Algorithmic Aspects of RNA Secondary Structures

Stéphane Vialette

CNRS & LIGM, Université Paris-Est Marne-la-Vallée, France

2014-1015
Plan

1. Introduction

2. RNA secondary structure prediction

3. Pseudoknot prediction and alternate models
Central dogma of molecular biology

- DNA Polymerase (replication: DNA → DNA)
- RNA Polymerase (transcription: DNA → RNA)
- Ribosome (translation: RNA → Protein)
Central dogma of molecular biology

- The central dogma of molecular biology deals with the detailed residue-by-residue transfer of sequential information. It states that such information cannot be transferred back from protein to either protein or nucleic acid.

- This has also been described as DNA makes RNA makes protein. However, this simplification does not make it clear that the central dogma as stated by Crick does not preclude the reverse flow of information.
Transcription

- **Transcription** is the process by which the information contained in a section of DNA is transferred to a newly assembled piece of messenger RNA (mRNA).

- It is facilitated by RNA polymerase and transcription factors.

- In eukaryotic cells the primary transcript (pre-mRNA) must be processed further in order to ensure translation.

- This normally includes a 5′ cap, a poly-A tail and splicing.

- Alternative splicing can also occur, which contributes to the diversity of proteins any single mRNA can produce.
Transcription

DNA is rewound
Coding strand
RNA polymerase
DNA is unwound
Template strand
Matching nucleotide is added
RNA strand created
RNA DNA hybrid region
NTPs
Alternative splicing
Translation

- Eventually, the mature mRNA finds its way to a ribosome, where it is translated.

- In prokaryotic cells, which have no nuclear compartment, the process of transcription and translation may be linked together. In eukaryotic cells, the site of transcription (the cell nucleus) is usually separated from the site of translation (the cytoplasm), so the mRNA must be transported out of the nucleus into the cytoplasm, where it can be bound by ribosomes.

- The mRNA is read by the ribosome as triplet codons, usually beginning with an AUG (adenine–uracil–guanine), or initiator methionine codon downstream of the ribosome binding site.

- Translation ends with a UAA, UGA, or UAG stop codon.
Translation
Simultaneous translation and transcription
Base pairing

- In molecular biology, two nucleotides on opposite complementary DNA or RNA strands that are connected via hydrogen bonds are called a base pair (often abbreviated bp).

- In the canonical Watson-Crick base pairing, adenine (A) forms a base pair with thymine (T), and guanine (G) forms one with cytosine (C) in DNA.

- In RNA, thymine is replaced by uracil (U).

- Alternate hydrogen bonding patterns, such as the wobble base pair and Hoogsteen base pair, also occur—particularly in RNA—giving rise to complex and functional tertiary structures.

- Importantly, pairing is the mechanism by which codons on messenger RNA molecules are recognized by anticodons on transfer RNA during protein translation.
Base pairing

Left, an *AT* base pair demonstrating two intermolecular hydrogen bonds; Right, a *GC* base pair demonstrating three intermolecular hydrogen bonds.
Base pairing

Adenine
Thymine
5' end
3' end
Phosphate-deoxyribose backbone
Guanine
Cytosine
5' end
miRNA

protein-coding gene

transcription

coding mRNA

3'UTR

miRNA

no translation

CANCER

defoy mRNA

tumor suppressor protein

translation

NO CANCER
Non-coding RNA
Structural conformations of biomolecules

- **Primary Structure**: sequence of monomers (ATCGAGATC…)
- **Secondary Structure**: 2D-fold, defined by hydrogen bonds
- **Tertiary Structure**: 3D-fold
- **Quarternary Structure**: complex arrangement of multiple folded molecules

RNA tertiary structure
The major role of tRNA is to translate mRNA sequence into amino acid sequence. A tRNA molecule consists of $70 - 80$ nucleotides.
RNA tertiary structure

A hairpin loop from a pre-mRNA. Highlighted are the nucleobases (green) and the ribose-phosphate backbone (blue). Note that this is a single strand of RNA that folds back upon itself.
RNA tertiary structure

Three-dimensional representation of the 50S ribosomal subunit. RNA is in ochre, protein in blue. The active site is in the middle (red).
Prediction of secondary structure: FASTA format

- **FASTA format** is a text-based format for representing either nucleotide sequences or peptide sequences, in which nucleotides or amino acids are represented using single-letter codes.
- The format also allows for sequence names and comments to precede the sequences.
- The format originates from the FASTA software package, but has now become a standard in the field of bioinformatics.
- The simplicity of FASTA format makes it easy to manipulate and parse sequences using text-processing tools and scripting languages like Python, Ruby, and Perl.

>MAMseq000312 Euarctos americanus mitochondrial transfer RNA-Pro and transfer RNA-Thr, 3′ ends.
aagactcaaggaagaagcaacagccccactattaacacccaaagctaatgttctatttaactattccttg
>MAMseq000315 Nasua narica mitochondrial transfer RNA-Pro and transfer RNA-Thr, 3′ ends.
aagactttcaaggaagaagcaacagccccactattaacacccaaagctaatgttctatttaactattccttg
>MAMseq000316 Procyon lotor mitochondrial transfer RNA-Pro and transfer RNA-Thr, 3′ ends.
aagactttcaaggaagaagcaacagccccactattaacacccaaagctaatgttctatttaactattccttg
>MAMseq000318 Potos flavus mitochondrial transfer RNA-Pro and transfer RNA-Thr, 3′ ends.
aagactttcaaggaagaagcaacagccccactattaacacccaaagctaatgttctatttaactattccttg
Digression: BioXXX projects

- **BioPython**: [http://biopython.org/wiki/Main_Page](http://biopython.org/wiki/Main_Page)
- **BioPerl**: [http://www.bioperl.org/wiki/Main_Page](http://www.bioperl.org/wiki/Main_Page)
- **BioJava**: [http://biojava.org/wiki/Main_Page](http://biojava.org/wiki/Main_Page)
- **BioRuby**: [http://bioruby.org](http://bioruby.org)
- **BioCaml**: [http://biocaml.org](http://biocaml.org)
Digression: BioPython

The Biopython Project is an international association of developers of non-commercial Python tools for computational molecular biology, as well as bioinformatics.

BioPython is one of a number of Bio* projects designed to reduce code duplication.

http://biopython.org/wiki/Main_Page
Introduction

Digression: BioPython

The main function is `Bio.SeqIO.parse()` which takes a file handle and format name, and returns a `SeqRecord` iterator.

```python
from Bio import SeqIO
handle = open("example.fasta", "rU")
for record in SeqIO.parse(handle, "fasta") :
    print record.id
handle.close()
```
from Bio import SeqIO
handle = open("example.fasta", "rU")
records = list(SeqIO.parse(handle, "fasta"))
handle.close()
print records[0].id  # first record
print records[-1].id # last record

from Bio import SeqIO
handle = open("example.fasta", "rU")
record_dict = SeqIO.to_dict(SeqIO.parse(handle, "fasta"))
handle.close()
print record_dict["gi:12345678"]  # use any record ID
Prediction of secondary structure: RNAfold

```
barbibul:rna-data$ RNAfold < trna.fa
>AF041468
GGGGGUAUAGCUCAGUUGGUAGAGCGCUGCCUUUGCACGGCAGAUGUCAGGGGUUCGAGUCCCCUUACCUCCA
(((((((........)))).(((((.......))))).....(((((.......)))))))))))). (-29.80)
```

Prediction of secondary structure: RNAfold

```
barbibul:rna-data$ RNAfold < trna.fa
>AF041468
GGGGGUAUAGCUCAGUUGGUAGAGCGCUGCCUUUGCACGGCAGAUGUCAGGGGUUCGAGUCCCCUUACCUCCA
(((((((((............))))).((((((........)))))))).(((((........)))))))))))).
barbibul:rna-data$
```
Prediction of secondary structure: RNAfold

http://rna.tbi.univie.ac.at/cgi-bin/RNAfold.cgi
Prediction of secondary structure: RNAfold

http://rna.tbi.univie.ac.at/cgi-bin/RNAfold.cgi
Plan

1. Introduction

2. RNA secondary structure prediction

3. Pseudoknot prediction and alternate models
RNA secondary structure prediction

- Many plausible secondary structures can be drawn from a sequence.
- The number increases exponentially with sequence length.
- An RNA only 200 bases long has over $10^{50}$ possible base-paired structures.
- We must distinguish the biologically correct structure from all the incorrect structures.
Base pair maximisation: the Nussinov folding algorithm

- One (naive) approach is to find the structure with the most base pairs.

- Nussinov introduced an efficient dynamic programming algorithm for this problem.

- Although the criterion is too simplistic to give accurate structure predictions, the algorithm is instructive because the mechanics of the Nussinov folding algorithm are the same as those in the more sophisticated energy minimisation folding algorithms (and of probabilistic SCFG-based algorithms).
**RNA secondary structure**

**Definition**

Let $u \in \{A, C, G, U\}^*$ be a sequence. An **RNA-structure** over $u$ is a set of pairs

$$P = \{(i, j) : i < j, u[i] \text{ and } u[j] \text{ form a WC or non-standard pair}\}$$

with the property that the associated graph has degree at most 1 (i.e., every base can have at most one bond).

**Remark**

$$\forall (i, j), \quad (i, j) \in P \Rightarrow \forall i', (i', j) \notin P$$

$$\forall (i, j), \quad (i, j) \in P \Rightarrow \forall j', (i, j') \notin P$$
RNA secondary structure

Purine riboswitch (Rfam RF00167)
The Nussinov folding algorithm

- **Idea (biological):** Stacked base pairs of helical regions are considered to stabilize an RNA molecule. Therefore, the goal is to maximize the number of base pairs.

- **Idea (algorithmic):** The optimal structure $S[i, j]$ on a subsequence $u[i, j]$ can only be formed by two distinct ways from a shorter subsequence $u[i + 1, j]$:
  1. Base $i$ is unpaired, followed by an arbitrary shorter structure.
  2. Base $i$ is paired with some partner base $k$ requiring the computation of two independent substructures: the structure enclosed by the bp and the remaining structure behind the pair.
The Nussinov folding algorithm

- **Initialisation**

  \[
  \gamma(i, i - 1) = 0 \quad 2 \leq i \leq n \\
  \gamma(i, i) = 0 \quad 1 \leq i \leq n 
  \]

- **Recursion**

  \[
  \gamma(i, j) = \max \left\{ \begin{array}{l}
  \gamma(i + 1, j) \\
  \gamma(i, j - 1) \\
  \gamma(i + 1, j - 1) + \alpha(i, j) \\
  \max_{i < k < j} \left\{ \gamma(i, k) + \gamma(k + 1, j) \right\}
  \end{array} \right. 
  \]

- **O\(n^3\)** time and **O\(n^2\)** space.
The Nussinov folding algorithm

1. $i+1$ unpaired
2. $j$ unpaired
3. $i,j$ pair
4. bifurcation
The Nussinov folding algorithm: Example
# The Nussinov folding algorithm: Example

The Nussinov folding algorithm is a method for predicting the secondary structure of RNA molecules. It is based on the principle that RNA molecules fold into stable structures, and these structures can be represented as a lattice of base pairs and unpaired nucleotides.

## Example

Consider the following RNA sequence:

```
GGGAAGAUCUCA
```

We can represent the possible base pairs and unpaired nucleotides using a matrix and a series of calculations to determine the most stable structure.

### Matrix Representation

The matrix below shows the possible base pairs and unpaired nucleotides for the RNA sequence:

<table>
<thead>
<tr>
<th></th>
<th>G</th>
<th>G</th>
<th>G</th>
<th>A</th>
<th>A</th>
<th>A</th>
<th>U</th>
<th>C</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>G</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>U</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

### Calculation

Each cell in the matrix represents the score of the base pair at that position, with higher scores indicating more stable base pairs. The scores for unpaired nucleotides are typically set to zero. The algorithm calculates the score for each possible structure by iteratively adding scores for each base pair and unpaired nucleotides.

### Stability Calculation

For each possible structure, the algorithm calculates the total score by summing the scores of all base pairs and unpaired nucleotides. The structure with the highest score is considered the most stable.

### Example Calculation

For the given RNA sequence, we can calculate the scores for different structures and select the one with the highest score. The algorithm continues this process until all possible structures are evaluated, and the most stable structure is determined.

### Conclusion

The Nussinov folding algorithm is a powerful tool for predicting RNA secondary structures. By iteratively calculating scores for each possible structure, it allows researchers to identify the most stable RNA conformation, which is crucial for understanding RNA function and structure.

---

S. Vialette (CNRS & LIGM)  
RNA Secondary Structures  
2014-2015 36 / 124
The Nussinov folding algorithm: Example
The Nussinov folding algorithm: Example
## The Nussinov folding algorithm: Example

### Table Representation

<table>
<thead>
<tr>
<th></th>
<th>G</th>
<th>G</th>
<th>G</th>
<th></th>
<th>A</th>
<th>A</th>
<th></th>
<th>C</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>G</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>U</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The Nussinov folding algorithm: Example

<table>
<thead>
<tr>
<th></th>
<th>G</th>
<th>G</th>
<th>G</th>
<th>A</th>
<th>A</th>
<th>A</th>
<th>U</th>
<th>C</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>G</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>U</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The Nussinov folding algorithm: Example

<table>
<thead>
<tr>
<th></th>
<th>G</th>
<th>G</th>
<th>G</th>
<th>A</th>
<th>A</th>
<th>A</th>
<th>U</th>
<th>C</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>G</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>G</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>G</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>U</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
The Nussinov folding algorithm

- The value $\gamma(1, n)$ is the number of base pairs in the maximally base-paired structure.

- There are often a number of alternatives structures with the same number of base-pairs.

- To find one of these maximally base-paired structures, we trace back through the values we calculated in the dynamic programming matrix, beginning from $\gamma(1, n)$. 
RNA secondary structure prediction

The Nussinov folding algorithm: Traceback stage

1. $S \leftarrow \text{emptyStack}$
2. $\text{push}(S, (1, n))$
3. while $S \neq \emptyset$ do
   4. $(i, j) \leftarrow \text{pop}(S)$
   5. if $i < j$ then
      6. if $\gamma(i + 1, j) = \gamma(i, j)$ then
         7. $\text{push}(S, (i + 1, j))$
      8. else if $\gamma(i, j - 1) = \gamma(i, j)$ then
         9. $\text{push}(S, (i, j - 1))$
      10. else if $\gamma(i + 1, j - 1) + \alpha(i, j) = \gamma(i, j)$ then
         11. record pair $(i, j)$
         12. $\text{push}(S, (i + 1, j - 1))$
   13. else
      14. for $j + 1 \leq k \leq j - 1$ do
         15. if $\gamma(i, k) + \gamma(k + 1, j) = \gamma(i, j)$ then
            16. $\text{push}(S, (k + 1, j))$
            17. $\text{push}(S, (i, k))$
            18. break
The Nussinov folding algorithm: Traceback stage

```
G G G A A A U C C
G 0 0 0 0 0 1 2 3
G 0 0 0 0 0 1 2 3
G 0 0 0 0 0 1 2 2
A 0 0 0 0 1 1 1 1
A 0 0 0 1 1 1 1
A 0 0 1 1 1
U 0 0 0 0
C 0 0 0
C 0 0
```

S. Vialette (CNRS & LIGM)
The Nussinov folding algorithm: Traceback stage

- Linear time and space.

- In the canonical implementation of the traceback step, whenever there are multiple structures that are equivalent in terms of number of base-pairs the first structure that works is chosen because the algorithm does not care about anything besides the number of base-pairs, so any structure with the same number of base pairs as the optimal one will do.

- However, this ignores important information that can lead it to choose an unstable structure over a more stable one.
The Nussinov folding algorithm: Traceback stage

Possible improvements:

- Incorporates the following assumptions:
  - Longer stems (consecutive base pairs) are more stable than shorter stems,
  - A single loop or bulge is more stable than one split in two by a base pair in the middle

- Report all optimal solutions.
The Nussinov folding algorithm: Drawbacks

The Nussinov folding algorithm does not determine biological relevant structures since:

- There are many (all!?) possibilities to form base pairs.
- Stackings of base pairs are not considered.
- The size of internals loops are not considered.
An SCFG version of the Nussinov folding algorithm

- A single non-terminal $S$;
- 14 production rules with associated probability parameters.

\[
S \rightarrow aS \mid cS \mid gS \mid uS \quad \text{(i unpaired)}
\]
\[
S \rightarrow Sa \mid Sc \mid Sg \mid Su \quad \text{(j unpaired)}
\]
\[
S \rightarrow aSu \mid cSg \mid gSc \mid uSa \quad \text{(i and j paired)}
\]
\[
S \rightarrow SS \quad \text{(bifurcation)}
\]
\[
S \rightarrow \epsilon \quad \text{(termination)}
\]
An SCFG version of the Nussinov folding algorithm

- Assume that the probability parameters are known.

- The maximum probability parse of a sequence with this SCFG is an assignment of sequence positions to productions.

- Because the productions correspond to secondary structure elements (base pairs and single-stranded bases), the maximum probability parse is equivalent to the maximum probability secondary structure.

- If base pair productions have relatively high probability, the SCFG will favour parses which tend to maximise the number of base pairs in the structure.
RNA structure prediction: MFE-folding

- More realistic: thermodynamics and statistical mechanics.
- Stability of an RNA secondary structure coincides with thermodynamic stability.
- Quantified as the amount of free energy released/used by forming base pairs.
RNA structure prediction: MFE-folding

RNA molecules basically exist in a distribution of structures rather than a single ground-state conformation.

- “Most likely” conformation: structure exhibiting minimum of free energy (MFE).

- Energy contributions of different loop types have been measured.

- Since free energies are additive, a more sophisticated model, the standard energy model for RNA secondary structures, can be proposed.

- Based on loop decomposition, the total energy $E$ of a structure $S$ can be computed as the sum over the energy contributions of each constituent loop $\ell$:

$$E(S) = \sum_{\ell \in S} E(\ell)$$
MFE folding: Example

- 4 nt loop +5.9
- 1 nt bulge +3.3
- 5' dangle -0.3
- unstructured single strand 0.0

- 1.1 terminal mismatch of hairpin
- 2.9 stack
- 2.9 stack (special case of 1 nt bulge)
- 1.8 stack
- 0.9 stack
- 1.8 stack
- 2.1 stack

overall $\Delta G = -4.6$ kcal/mol
Structural elements: Formal definition

Definition

Secondary structure elements Let \( u \) be a fixed sequence. Further, let \( S \) be an RNA secondary structure for \( u \).

- A base pair \((i, j) \in S\) is a **hairpin loop** if
  \[
  \forall i < i' \leq j' < i, \quad (i', j') \not\in S
  \]

- A base pair \((i, j) \in S\) is called **stacking** if \((i + 1, j - 1) \in P\).

- Two base pairs \((i, j) \in S\) and \((i', j') \in S\) form an **internal loop** \((i, j, i', j')\) if
  - \(i < i' < j' < j\)
  - \((i' - i) + (j - j') > 2\) (no stack)
  - there is no base pair \((k, l)\) between \((i, j)\) and \((i', j')\)
Structural elements: Formal definition

**Definition**

Secondary structure elements (ctd)

- An internal loop \((i, j, i', j')\) is called **left** (resp. **right**) **bulge**, if \(j = j' + 1\) (resp. \(i' = i + 1\)).

- A **k-multiloop** consists of \(k\) base pairs \((i_1, j_1), (i_2, j_2), \ldots, (i_k, j_k) \in S\) and a closing base pair \((i, j) \in S\) with the property that
  - \(i < i_1 < j_1 < i_2 < j_2 < \ldots < i_k < j_k < j\)
  - \(i + 1, \ldots, i_1 - 1, j_1 + 1, \ldots, i_2 - 1, \ldots, j_k - 1 + 1, \ldots, i_k - 1, j_k + 1, \ldots, j - 1\) are unpaired in \(S\)
  - \((i_1, j_1), (i_2, j_2), \ldots, (i_k, j_k)\) close the inner base pairs of the multiloop.
Structural elements: Formal definition

(a) Stack  (b) Hairpin  (c) Bulge  (d) Internal  (e) Multiple
Structural elements: \( k \)-multiloop

Remarks

- Usually hairpin loops have minimal loop size of \( m = 3 \) (i.e., for all \( (i, j) \in S, i < j - 3 \)).
- Each secondary structure element is defined uniquely by its closing basepair.
- For any basepair \( (i, j) \) we denote the corresponding secondary structure element with \( \text{sec}(i, j) \).
Energy of secondary structural elements

**Definition (Energy contribution of loops)**

Energy contributions of the various structure elements:

- Hairpin loop \((i, j)\): \(e_H(i, j)\)
- Stacking \((i, j)\): \(e_S(i, j)\)
- Internal loop \((i, j, i, j')\): \(e_L(i, j, i', j')\)
- Multiloop \((i, j, i_1, j_2, \ldots, i_k, j_k)\): \(e_M(i, j, i_1, j_2, \ldots, i_k, j_k)\)

**Remarks**

- General multi loop contribution will be too expensive in prediction: exponential explosion!
- Use a simplified contribution scheme.
Definition (Simplified energy contribution of multiloops)

\[ eM(i, j, k, k') = a + bk + ck' \]

where

- \( a, b \) and \( c \) are weights (\( a \) is the energy contribution for closing of loop),
- \( k \) is the number of inner base pairs, and
- \( k' \) is the number of unpaired bases within loop.
MFE folding

- The complexity of the dynamic programming algorithm is $O(n^4)$ time and $O(n^2)$ space.

- Using a trick, the time complexity can be reduced to $O(n^3)$.

- We assume traceback is done analogously to Nussinov-Traceback. Same reduced complexity. Only extension: trace through three matrices, (i.e., keep track of matrix).
Plan

1. Introduction

2. RNA secondary structure prediction

3. Pseudoknot prediction and alternate models
RNA pseudoknots

- A pseudoknot is a nucleic acid secondary structure containing at least two stem-loop structures in which half of one stem is intercalated between the two halves of another stem.

- The pseudoknot was first recognized in the turnip yellow mosaic virus in 1982.

- Pseudoknots fold into knot-shaped three-dimensional conformations but are not true topological knots.
RNA pseudoknots
RNA pseudoknots

A

B
RNA pseudoknots: Prediction and identification

- The structural configuration of pseudoknots does not lend itself well to bio-computational detection due to its context-sensitivity or "overlapping" nature.
- The presence of pseudoknots in RNA sequences is more difficult to predict by the standard method of dynamic programming, which use a recursive scoring system to identify paired stems and consequently, most cannot detect non-nested base pairs.
- Popular secondary structure prediction methods do not predict pseudoknot structures present in a query sequence.
- It is possible to identify a limited class of pseudoknots using dynamic programming, but these methods are not exhaustive and scale worse as a function of sequence length than non-pseudoknotted algorithms.
- The general problem of predicting lowest free energy structures with pseudoknots has been shown to be \textbf{NP}-complete.
RNA pseudoknots: Biological significance

- Several important biological processes rely on RNA molecules that form pseudoknots, which are often RNAs with extensive tertiary structure.

- For example, the pseudoknot region of RNase P is one of the most conserved elements in all of evolution.

- The telomerase RNA component contains a pseudoknot that is critical for activity.

- Several viruses use a pseudoknot structure to form a tRNA-like motif to infiltrate the host cell.
RNA pseudoknot type

- Simple, H-type

![Diagram of RNA pseudoknot types](image-url)
RNA pseudoknot type

- Kissing hairpin
RNA pseudoknot type

- Three-knot
RNA pseudoknot prediction

<table>
<thead>
<tr>
<th></th>
<th>R&amp;G</th>
<th>A/U</th>
<th>L&amp;P</th>
<th>D&amp;P</th>
<th>CCJ</th>
<th>R&amp;E</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Time</strong></td>
<td>$O(n^4)$</td>
<td>$O(n^4)/O(n^5)$</td>
<td>$O(n^5)$</td>
<td>$O(n^5)$</td>
<td>$O(n^5)$</td>
<td>$O(n^6)$</td>
</tr>
<tr>
<td><strong>Space</strong></td>
<td>$O(n^2)$</td>
<td>$O(n^3)/O(n^3)$</td>
<td>$O(n^3)$</td>
<td>$O(n^4)$</td>
<td>$O(n^4)$</td>
<td>$O(n^4)$</td>
</tr>
</tbody>
</table>
Nearest Neighbor Energy Model

For a secondary structure $S$

- the number of **base pairs stackings** is

$$\text{BPS}(S) = |\{(i,j) \in S : (i+1,j-1) \in S\}|$$

- the number of **stacking base pairs** is

$$\text{SBP}(S) = |\{(i,j) \in S : (i+1,j-1) \in S \text{ or } (i-1,j+1) \in S\}|$$
Without pseudoknots

- Maximizing the number of base pairs is $O(n^3)$ time and $O(n^2)$ space:

$$\gamma(i,j) = \max \begin{cases} 
\gamma(i+1,j), \\
\gamma(i,j-1), \\
\gamma(i+1,j-1) + \alpha(i,j), \\
\max_{i < k < j} \{ \gamma(i,k) + \gamma(k+1,j) \}
\end{cases}$$

- To maximize BPS or SBP, dynamic programming can be extended.
With arbitrary pseudoknots

- Maximizing the number of base pairs is $O(n^{2/5})$ time.
  The problem reduces to finding a maximum matching in a graph (solvable in $O(m\sqrt{n})$ time)

- To maximize BPS or SBP, matching becomes inadequate, and dynamic programming cannot be extended.
Complexity $\text{BPS}(S)$

**Theorem**

*Given a sequence $u$ and a positive integer $K$, it is $\text{NP}$-complete to decide whether there exists a structure $S$ that is legal under the canonical RNA folding model and with $\text{BPS}(S) \geq k$.***
**Bin Packing**

**Definition (The Bin Packing problem)**

- **Input:** $k$ items of sizes $a_1, a_2, \ldots, a_k$ and $B$ bins each with capacity $C$.
- **Question:** Decide whether the items fit into the bins.

**Remarks**

- Strongly NP-complete.
- A straightforward greedy algorithm achieves an approximation factor of 2.
- Does not have a polynomial-time approximation scheme (PTAS) unless $P = NP$.
- For any $0 < \epsilon \leq 1$, it is possible to find a solution using at most $(1 + \epsilon)opt + 1$ bins in polynomial time (asymptotic PTAS).
Strongly NP-complete

Definition

A problem is said to be strongly NP-complete (or NP-complete in the strong sense), if it remains so even when all of its numerical parameters are bounded by a polynomial in the length of the input.

- Normally numerical parameters to a problem are given in binary notation, so a problem of input size $n$ might contain parameters whose size is exponential in $n$.

- If we redefine the problem to have the parameters given in unary notation, then the parameters must be bounded by the input size.

- Thus strong NP-completeness or NP-hardness may also be defined as the NP-completeness or NP-hardness of this unary version of the problem.
Strongly NP-complete

Definition
A problem is said to be strongly NP-complete (or NP-complete in the strong sense), if it remains so even when all of its numerical parameters are bounded by a polynomial in the length of the input.

- **BIN PACKING** is strongly NP-complete while the 0-1 **KNAPSACK** problem is only weakly NP-complete.

Thus the version of **BIN PACKING** where the object and bin sizes are integers bounded by a polynomial remains **NP**-complete, while the corresponding version of the **KNAPSACK** problem can be solved in polynomial time by dynamic programming.

- Any strongly NP-hard optimization problem with a polynomially bounded objective function cannot have an **FPTAS** unless P = NP.
Theorem

Given a sequence \( u \) and a positive integer \( K \), it is \( \text{NP} \)-complete to decide whether there exists a structure \( S \) that is legal under the canonical RNA folding model and with \( \text{BPS}(S) \geq k \).

Proof.

Construction:

\[
\begin{align*}
    u &= C^{a_1}AC^{a_2}A \ldots AC^{a_k}AA(AG^C)^B \\
    K &= \sum_{1 \leq i \leq k} a_i - k
\end{align*}
\]

As \( A \)'s can only form base pairs with \( U \)'s in the canonical folding model, all base pairs in a legal structure for \( u \) will be \( G \cdot C \).
Complexity $\text{BPS}(S)$

**Theorem**

Given a sequence $u$ and a positive integer $K$, it is $\text{NP}$-complete to decide whether there exists a structure $S$ that is legal under the canonical RNA folding model and with $\text{BPS}(S) \geq k$.

**Proof.**

- $|u| = \sum_{1 \leq i \leq k} a_i + BC + B + k + 1$.
- Since the $\text{BIN PACKING}$ problem is strongly $\text{NP}$-complete we can assume that the $B, C, a_1, a_2, \ldots, a_k$ are all polynomially bounded by the size of the originally $\text{BIN PACKING}$ instance.
Theorem

Given a sequence $u \in \{0, 1\}^*$ and a positive integer $K$, it is NP-complete to decide whether there exists a structure $S$ that is legal under the general RNA folding model with $B = \{(0, 1), (1, 0)\}$ with $\text{BPS}(S) \geq k$.

Proof.

- Reduction from BIN PACKING:
  - $3 \leq a_i \leq C$ for $1 \leq i \leq k$, and
  - $2 \leq B \leq k$.

- $u = 0^{a_1}110^{a_2}11\ldots110^{a_k}(01^C)^B$.

- $K = \sum_{1 \leq i \leq k} a_i - k + B$. 
**Complexity SBP(S)**

**Theorem**

Given an alphabet $\Sigma$, a set of legal base pairs $B \subseteq \Sigma \times \Sigma$, a sequence $u \in \Sigma^*$ and a positive integer $K$, it is NP-complete to decide whether there exists a structure $S$ that is legal under the general RNA folding model with $SBP(S) \geq k$.

**Proof.**

- Reduction from **RESTRICTED SATISFIABILITY:**

![Diagram](image_url)
2-intervals

Definition (2-intervals)

- A 2-interval $D = (I, J)$ consists of two disjoint (closed) intervals $I$ and $J$ such that $I < J$ (i.e., $I$ is completely on the left of $J$).

- Two 2-intervals $D_1 = (I_1, J_1)$ and $D_2 = (I_2, J_2)$ are disjoint if the four intervals $I_1, J_1, I_2$ and $J_2$ are pairwise disjoint.

Definition (Relations between disjoint 2-intervals)

Let $D_1 = (I_1, J_1)$ and $D_2 = (I_2, J_2)$ be two disjoint 2-intervals. We have the following relations:

- $D_1 < D_2$ if $I_1 < J_1 < I_2 < J_2$.

- $D_1 \sqsubset D_2$ if $I_2 < I_1 < J_1 < J_2$.

- $D_1 \nsubseteq D_2$ if $I_1 < I_2 < J_1 < J_2$. 
2-intervals: Models

Definition (Models)

- A **model** is a non-empty subset of \( \{<, \sqsubset, \emptyset\} \).
- A set of 2-intervals \( \mathcal{D} \) is \( \mathcal{R} \)-structured if any two distinct 2-intervals in \( \mathcal{D} \) is \( R \)-comparable for some \( R \in \mathcal{R} \).

Key idea: Model \( \mathcal{R} = \{<, \sqsubset\} \) denotes pseudoknot-free structures.
2-intervals: Structured subsets

Definition (The 2-INTERVAL PATTERN problem)

- **Input**: A set of 2-intervals $\mathcal{D}$, a model $\mathcal{R}$ and a positive integer $K$.
- **Question**: Decide whether there exists a $\mathcal{R}$-structured subset $\mathcal{D}' \subseteq \mathcal{D}$ of size $K$.

If each 2-interval $D \in \mathcal{D}$ is associated with a non-negative weight $w(D)$, we are left with the WEIGHTED 2-INTERVAL PATTERN problem (i.e., decide whether there exist a $\mathcal{R}$-structured subset $\mathcal{D}' \subseteq \mathcal{D}$ of total weight at least $K$).
2-intervals: Support and restriction

Definition (Support)
The support of a set of 2-intervals $\mathcal{D}$ is the set of intervals $\{I, J : (I, J) \in \mathcal{D}\}$.

Definition (Restriction)
Support restrictions:

- **Unlimited**: no restriction.
- **Balanced**: every 2-intervals consists of two intervals of equal length.
- **Unit**: every 2-intervals consists of two intervals of unit length.
- **Point**: the intervals in the support are pairwise disjoint.
2-intervals: State-of-the-art

<table>
<thead>
<tr>
<th>Model</th>
<th>Unlimited</th>
<th>Balanced</th>
<th>Unit</th>
<th>Point</th>
</tr>
</thead>
<tbody>
<tr>
<td>{&lt;, ⊑, ⊏}</td>
<td>APX-hard</td>
<td></td>
<td></td>
<td>$O(n\sqrt{n})$</td>
</tr>
<tr>
<td>{⊑, ⊏}</td>
<td>APX-hard</td>
<td></td>
<td></td>
<td>$O(n \log(n) + \mathcal{L})$</td>
</tr>
<tr>
<td>{&lt;, ⊏}</td>
<td></td>
<td></td>
<td></td>
<td>NP-complete</td>
</tr>
<tr>
<td>{&lt;, ⊑}</td>
<td>\multirow{4}{*}{\begin{array}{c}O(n \log(n) + dn) \ {} \ {&lt;} \ {} \end{array}}</td>
<td>\multirow{4}{*}{\begin{array}{c}O(n \log(n) + \mathcal{L}) \ O(n \log(n)) \ O(n \log(n)) \end{array}}</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
2-intervals: Approximation ratios

<table>
<thead>
<tr>
<th>Model</th>
<th>Unlimited</th>
<th>Balanced</th>
<th>Unit</th>
<th>Point</th>
</tr>
</thead>
<tbody>
<tr>
<td>{&lt;, ≤, ⊑}</td>
<td>4</td>
<td>4</td>
<td>$2 + \epsilon$</td>
<td>N/A</td>
</tr>
<tr>
<td>{≤, ⊑}</td>
<td>4</td>
<td>4</td>
<td>$2 + \epsilon$</td>
<td>N/A</td>
</tr>
<tr>
<td>{&lt;, ⊑}</td>
<td></td>
<td></td>
<td>PTAS</td>
<td></td>
</tr>
</tbody>
</table>
2-intervals: Model $\mathcal{R} = \{<, \sqsubset, \sqsupset\}$

**Theorem**

The 2-INTERVAL PATTERN problem for unlimited (resp. unit) 2-intervals and model $\mathcal{R} = \{<, \sqsubset, \sqsupset\}$ is approximable within ratio 4 (resp. 3.).

**Remarks**

- The approximation algorithm for unit 2-intervals is $O(n \log(n))$ time, where $n$ is the number of input 2-intervals.
- The approximation algorithm for unlimited 2-interval uses linear programming techniques, which in practice are very often too time costly.
2-intervals: Model $\mathcal{R} = \{<, \sqsubset, \sqsupseteq\}$

**Theorem**

The 2-INTERVAL PATTERN problem for balanced 2-intervals and model $\mathcal{R} = \{<, \sqsubset, \sqsupseteq\}$ is approximable within ratio 4 ($O(n^2)$ time algorithm).

**Proof.**

**Data:** A set of balanced 2-intervals $\mathcal{D}$

**Result:** A $\{<, \sqsubset, \sqsupseteq\}$-structured subset of $\mathcal{D}$

1. $\mathcal{D}_{\text{sol}} \leftarrow \emptyset$
2. while $\mathcal{D} \neq \emptyset$ do
3.   Let $\mathcal{D}_{\text{min}}$ be the smallest 2-interval left in $\mathcal{D}$
4.   $\mathcal{D}_{\text{sol}} \leftarrow \mathcal{D}_{\text{sol}} \cup \{\mathcal{D}_{\text{min}}\}$
5.   $\mathcal{D} \leftarrow \mathcal{D} \setminus \{\mathcal{D} \in \mathcal{D} : \mathcal{D} \cap \mathcal{D}_{\text{min}} \neq \emptyset\}$
6. return $\mathcal{D}_{\text{sol}}$
2-intervals: Covering intervals

Definition

Let $D = (I, J)$ be a 2-interval. The covering interval of $D$, denoted $c(D)$, is the smallest interval that covers $D$ (i.e., $c(D) = [l(I), r(J)]$, where $l(i)$ (resp. $r(J)$) is the left (resp. right) endpoint of $I$ (resp. $J$).

Observation

Let $\mathcal{D}$ be a set of 2-intervals. For any $\{\sqsubseteq, \sqnot\}$-structured subset $\mathcal{D}' \subseteq \mathcal{D}$, the associated covering intervals $c(\mathcal{D}')$ are pairwise intersecting.
Interval graphs

Definition (Interval graph)

An interval graph is the intersection graph of a multiset of intervals on the real line. It has one vertex for each interval in the set, and an edge between every pair of vertices corresponding to intervals that intersect.
Interval graphs

Remarks

- Determining whether a given graph $G = (V, E)$ is an interval graph can be done in $O(|V| + |E|)$ time by seeking an ordering of the maximal cliques of $G$ that is consecutive with respect to vertex inclusion.

- A graph is an interval graph if and only if it is chordal and its complement is a comparability graph.

- The number of maximal cliques in a chordal graph is linear in the size of the graph.
Chordal graphs

Definition (Chordal graph)

A graph is **chordal** if each of its cycles of four or more vertices has a chord, which is an edge that is not part of the cycle but connects two vertices of the cycle.
Definition (Comparability graph)

A **comparability graph** is a graph that has a transitive orientation, an assignment of directions to the edges of the graph (i.e., an orientation of the graph) such that the adjacency relation of the resulting directed graph is transitive: whenever there exist directed edges \((x, y)\) and \((y, z)\), there must exist an edge \((x, z)\).
2-intervals: \( \mathcal{R} = \{ \sqsubseteq, \sqsupseteq \} \)

**Theorem**

The 2-INTERVAL PATTERN problem for unlimited (resp. unit) 2-intervals and model \( \mathcal{R} = \{ \sqsubseteq, \sqsupseteq \} \) is approximable within ratio 4 (resp. 3). (The algorithm is \( O(n^2 \log(n)) \) time for unit 2-intervals.)

**Proof.**

**Data:** A set of 2-intervals \( \mathcal{D} \)

**Result:** A \( \{ \sqsubseteq, \sqsupseteq \} \)-structured subset of \( \mathcal{D} \)

\begin{verbatim}
1 c(\mathcal{D}) \leftarrow \{c(D) : D \in \mathcal{D}\}
2 \mathcal{K} \leftarrow \text{all maximal cliques of } \Omega(c(\mathcal{D}))
3 \text{foreach maximal clique } K \in \mathcal{K} \text{ do}
4 \quad \mathcal{D}_K \leftarrow \{D \in \mathcal{D} : c(D) \in K\}
5 \quad \mathcal{D}'_K \leftarrow \text{(Approximate) pairwise disjoint subset of } \mathcal{D}_K
6 \text{return the largest } \mathcal{D}'_K \text{ found}
\end{verbatim}
Trapezoid graphs

Definition (Trapezoid)

Consider two intervals $I$ and $J$ defined over two distinct horizontal lines. The trapezoid $T = (I, J)$ is the convex set of points bounded by $I$ and $J$, and the two arcs connecting the right and left endpoints of $I$ and $J$. The interval $I$ and $J$ are the top interval and the bottom interval of $T$.

A family of trapezoids is a finite set of trapezoids which are all defined over the same two horizontal lines.

Definition (Trapezoid graph)

A trapezoid graph is the intersection graph of a family of trapezoids.
Trapezoid graphs
Turning 2-intervals to trapezoids

Definition (Corresponding trapezoid family)

Let $\mathcal{D}$ be a set of 2-intervals and let $\alpha$ and $\beta$ be two distinct horizontal lines such that $\alpha$ is below $\beta$. The corresponding trapezoid family of $\mathcal{D}$, denoted $\mathcal{T}(\mathcal{D})$, is defined as the family containing a single trapezoid $T = (I', J')$ for each 2-interval $D = (I, J) \in \mathcal{D}$, where $I'$ is defined over $\alpha$, $J'$ is defined over $\beta$, and $I' = I$ and $J' = J$. 
2-intervals: Model $\mathcal{R} = \{<, \sqsubset\}$

**Observation**

Any two **disjoint** 2-intervals in $\mathcal{D}$ are $\{<, \sqsubset\}$-comparable if and only if their corresponding trapezoids in $T(\mathcal{D})$ are disjoint.

**Remarks**

- Felsner et al. gave a $O(n \log(n))$ time algorithm for finding a maximum disjoint subset in a family of trapezoids.

- But there may be disjoint trapezoids in $T(\mathcal{D})$ that correspond to non-disjoint 2-intervals in $\mathcal{D}$.

- $\{\sqcap\}$-comparable 2-intervals in $\mathcal{D}$ correspond to intersecting trapezoids in $T(\mathcal{D})$. 
Turning 2-intervals to trapezoids
Turning 2-intervals to trapezoids

\[ \begin{align*}
  I_1 &\quad I_2 &\quad J_2 \\
  D' &\quad D
\end{align*} \]
Turning 2-intervals to trapezoids

Definition (Clashing intervals)

Let $I_1 = [l(I_1), r(I_1)]$ and $I_2 = [l(I_2), r(I_2)]$ be two distinct intervals defined over two distinct horizontal lines such that $l(I_1) < l(I_2)$. The two intervals $I_1$ and $I_2$ clash if either

- $l(I_1) \leq l(I_2) \leq r(I_2) \leq r(I_1)$, or
- $l(I_1) \leq l(I_2) \leq r(I_1) \leq r(I_2)$.
Turning 2-intervals to trapezoids

Definition (Clashing trapezoids)

Let $T_1 = (I_1, J_1)$ and $T_2 = (I_2, J_2)$ be two distinct trapezoids in a family of trapezoids. The two trapezoids $T_1$ and $T_2$ clash if either

- $I_1$ and $J_2$ clash, or
- $I_2$ and $J_1$ clash.
Turning 2-intervals to trapezoids

Observation
Any two distinct 2-intervals in $\mathcal{D}$ are $\{<, \parallel\}$-comparable if and only if their corresponding trapezoids in $\mathcal{T}(\mathcal{D})$ are disjoint and do not clash.
2-intervals: Model $\mathcal{R} = \{<, \emptyset\}$

**Theorem**

The 2-INTERVAL PATTERN problem for unit (res. point) 2-intervals and model $\mathcal{R} = \{<, \emptyset\}$ is approximable within ratio 3 (resp. 2). (The algorithm is $O(n^2)$ time for unit 2-intervals.)

**Proof.**

Data: A set of 2-intervals $\mathcal{D}$
Result: A $\{<, \emptyset\}$-structured subset of $\mathcal{D}$

1. Construct the corresponding trapezoid family $\mathcal{T}(\mathcal{D})$
2. Compute $\mathcal{T}' \subseteq \mathcal{T}(\mathcal{D})$: the maximum pairwise disjoint subset of $\mathcal{T}(\mathcal{D})$
3. $\mathcal{T}_{\text{sol}} \leftarrow \emptyset$
4. while $\mathcal{T}' \neq \emptyset$ do
   5. Let $T_0$ be the leftmost trapezoid in $\mathcal{T}'$
   6. $\mathcal{T}_{\text{sol}} \leftarrow \mathcal{T}_{\text{sol}} \cup \{T_0\}$
   7. Omit $T_0$ and all trapezoids clashing with $T_0$ in $\mathcal{T}'$
8. return the 2-intervals corresponding to trapezoids in $\mathcal{T}_{\text{sol}}$
Pseudoknot prediction and alternate models

2-intervals: Model $\mathcal{R} = \{<, \leq\}$

**Theorem**

The 2-INTERVAL PATTERN problem for unit (resp. point) 2-intervals and model $\mathcal{R} = \{<, \leq\}$ is approximable within ratio 3 (resp. 2). (The algorithm is $O(n^2)$ time for unit 2-intervals.)

**Proof.**

**Data:** A set of 2-intervals $\mathcal{D}$

**Result:** A $\{<, \leq\}$-structured subset of $\mathcal{D}$

1. Construct the corresponding trapezoid family $\mathcal{T}(\mathcal{D})$
2. Compute $\mathcal{T}' \subseteq \mathcal{T}(\mathcal{D})$: the maximum pairwise disjoint subset of $\mathcal{T}(\mathcal{D})$
3. $\mathcal{T}_{\text{sol}} \leftarrow \emptyset$
4. **while** $\mathcal{T}' \neq \emptyset$ **do**
5. Let $T_0$ be the leftmost trapezoid in $\mathcal{T}'$
6. $\mathcal{T}_{\text{sol}} \leftarrow \mathcal{T}_{\text{sol}} \cup \{T_0\}$
7. Omit $T_0$ and all trapezoids clashing with $T_0$ in $\mathcal{T}'$
8. **return** the 2-intervals corresponding to trapezoids in $\mathcal{T}_{\text{sol}}$
2-intervals: Model $\mathcal{R} = \{<, \not\leq\} \}$

Theorem

The 2-INTERVAL PATTERN problem for unit (res. point) 2-intervals and model $\mathcal{R} = \{<, \not\leq\}$ is approximable within ratio 3 (resp. 2). (The algorithm is $O(n^2)$ time for unit 2-intervals.)

Proof.

- Let $T_0$ be the leftmost trapezoid $\mathcal{T}'$ and let $D_0$ be its corresponding 2-interval in $\mathcal{D}$.
- By our definition of a 2-interval and of $\mathcal{T}(\mathcal{D})$, any trapezoid in $\mathcal{T}(\mathcal{D})$ has a bottom interval which is completely to the left of its top interval.
- Thus, $T_0$ can only clash with trapezoids on its right in $\mathcal{T}'$. 
2-intervals: Model $\mathcal{R} = \{<, \nmid, \nngtr\}$

Theorem

The 2-INTERVAL PATTERN problem for unit (res. point) 2-intervals and model $\mathcal{R} = \{<, \nmid, \nngtr\}$ is approximable within ratio $3$ (resp. $2$). (The algorithm is $O(n^2)$ time for unit 2-intervals.)

Proof.

- if $\mathcal{D}$ is a point 2-interval set, then all 2-intervals with left intervals intersecting the right interval of $D_0$ have the same left interval, and as $\mathcal{T}'$ is pairwise disjoint, at most one of these has a corresponding trapezoid in $\mathcal{T}'$. 
2-intervals: Model $\mathcal{R} = \{<, \emptyset\}$

**Theorem**

The 2-INTERVAL PATTERN problem for unit (res. point) 2-intervals and model $\mathcal{R} = \{<, \emptyset\}$ is approximable within ratio 3 (resp. 2). (The algorithm is $O(n^2)$ time for unit 2-intervals.)

**Proof.**

- if $\mathcal{D}$ is a unit 2-interval set, intersecting intervals involved in $\mathcal{D}$ must overlap.
- Thus, any trapezoid in $\mathcal{T}'$ clashing with $T_0$ corresponds to a 2-interval with a left interval which contains either endpoints, but not both, of the right interval of $D_0$.
- Since $\mathcal{T}'$ is pairwise disjoint, there can be at most two such trapezoids in $\mathcal{T}'$. 
2-intervals: Model $\mathcal{R} = \{<, \emptyset\}$

Theorem

The **2-INTERVAL PATTERN** problem for balanced 2-intervals and model $\mathcal{R} = \{<, \emptyset\}$ is approximable within ratio 5. (The algorithm is $O(n^2)$ time.)

Proof.

**Data:** A set of balanced 2-intervals $\mathcal{D}$

**Result:** A $\{<, \emptyset\}$-structured subset of $\mathcal{D}$

1. Construct the corresponding trapezoid family $\mathcal{T}(\mathcal{D})$
2. Compute $\mathcal{T}' \subseteq \mathcal{T}(\mathcal{D})$: the maximum pairwise disjoint subset of $\mathcal{T}(\mathcal{D})$
3. $\mathcal{T}_{\text{sol}} \leftarrow \emptyset$
4. **while** $\mathcal{T}' \neq \emptyset$ **do**
5. Let $T_0$ be the smallest trapezoid in $\mathcal{T}'$
6. $\mathcal{T}_{\text{sol}} \leftarrow \mathcal{T}_{\text{sol}} \cup \{T_0\}$
7. Omit $T_0$ and all trapezoids clashing with $T_0$ in $\mathcal{T}'$
8. **return** the 2-intervals corresponding to trapezoids in $\mathcal{T}_{\text{sol}}$
2-intervals: Model $\mathcal{R} = \{<, \emptyset\}$

Definition (Proper trapezoid family)

A family of trapezoids $\mathcal{T}$ is proper if for any two distinct trapezoids $T_1 = (I_1, J_1)$ and $T_2 = (I_2, J_2)$ in $\mathcal{T}$, $I_1 \cap I_2 = \emptyset$ and $J_1 \cap J_2 = \emptyset$.
2-intervals: Model $\mathcal{R} = \{<, \llbracket\}\}$

Definition (Strongly proper trapezoid family)

A proper family of trapezoids $\mathcal{T}$ is strongly proper if for any two distinct trapezoids $T_1 = (I_1, J_1)$ and $T_2 = (I_2, J_2)$ in $\mathcal{T}$, if $J_1$ and $I_2$ clash then $l(I_2) \leq l(J_1) < r(J_1) \leq r(I_2)$, where $l(J_1), r(J_1)$ and $l(I_2), r(I_2)$ are the left and right endpoints of $J_1$ and $I_2$, respectively.
2-intervals: Model $\mathcal{R} = \{<, \emptyset\}$

Remarks

- Any pairwise disjoint family of trapezoids is proper (but the converse is not true).
- Thus Step 2 of the preceding algorithm yields a proper family of trapezoids $\mathcal{T}'$. 
2-intervals: Model \( \mathcal{R} = \{<, \parallel\} \)

Remarks

- Computing a strongly proper subset \( \mathcal{T}' \subseteq \mathcal{T}'' \) can be done easily by adjusting the loop step: Instead of omitting all trapezoids clashing with the leftmost trapezoid in this iteration, we need only to omit a small subset of these trapezoids.

More specifically, let \( T_0 = (I_0, J_0) \) be the leftmost trapezoid in \( \mathcal{T}' \). We only omit trapezoids \( T = (I, J) \) with
  - \( l(I) \leq l(I_0) \leq r(I) \), or
  - \( l(I) \leq r(I_0) \leq r(I) \).

- We obtain a strongly proper trapezoid family \( \mathcal{T}'' \subseteq \mathcal{T}' \) if we proceed in this fashion such that \( 3|\mathcal{T}''| \geq |\mathcal{T}'| \).
2-intervals: Model $\mathcal{R} = \{<, \parallel\}$

Definition (Clashing trapezoid graph)
Let $\mathcal{T}$ be a family of trapezoids. The **clashing trapezoid graph** of $\mathcal{T}$, denoted $G_\mathcal{T}$, is the graph such that each vertex in $G_\mathcal{T}$ correspond to a distinct trapezoid in $\mathcal{T}$, and two vertices are connected by an edge if and only if their corresponding trapezoid clash.
2-intervals: Model $\mathcal{R} = \{<,\emptyset\}$

**Theorem**

Let $T$ be a family of trapezoids. If $T$ is strongly proper then $G_T$ is a forest.

**Proof.**
2-intervals: Model $\mathcal{R} = \{<, \rangle\}$

**Theorem**

Let $\mathcal{T}$ be a family of trapezoids. If $\mathcal{T}$ is strongly proper then $G_{\mathcal{T}}$ is a forest.

**Proof.**

- Define $\tilde{G}_{\mathcal{T}}$ as the directed graph obtained by orienting the edges of $G_{\mathcal{T}}$ according to the precedence relation in $\mathcal{T}$:

$$V(\tilde{G}_{\mathcal{T}}) = V(G_{\mathcal{T}})$$
$$E(\tilde{G}_{\mathcal{T}}) = \{(T_1, T_2) : \{T_1, T_2\} \in E(G_{\mathcal{T}}) \text{ and } T_1 < T_2\}$$

- Since $\mathcal{T}$ is strongly proper, every trapezoid in $\mathcal{T}$ clashes with at most one trapezoid on its left, and hence the in-degree of every vertex $T \in V(\tilde{G}_{\mathcal{T}})$ is at most 1.
2-intervals: Model $\mathcal{R} = \{<, \emptyset\}$

**Theorem**

Let $\mathcal{T}$ be a family of trapezoids. If $\mathcal{T}$ is strongly proper then $G_\mathcal{T}$ is a forest.

**Proof.**

- Hence any cycle $(T_0, \ldots, T_k, T_0)$ in $G_\mathcal{T}$ is a directed cycle in $\tilde{G}_\mathcal{T}$.
- Then we must have $T_0 < T_k < T_0$ by definition of $\tilde{G}_\mathcal{T}$. This is a contradiction.
- Then it follows that $G_\mathcal{T}$ is acyclic.
2-intervals: Model $\mathcal{R} = \{<, \emptyset\}$

Remarks

- A maximum independent set in any forest of size $n$ is of size at least $\frac{n}{2}$. (This set can be found in linear time with respect to $n$.)

- if $\mathcal{T}$ is a pairwise disjoint family of trapezoids, then any independent set of $G_{\mathcal{T}}$ corresponds to a pairwise disjoint non-clashing set of trapezoids, and hence corresponds to a $\{<, \emptyset\}$-comparable set of 2-intervals.
2-intervals: Model $\mathcal{R} = \{<, \bar{\cdot}\}$

**Theorem**

The 2-INTERVAL PATTERN problem for model $\mathcal{R} = \{<, \bar{\cdot}\}$ is approximable within ratio 6. (The algorithm is $O(n^2)$ time.)

**Proof.**

Data: A set of 2-intervals $\mathcal{D}$  
Result: A $\{<, \bar{\cdot}\}$-structured subset of $\mathcal{D}$

1. Construct $\mathcal{T}(\mathcal{D})$, the corresponding trapezoid set of $\mathcal{D}$
2. Compute $\mathcal{T}'$, the maximum pairwise disjoint subset of $\mathcal{T}(\mathcal{D})$
3. Compute $\mathcal{T}''$, a strongly proper subset of $\mathcal{T}'$, such that $3|\mathcal{T}''| \geq |\mathcal{T}'|
4. Compute $G_{\mathcal{T}''}$ and the maximum independent set of $G_{\mathcal{T}''}$
5. return the 2-intervals corresponding to the maximum independent set of $G_{\mathcal{T}''}$
2-intervals: Model $\mathcal{R} = \{<,\emptyset\}$

**Theorem**

The **2-Interval Pattern** problem for model $\mathcal{R} = \{<,\emptyset\}$ is approximable within ratio 2. (The algorithm is $O(n^3 \log(n))$ time.)
2-intervals: Model $\mathcal{R} = \{<, \bowtie\} $

### Definition (Precedence or crossing)

For two 2-intervals $D_1 = (I_1, J_1)$ and $D_2 = (I_2, J_2)$, the **preference or crossing** relation $\bowtie$ is defined by:

$$D_1 \bowtie D_2 \text{ if and only if } D_1 < D_2 \text{ or } D_1 \bowtie D_2$$

### Remarks

- If $D_1 \bowtie D_2$ then $I_1 < I_2$ and $J_1 < J_2$.
- The $\bowtie$ relation specifies a total order for any $\{<, \bowtie\}$-structured 2-interval set.
2-intervals: Model $\mathcal{R} = \{<, \triangleright\}$

Let $\mathcal{D}$ be a $\{<, \triangleright\}$-structured 2-interval set and let $\mathcal{D}_\triangleright$ denotes $\mathcal{D}$ ordered by the $\triangleright$ relation (viewed as an ordered sequence).

- $\mathcal{D}_\triangleright[i]$ denotes the 2-interval with rank $i$ in $\mathcal{D}$.
- $\mathcal{D}_\triangleright[i, j]$ denotes the subsequence $\mathcal{D}_\triangleright[i], \mathcal{D}_\triangleright[i + 1], \ldots, \mathcal{D}_\triangleright[j]$.
2-intervals: Model $\mathcal{R} = \{<, \hat{\cdot}\}$

- For each index $1 \leq i \leq |\mathcal{D}_{\hat{\cdot}}|$, define $\text{next}(i)$ as the smallest index $j$, $1 \leq j \leq |\mathcal{D}_{\hat{\cdot}}|$ such that $\mathcal{D}_{\hat{\cdot}}[i] < \mathcal{D}_{\hat{\cdot}}[j]$. If such an index $j$ does not exist define $\text{next}(i) = |\mathcal{D}_{\hat{\cdot}}| + 1$.

- Define the **backbone indices** of $\mathcal{D}_{\hat{\cdot}}$ as the sequence of indices $i_1, i_2, \ldots, i_k$ such that $i_1 = 1$, $i_j = \text{next}(i_{j-1})$ and $\text{next}(i_k) = |\mathcal{D}_{\hat{\cdot}}| + 1$.

  (For convenience, we define $\text{next}(i_k) = |\mathcal{D}_{\hat{\cdot}}| + 1$ and imagine a 2-interval $\mathcal{D}_{\hat{\cdot}}[i_{k+1}]$ such that $\mathcal{D}_{\hat{\cdot}}[i] < \mathcal{D}_{\hat{\cdot}}[i_{k+1}]$ for all $1 \leq i \leq |\mathcal{D}_{\hat{\cdot}}|$.)
2-intervals: Model $\mathcal{R} = \{<, \parallel\}$
2-intervals: Model $\mathcal{R} = \{<, \|\}$. 

- For each backbone index $1 \leq i_s \leq k$, define a stripe
  $\mathcal{T}(i_s) = D_{\|}[i_s + 1, i_{s+1} - 1]$.

- The stripe is **odd** if $s$ is odd; it is **even** if $s$ is even.

- For each 2-interval $D \in \mathcal{T}(i_s)$, we observe that
  $D_{\|}[i_s] \| D \| D_{\|}[i_s + 1]$.

- Every stripe of $D_{\|}$ is $\{\|\}$-structured.
2-intervals: Model $\mathcal{R} = \{<, \nothing\}$

$\text{next}(1) = 3$

$\text{next}(3) = 6$

$\text{next}(6) = 6 + 1$
2-intervals: Model \( \mathcal{R} = \{<, \loom\} \)

- A \( \{<, \loom\}\)-structured sequence \( \mathcal{D}_{\loom} \) is **striped** if either if either
  - its odd stripes are all empty, or
  - its even stripes are all empty.

- Although \( \mathcal{D}_{\loom} \) is not always striped, it contains two striped subsequences:

\[
\mathcal{D}_{\loom}[i_1] \loom(i_1) \mathcal{D}_{\loom}[i_2] \mathcal{D}_{\loom}[i_3] \loom(i_3) \mathcal{D}_{\loom}[i_4] \ldots \\
\mathcal{D}_{\loom}[i_1] \mathcal{D}_{\loom}[i_2] \loom(i_2) \mathcal{D}_{\loom}[i_3] \mathcal{D}_{\loom}[i_4] \loom(i_4) \ldots
\]

These two subsequences together cover the sequence \( \mathcal{D}_{\loom} \): the 2-intervals at the backbone indices are covered twice, each remaining 2-interval is covered once.

One of the two subsequences has a length of at least \( |\mathcal{D}_{\loom}|/2 \).
2-intervals: Model $\mathcal{R} = \{<, \emptyset\}$

- A $\{<, \emptyset\}$-structured sequence $\mathcal{D} \emptyset$ is **striped** if either if either
  - its odd stripes are all empty, or
  - its even stripes are all empty.

- Although $\mathcal{D} \emptyset$ is not always striped, it contains two striped subsequences:

  \[
  \mathcal{D} \emptyset[i_1] \mathcal{T}(i_1) \mathcal{D} \emptyset[i_2] \mathcal{D} \emptyset[i_3] \mathcal{T}(i_3) \mathcal{D} \emptyset[i_4] \ldots
  
  \mathcal{D} \emptyset[i_1] \mathcal{D} \emptyset[i_2] \mathcal{T}(i_2) \mathcal{D} \emptyset[i_3] \mathcal{D} \emptyset[i_4] \mathcal{T}(i_4) \ldots
  \]

  These two subsequences together cover the sequence $\mathcal{D} \emptyset$: the 2-intervals at the backbone indices are covered twice, each remaining 2-interval is covered once.

  **One of the two subsequences has a length of at least $|\mathcal{D} \emptyset|/2$.**
2-intervals: Model \( \mathcal{R} = \{<, \emptyset\} \)

**Theorem**

*The 2-Interval Pattern problem for model \( \mathcal{R} = \{<, \emptyset\} \) is approximable within ratio 2. (The algorithm is \( O(n^3 \log(n)) \) time.)*

**Proof.**
2-intervals: Model $\mathcal{R} = \{<, \emptyset\}$

Theorem

The 2-INTERVAL PATTERN problem for model $\mathcal{R} = \{<, \emptyset\}$ is approximable within ratio 2. (The algorithm is $O(n^3 \log(n))$ time.)

Proof.

- **Step 1.** Make a dummy 2-interval $D_\omega$ such that $D_\gamma < D_\omega$ for all $D_\gamma \in \mathcal{D}$.

  Set $\mathcal{D}^+ = \mathcal{D} \cup \{D_\omega\}$. 
2-intervals: Model $\mathcal{R} = \{<, \emptyset\}$

**Theorem**

The 2-INTERVAL PATTERN problem for model $\mathcal{R} = \{<, \emptyset\}$ is approximable within ratio 2. (The algorithm is $O(n^3 \log(n))$ time.)

**Proof.**

- **Step 2.** For each pair of 2-intervals $D_\alpha$ and $D_\beta$ in $\mathcal{D}^+$, $D_\alpha < D_\beta$, find the subset of 2-intervals

  $$\mathcal{D}^+_{\alpha,\beta} = \{D_\gamma : D_\gamma \in \mathcal{D}^+ \text{ and } D_\alpha \emptyset D_\gamma \emptyset D_\beta\}$$

  Then compute $C_{\alpha,\beta}$, a maximum size $\{\emptyset\}$-structured subset of $\mathcal{D}^+_{\alpha,\beta}$. 
2-intervals: Model $\mathcal{R} = \{<, \varnothing\}$

**Theorem**

The 2-INTERVAL PATTERN problem for model $\mathcal{R} = \{<, \varnothing\}$ is approximable within ratio 2. (The algorithm is $O(n^3 \log(n))$ time.)

**Proof.**

- **Step 3.1.** Process the 2-intervals in $D^\alpha _{\alpha, \beta}$ in an arbitrary order that conforms to the partial order specified by the $<$ relation. For each 2-interval $D_\beta$ in $D^+$, find the subset of 2-intervals

$$D^+_\beta = \{D_\alpha : D_\alpha \in D^+ \text{ and } D_\alpha < D_\beta\}$$
2-intervals: Model $\mathcal{R} = \{<, \emptyset\}$

**Theorem**

The 2-INTERVAL PATTERN problem for model $\mathcal{R} = \{<, \emptyset\}$ is approximable within ratio 2. (The algorithm is $O(n^3 \log(n))$ time.)

**Proof.**

1. **Step 3.2.** If $D_\beta^+ = \emptyset$, $A_\beta \leftarrow \{\beta\}$ and $B_\beta \leftarrow \{\beta\}$. Otherwise
   - Find $D_\alpha \in D_\beta^+$ such that $|B_\alpha|$ is maximum and set $A_\beta \leftarrow B_\beta \cup \{D_\beta\}$,
   - Find $D_\alpha \in D_\beta^+$ such that $|A_\alpha| + |C_{\alpha,\beta}|$ is maximum and set $B_\beta \leftarrow A_\alpha \cup C_{\alpha,\beta} \cup \{D_\beta\}$. 

\[\square\]
2-intervals: Model $\mathcal{R} = \{<, \emptyset\}$

**Theorem**

The **2-Interval Pattern** problem for model $\mathcal{R} = \{<, \emptyset\}$ is approximable within ratio 2. (The algorithm is $O(n^3 \log(n))$ time.)

**Proof.**

- **Step 4.** Let $D_{sol}$ be either $A_\omega$ or $B_\omega$ such that $|D_{sol}|$ is maximum.

  Return $D_{sol} \setminus \{\omega\}$. 
2-intervals: Model $\mathcal{R} = \{<, \}\}$

Theorem

*The 2-Interval Pattern problem for model $\mathcal{R} = \{<, \} \}$ is approximable within ratio 2. (The algorithm is $O(n^3 \log(n)))$ time.*

Proof.

Notes

- In the algorithm we use $A_\beta$ and $B_\beta$ to represent the two different alternating patterns, with $\beta$ as both the last element backbone element.
- The 2-interval $D_\alpha$ in steps 3.1 and 3.2 represents the second-to-last backbone element in $A_\beta$ and $B_\beta$.
- The subset $C_{\alpha, \beta}$ represents the maximum size stripe between the two backbone elements $D_\alpha$ and $D_\beta$. 


**$d$-claw free graphs**

**Definition ($d$-claw, $d$-claw-free)**

For an undirected graph $G$, a $d$-claw $C$ is an induced subgraph $K_{1,d}$ that consists of an independent set $T_c$ of $d$ vertices (called talons) and a center vertex $z_C$ that is connected to all the talons.

A graph is $d$-claw-free if it has no $d$-claws.

(A 3-claw is commonly called a claw so that a graph is claw-free if and only if it does not contain the complete bipartite graph $K_{1,3}$ (known as the "claw graph") as an induced subgraph.)
\(d\)-claw free graphs

The regular icosahedron, a polyhedron whose vertices and edges form a claw-free graph.
**d-claw free graphs**

**Definition (The MAXIMUM WEIGHT INDEPENDENT SET problem)**

- **Input**: A graph $G = (V, E)$ and a weight function $w : V \to \mathbb{N}$.
- **Solution**: A set of independent vertices $V' \subseteq V$ that maximises $\sum_{v \in V'} w(v)$.

**Remarks**

In $d$-claw-free graphs

- **Arbitrary weight**: $(d/2 + \epsilon)$-approximation
- **Small weight**: $(d/2)$-approximation
- **Unit weight**: $((d - 1)/2 + \epsilon)$-approximation
2-intervals and $d$-claw free graphs

**Theorem**

For a set of 2-intervals $D$ with interval length $\ell$, $a \leq \ell \leq b$, the 2-interval graph $G(D)$ is $d$-claw-free for

$$d = 5 + \frac{2(b - 2)}{a}.$$ 

**Proof.**

- Let $I$ be an interval and let $\mathcal{I}$ be a set of disjoint intervals that intersect $I$.
- All intervals in $\mathcal{I}$ are completely contained in $I$ except possibly the leftmost one and the rightmost one.
2-intervals and $d$-claw free graphs

**Theorem**

For a set of 2-intervals $\mathcal{D}$ with interval length $\ell$, $a \leq \ell \leq b$, the 2-interval graph $G(\mathcal{D})$ is $d$-claw-free for

$$d = 5 + \frac{2(b - 2)}{a}.$$ 

**Proof.**

- Let $I$ be an interval and let $\mathcal{I}$ be a set of disjoint intervals that intersect $I$.
- All intervals in $\mathcal{I}$ are completely contained in $I$ except possibly the leftmost one and the rightmost one.
2-intervals and $d$-claw free graphs

Corollary

Let $D$ be a set of 2-intervals.

- if all intervals have the same length (unit support), the associated 2-interval graph $G(D)$ is 5-claw-free;

- if all intervals have length 2 or 3, the associated 2-interval graph $G(D)$ is 5-claw-free

Corollary

*The (Weighted) 2-Interval Pattern problem is approximable within ratio $2.5 + \epsilon$ for arbitrary weights and $2 + \epsilon$ for unit weights.*
Nearest Neighbor Energy Model

For a secondary structure $S$

- the number of **base pairs stackings** is

$$\text{BPS}(S) = |\{(i, j) \in S : (i + 1, j - 1) \in S\}|$$

- the number of **stacking base pairs** is

$$\text{SBP}(S) = |\{(i, j) \in S : (i + 1, j - 1) \in S \text{ or } (i - 1, j + 1) \in S\}|$$
Nearest Neighbor Energy Model

Definition (The **Maximum Base Pairs Stacking** (BPS) problem)

- **Input**: A sequence $u$.
- **Solution**: A secondary structure $S$ for $u$ that maximises $\text{BPS}(S)$.

Definition (The **Maximum Stacking Base Pairs** (SBP) problem)

- **Input**: A sequence $u$.
- **Solution**: A secondary structure $S$ for $u$ that maximises $\text{SBP}(S)$.
Approximating the **Maximum Base Pairs Stacking** problem

**Theorem**

The **Maximum Base Pairs Stacking** problem is approximable within ratio $8/3$.

**Proof.**
Approximating the **Maximum Base Pairs Stacking** problem

**Theorem**

The **Maximum Base Pairs Stacking** problem is approximable within ratio $8/3$.

**Proof.**

- **Step 1.** Repeatedly find the leftmost 5 consecutive stacking loops (i.e., find the 2-interval $([x, x + 5], [y - 5, y])$ where $x$ is as small as possible). Add these stacking loops to $S$.

- **Step 2.** Repeatedly find any 4 consecutive stacking loops. Add these stacking loops to $S$.

- **Step 3.** Repeatedly find any 3 consecutive stacking loops. Add these stacking loops to $S$. 

Approximating the **Maximum Base Pairs Stacking** problem

**Theorem**

The **Maximum Base Pairs Stacking** problem is approximable within ratio $8/3$.

**Proof.**

- **Step 4.1.** Construct a 2-interval set $\mathcal{D}$ by associating a 2-interval to each 2 consecutive stacking loop.

- **Step 4.2.** Construct the 2-interval graph $G(\mathcal{D})$ and assign each vertex a weight: 1 for a single stacking loop and 2 for two consecutive stacking loops.
Approximating the **Maximum Base Pairs Stacking** problem

**Theorem**

The **Maximum Base Pairs Stacking** problem is approximable within ratio $8/3$.

**Proof.**

- **Step 4.3.** Find a maximum weight independent set $D'$ in $G(D)$ (5/2-approximation algorithm for 5-claw-free graphs).
- **Step 4.4.** For each 2-interval in $D'$, add the corresponding stacking loop in $S$. 
Approximating the **Maximum Base Pairs Stacking** problem

**Theorem**

*The Maximum Base Pairs Stacking* problem is approximable within ratio $8/3$.

**Proof.**

- Let $s_1, s_2, s_3$ and $s_4$, respectively, be the number of stacking loops found by the first, second, third and fourth of our algorithm.
Approximating the Maximum Base Pairs Stacking problem

Theorem

The Maximum Base Pairs Stacking problem is approximable within ratio $8/3$.

Proof.

- Let $S^*$ be the set of stacking loops in an optimal secondary structure.
- Let $s_1^*$, $s_2^*$ and $s_3^*$, respectively, be the number of stacking loops in $S^*$ that intersect the stacking loops found by the first, second and third step of our algorithm.
- Let $s_4^*$ be the number of remaining stacking loops in $S^*$ which are represented by 2-intervals in $\mathcal{D}$. 
Approximating the **Maximum Base Pairs Stacking** problem

**Theorem**

The **Maximum Base Pairs Stacking** problem is approximable within ratio 8/3.

**Proof.**

\[ |S| = s_1 + s_2 + s_3 + s_4 \]
\[ |S^*| = s_1^* + s_2^* + s_3^* + s_4^* \]
Approximating the **Maximum Base Pairs Stacking** problem

**Theorem**

The **Maximum Base Pairs Stacking** problem is approximable within ratio $8/3$.

**Proof.**

For each $k$ consecutive stacking loops $D$ found by the first three steps of our algorithm, the number of stacking loops in $S^*$ that intersect them is at most $2(k + 2)$ (i.e., $k + 2$ for each interval of the 2-interval $D$).
Approximating the **Maximum Base Pairs Stacking** problem

**Theorem**

The **Maximum Base Pairs Stacking** problem is approximable within ratio 8/3.

**Proof.**

**Step 1.**

- By always choosing the leftmost 5-consecutive stacking loop $D_5$, we can guarantee that the left interval of the 2-interval $D_5$ intersects at most $5 + 1$ stacking loops in $S^*$. 
Approximating the **Maximum Base Pairs Stacking** problem

**Theorem**

*The Maximum Base Pairs Stacking* problem is approximable within ratio $8/3$.

**Proof.**

**Step 1.**

- Suppose the contrary that the left interval of $D_5$ intersects 7 stacking loops in $S^*$.
- Then these 7 stacking loops must be consecutive, and the leftmost 5 of these stacking loops should have been choosen instead of $D_5$. 
Approximating the **MAXIMUM BASE PAIRS STACKING** problem

**Theorem**

The **MAXIMUM BASE PAIRS STACKING** problem is approximable within ratio $8/3$.

**Proof.**

**Step 1.**

$$\frac{s_1^*}{s_1} \leq \frac{5 + 1 + 5 + 2}{5} = \frac{13}{5} = 2.6$$
 Approximating the **Maximum Base Pairs Stacking** problem

**Theorem**

*The Maximum Base Pairs Stacking problem is approximable within ratio 8/3.*

**Proof.**

**Step 2.**

- With all 5 consecutive stacking loops found by the first step, we can guarantee that each interval of a 2-interval $D_4$ (consisting of 4 consecutive stacking loops) found by the second step of our algorithm intersects at most $4 + 1$ stacking loops in $S^*$. 
Approximating the **Maximum Base Pairs Stacking** problem

**Theorem**

The **Maximum Base Pairs Stacking** problem is approximable within ratio 8/3.

**Proof.**

**Step 2.**

- Suppose the contrary that an interval of $D_5$ intersects 6 stacking loops in $S^*$.
- Then these 6 stacking loops must be consecutive, and hence must contain 5 consecutive stacking loops.
- This is a contradiction.
Approximating the Maximum Base Pairs Stacking problem

Theorem

The Maximum Base Pairs Stacking problem is approximable within ratio $8/3$.

Proof.

Step 2.

$$\frac{s_2^*}{s_2} \leq \frac{4 + 1 + 4 + 1}{4} = \frac{10}{4} = 2.5$$
Approximating the **Maximum Base Pairs Stacking** problem

**Theorem**

The **Maximum Base Pairs Stacking** problem is approximable within ratio $8/3$.

**Proof.**

**Step 3.**

$$
\frac{s_3^*}{s_3} \leq \frac{3 + 1 + 3 + 1}{4} = \frac{8}{3} \approx 2.67
$$
Approximating the **Maximum Base Pairs Stacking** problem

**Theorem**

The **Maximum Base Pairs Stacking** problem is approximable within ratio $8/3$.

**Proof.**

**Step 4.**

- Each 2-interval in $D$ is balanced and corresponds to either a single stacking loop (with interval length 2) or two consecutive stacking loops (with interval length 3).
- Therefore the 2-interval graph $G(D)$ is 5-claw-free.
Approximating the **MAXIMUM BASE PAIRS STACKING** problem

**Theorem**

The **MAXIMUM BASE PAIRS STACKING** problem is approximable within ratio 8/3.

**Proof.**

**Step 4.**

- The **MAXIMUM WEIGHT INDEPENDENT SET** problem in 5-claw-free graphs is approximable within ratio 5/2.

\[
\frac{s_4^*}{s_4} \leq \frac{5}{2} = 2.5
\]
Approximating the **Maximum Base Pairs Stacking** problem

**Theorem**

The **Maximum Base Pairs Stacking** problem is approximable within ratio $8/3$.

**Proof.**

$$\frac{|S^*|}{|S|} = \frac{\sum_{i=1}^{4} s_i^*}{\sum_{i=1}^{4} s_i}$$

$$= \sum_{i=1}^{4} \frac{s_i^*}{\sum_{j=1}^{4} s_j}$$
Approximating the **Maximum Base Pairs Stacking** problem

**Theorem**

The **Maximum Base Pairs Stacking** problem is approximable within ratio $8/3$.

**Proof.**

$$
\frac{|S^*|}{|S|} \leq \frac{13}{5} \left( \frac{s_1}{\sum_{j=1}^{4} s_j} \right) + \frac{10}{4} \left( \frac{s_2}{\sum_{j=1}^{4} s_j} \right) + \frac{8}{3} \left( \frac{s_3}{\sum_{j=1}^{4} s_j} \right) + \frac{5}{2} \left( \frac{s_4}{\sum_{j=1}^{4} s_j} \right)
\leq \frac{8}{3} \left( \frac{s_1}{\sum_{j=1}^{4} s_j} \right) + \frac{8}{3} \left( \frac{s_2}{\sum_{j=1}^{4} s_j} \right) + \frac{8}{3} \left( \frac{s_3}{\sum_{j=1}^{4} s_j} \right) + \frac{8}{3} \left( \frac{s_4}{\sum_{j=1}^{4} s_j} \right) = \frac{8}{3}
$$