Designing RNA Structures is Hard

- A report on the paper by Édouard Bonnet, Paweł Rzążewski, and Florian Sikora

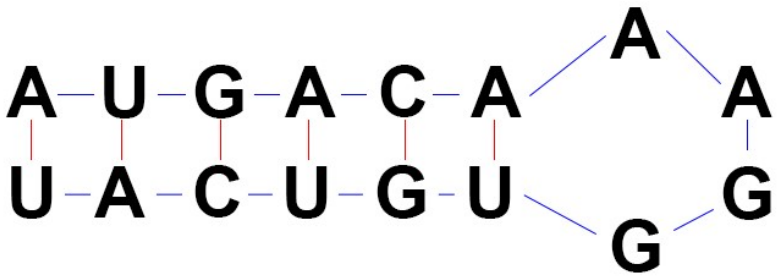
RNA Secondary Structure Design

- Secondary structure is important in the biological function of the molecule
- Solving the problem has real world applications:
  - Pharmaceutical research
  - Biochemistry
  - Synthetic biology
  - RNA nanostructures

RNA Structures

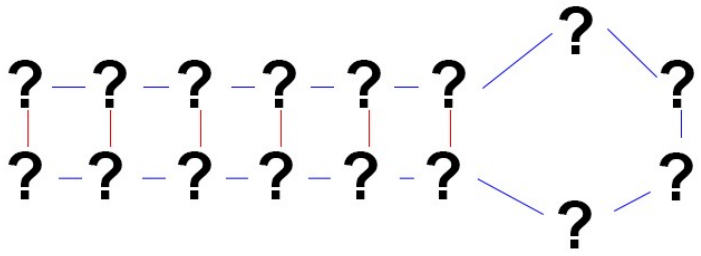
- RNA is made up of 4 nucleotides, labeled A, U, C, and G
- The **primary** structure of an RNA molecule is just a string over the alphabet {A, U, C, G} – each element is a **base**
- Nucleotides can bind together: A with U, C with G in the Watson-Crick energy model
- Each binding reduces the free energy of the molecule
- The **secondary** structure of a molecule is the set of positional pair bindings

RNA Structures



- This structure is called a **stem loop**
- In a **pseudo-knot-free** structure, no stem contains part of a stem from another stem loop
- A **minimum free energy** (MFE) structure is one with the maximal number of bases paired (each pair is worth -1 energy)
- Sequences want to fold into an MFE configuration

RNA Structures



- The RNA-DESIGN problem
  - Given a secondary structure, fill in the bases
  - The solution cannot fold into any other structure with more bound pairs
- RNA-DESIGN-EXTENSION (RDE): we are given some fixed bases in the sequence
- Thesis: RNA-DESIGN-EXTENSION is NP-Complete

Robustness of Proof

- Uses Watson-Crick energy model (simplest)
  - Other models are not uniform, more complex
  - Prove hardness independently of energy model
- Ignore pseudo-knots
  - Again, proving hardness in the easier case
- Reductions structure maps well to stem loops
  - This is realistic, stem loops are a basic RNA building block

RNA-DESIGN-EXTENSION is in NP

- The inverse problem is RNA-FOLDING
  - Given an RNA sequence, compute its MFE folding
  - Can be solved with DP in polynomial time
- We can use RNA-FOLDING as an oracle to verify solutions to RNA-DESIGN-EXTENSION
- Having such an oracle means that RNA-DESIGN-EXTENSION is in NP

Proving RDE is NP-Complete

- Reduce from E3-SAT
  - Each clause contains 3 distinct variables
  - Known NP-Hard if each variable is used up to 4 times
- Map SAT instance onto a string representation of RDE
- Label the string representation with bases
- Show that the base sequence labels are a solution to the RDE instance iff the SAT instance is satisfiable

Representing Secondary Structures

- A **structure** a well-parenthesized expression with dots (((..)(..)))
- ()'s represent a base pair, . represents an unpaired base
- A **sequence** is a string of the same length from {1,2,3,4}
  - 1=A, 2=C, 3=G, 4=U so proper pairs sum to 5
  - Sequence w corresponds to structure S if proper pairs match
- Sequence is a **design** if it can't fold into anything with more pairs
- A **partial sequence** is a sequence with some ?'s (unassigned)
- A partial sequence w' is a **design extension** if the ?'s can be filled in to turn it into a design.

Building the RDE instance: variables

- Start with a E3-SAT instance with **n** variable and **m** clauses
- Define **t** := n^2 and **y** := (n+3m)t
  - Note t=Θ(n^2), y=Θ(n^3); y >> t >> n
- A variable gadget V<Xi> is:

$$V\langle x_i \rangle := \underbrace{((((((((((((((}_{\text{length } i(m+1)y}\underbrace{\ldots\ldots\ldots}_{\text{length } t}\underbrace{))))))))))))))}_{\text{length } i(m+1)y}$$

- Where (‘s are labeled 1,)’s are labeled 4, and .’s are labeled 2 if the variable is true, else 3
- The parentheses are the **arch** of the variable

UCF

Building the RDE instance: clauses

- Let a clause Cj contain three literals la, lb, lc; a < b < c
  - la is the same gadget as V<Xa>
  - lb and lc are V<Xb>, V<Xc> with jy parentheses pairs remove
  - The clause is

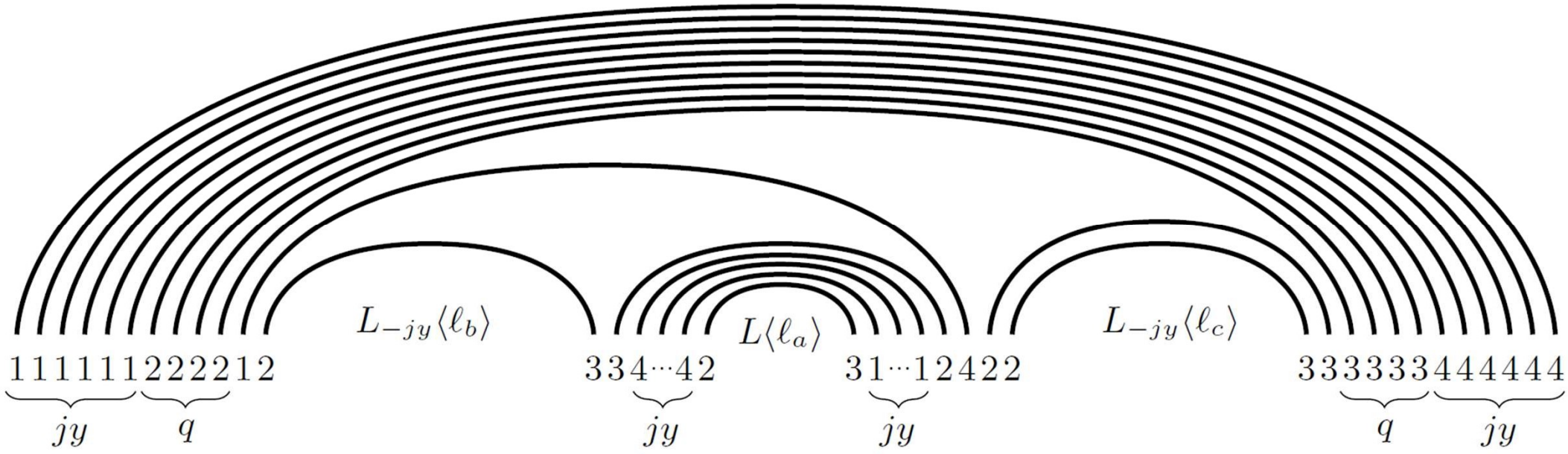
$$S\langle C_j\rangle := \underbrace{(\ldots(}_{jy}\underbrace{(\ldots(}_{q}((L_{-jy}\langle \ell_b\rangle)(\ \underbrace{(\ldots(}_{jy}(L\langle \ell_a\rangle)\underbrace{)\ldots)}_{jy})\underbrace{)\ldots)}_{q}\underbrace{)\ldots)}_{jy}$$

- The jy+q parentheses are the **arch** of the clause
  - The outer jy parentheses are the first arch layer
  - The inner q parentheses are the second arch layer

UCF

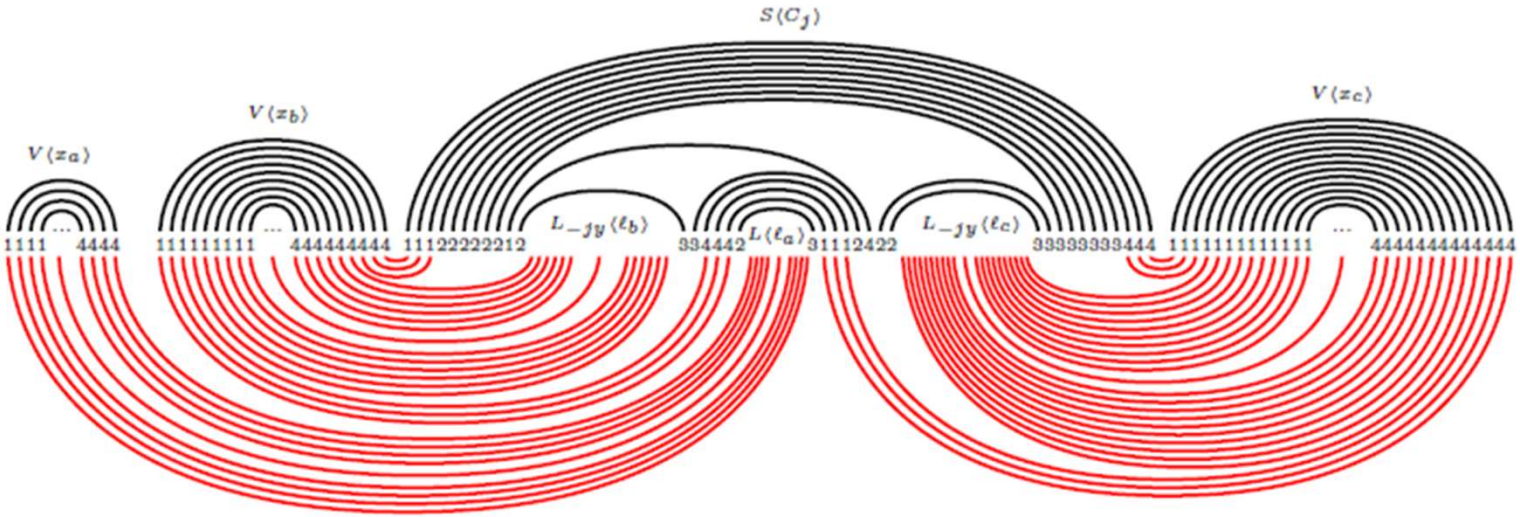
Building the RDE instance

- The entire instance is clauses interleaved with variables such that every clause is between the corresponding V<Xb> and V<Xc> gadgets



SAT unsatisfiable -> no design extension

- Black lines are by given construction, red lines are a re-matching by removing some parentheses
- If clause is unsatisfiable, then it is possible to rematch to a better structure
- This implies the originally constructed structure is NOT a design extension



SAT satisfiable -> design extension

- This direction is too gory for full details
- Given a structure S with a satisfiable SAT instance, assume
  - There is a better structure S’ for the corresponding sequence w
  - Assume wlog that S’ actually is the maximal matching
- By an argument of counting matching parts of S’, can prove that there exists S’’ which matches even MORE bases in w
- But this is a contradiction, so S’ cannot exist
- Therefore S itself must be the maximal matching and is a design extension
- We have proven SAT satisfiable iff the corresponding sequence is a design extension, thus RDE is NP-Hard
- We showed earlier than RDE is in NP, thus RDE is NP-Complete

Algorithmic Consequences

- Taking advantage of the structures in the proof leads to a faster algorithm
- Naïve: $O^*(4^n)$
- Using insights from the proof, can prune search space
  - $\sqrt{3}^n n^{O(1)}$, where n is the length of the input structure
  - $2^s n^{O(1)}$, where s is the number of unlabeled elements in the input structure
- RNA-DESIGN is known to be in P for saturated structures
  - Using DP based on ideas from the proof, RDE is also tractable on saturated structures
- There are many other tree-structured problems in computer science
  - This gadget mapping may have uses elsewhere

Questions?