<tr>
<td>LAPCS</td>
<td>NPC</td>
</tr>
<tr>
<td>MLG</td>
<td>$O(n^4 \log^3 n)$</td>
</tr>
</tbody>
</table>

Stéphane Vialette

Pattern matching and common structure inference in RNA (secondary) structures
What is an occurrence of a pattern $T$ in a linear graph $S$?
What is an occurrence of a pattern $T$ in a linear graph $S$?
Occurrences in linear graphs

What is an occurrence of a pattern $T$ in a linear graph $S$?
Common structure inference

$S_1$

$S_2$

$S_3$
Common structure inference

$S_1$

$S_2$

$S_3$
The problem of finding the maximum size common secondary structure in a set of \( n \) linear graphs is solvable in \( O(m^{2k} \log^{k-2} m^k \log \log m^k) \) time, where \( m \) is the maximum size of an input linear graph.

Remarks

- The result holds true even if the input structures are crossing.
- We still don’t have an efficient implementation of the algorithm.
Outline

Stéphane Vialette

Pattern matching and common structure inference in RNA (secondary) structures
Tertiary but planar

Domain I: Specificity Domain

Domain II: Catalytic Domain

Stéphane Vialette

Pattern matching and common structure inference in RNA (secondary) structures
RNA bi-secondary structures

- Include most real practical RNA structures.
- Several heuristic algorithms so exist.
- Well-known combinatorial structure in graph theory (2-pages linear graphs).
- Very little is known on algorithic issues.