A report on "Designing RNA Secondary Structures is Hard [1]"

Jim Geist – COT 6410, Spring 2020

Abstract. An RNA secondary structure is the shape that a given strand of RNA will fold into, and largely determines an RNA molecule's function. The ability to design an RNA molecule is an important research topic in many subfields of biology and medicine. Central to this process is the problem of discovering an RNA sequence that will fold into a desired secondary structure. In this report, we will discuss a paper which proves that this problem is NP-Complete, and then gives a practical algorithm for reducing the exponential complexity of the problem.

Introduction

RNA – ribonucleic acid – is one of the fundamental building blocks of life. RNA molecules are the templates from which proteins are built. Proteins, in turn, perform most of the complex work that occurs at the cellular level – from metabolism to recognizing neurotransmitters in nerve cell membranes. Some viruses are strands of foreign RNA which invade a host organism and use that organism's resources to replicate. [2]

A strand of RNA is a sequence of four nucleotides (A, C, G and U) called *bases*. Bases can bind together in pairs; each pair reduces the free energy of the molecule. In the simplest energy model, Watson-Crick, the only allowed pairs are AU and CG, and each pair of bases decreases the free energy of the molecule by 1. Minimum free energy is achieved by maximizing the number of pairs in the strand [1]. More complex, realistic models, such as the Nussinov-Jacobson model, have been developed which take into consider the locality of bases which are not actually paired [3].

In the simplest form of pairing, called a stem loop, the RNA strand loops back on itself, resulting in a hairpin loop of unpaired bases and a stem of paired bases. An RNA molecule consists of many stem loops. A molecule is said to be *pseudo-knot free* if no part of a stem participates as a stem in a second stem loop. The *primary* structure of an RNA molecule is the sequence of bases which comprise the unfolded strand. The *secondary* structure is the minimum free energy configuration the strand folds into.

The RNA-FOLDING problem is, given a sequence of bases, finding the minimum free energy, pseudoknot-free secondary structure that the sequence folds into. This problem is known to be solvable in O(n^3) time by dynamic programming [4] (where n is the length of the base sequence), or in slightly less polynomial time by more clever methods [5].

The inverse problem of RNA-FOLDING is RNA-DESIGN. In RNA-FOLDING we have a sequence and would like to find the optimal structure it will fold into; in RNA-DESIGN we have a secondary structure and would like to find a sequence of bases that folds into that structure. RNA-FOLDING can determine what a sequence of RNA might do by finding its shape; RNA-DESIGN determines how to implement an engineered secondary structure in terms of bases. A specialization of RNA-DESIGN, RNA-DESIGN-EXTENSION, is the problem solved in practice. This is RNA-DESIGN with the additional constraint that some of the bases in the solution sequence are predetermined and may not be changed.

Motivation

The secondary structure of RNA largely determines its biological function. The earlier problem, RNA-FOLDING, allowed researchers to determine what function a sequenced strand of RNA might perform. RNA-DESIGN-EXTENSION, on the other hand, allows researchers to design a secondary structure which performs some function, and compute an RNA sequence which implements that function. This has applications in nanostructures and synthetic biology [6], as well as pharmaceutical research [7]. RNA

structures can be quite large; the largest known gene in the human body is for a muscle protein called titin, containing over 80,000 base pairs [8]. Existing solutions used in these areas fall into one of three categories:

- Models which use heuristics.
- Models which take exponential time.
- Models which are crowd sourced, such as EteRNA, where the folding problem is presented in a video game interface and humans try to solve it [9].

Given the importance of the problem in growing medical fields and problem sizes which are large in the context of an exponential algorithm, decisively proving that the problem is NP-Complete can both direct resources to developing heuristic algorithms (rather than wasting time searching for a probably non-existent polynomial-time solution) and provide deeper insight into the problem structure to inform algorithm design.

Usefulness of problem as stated

The authors argue that proving the complexity of the problem as stated is a useful result. The Watson-Crick energy model is simple and reduces to maximal pairings; the proof does not depend on more complex energy models to prove hardness. Similarly, the proof is valid for pseudo-knot free structures; structures containing pseudo-knots will be at least as complex. Finally, the problem as presented at this level of simplicity is still useful in real-world structure design [1].

Terminology

To specify a problem instance, we need two things: a way to represent the desired structure, and an encoding of a potential solution. The paper uses the dot-bracket notation, also used in the popular open-source RNA toolkit ViennaRNA [10]. In this notation, parentheses and dots represent the

secondary structure. A matching pair of parentheses represent a base pair, and a dot represents an unpaired base. For example, the structure (((….))) represents a stem loop with three matched base pairs in the stem part and a loop of 4 unpaired bases.

A potential solution is a string of bases of the same length as the structure. The canonical way to do this would be the letters A, U, C, and G; however, the authors chose to represent the solution as the digits 1, 2, 3, and 4 such that valid base pairs sum to 5. This simplifies discussion of some of the more detailed parts of the proof.

A problem instance has, as well as a structure, an initial labeling. The alphabet for this labeling includes the letter '?' as well as {1,2,3,4}. A '?' may be assigned any base to form a solution; however, if a base is specified in the initial labeling, it may not be changed. This represents the real-world constraint that there are some restrictions on the sequences which may be used to build a structure. The paper uses the term *partial sequence* to refer to a solution that has '?''s, and *sequence* to refer to a solution that just has bases. A partial sequence is said to be *extended* to a sequence by replacing all instances of '?' with a base.

A sequence qualifies as a design if it is compatible with the desired structure and can fold into no other structure with more base pairs. A partial sequence can be extended into a design if there exists an extension to it which is a design.

Proof of Hardness

Bonnet et al. prove that RNA-DESIGN-EXTENSION, as specified here, is NP-Complete [1].

As mentioned above, the inverse problem of RNA-DESIGN-EXTENSION – RNA-FOLDING – can be solved in polynomial time. RNA-FOLDING can be used to check if a proposed solution to RNA-DESIGN-EXTENSION is correct, so RNA-DESIGN-EXTENSION is in NP.

The authors prove that RNA-DESIGN-EXTENSION is NP-Complete by reduction from E3-SAT. E3-SAT is 3-SAT with the restriction that each clause contains three distinct variables. From an arbitrary E3-SAT instance, an RNA-DESIGN-EXTENSION (henceforth RDE) instance is constructed. The proof proceeds by showing that the RDE instance has a design if and only if the SAT instance is satisfiable.

The construction of the RDE instance is done by defining gadgets for the SAT instance variables, literals and clauses in the RNA bracket-dot form. Assume the instance has *n* variables x_i and *m* clauses C_j. Define $t = n^2$ and $y = (n + 3m)t$. In the construction that follows, it will turn out that there will be *y* unpaired bases. A variable gadget is of the form $V < x_i > = ((\ldots))$ where there are $i(m + 1)y$ pairs of parentheses and *t* dots. The open parentheses are labeled 1 and the close parentheses are labeled 4. The dots are labeled 2 if the variable is assigned a true value; else they are labeled 3. Note that by construction, the parentheses are properly labeled with matching base pairs and the dots match nothing.

There are two forms of the literal gadget depending on the literal's position in the clause. The middle literal ($L < l_a >$ in the paper) is identical to the variable gadget the literal maps to. The exterior ($L_{-jy} < l_b >$ and ($L_{-jy} < l_c >$) literals are the same but remove jy pairs of parentheses.

A clause gadget is the concatenation of three literals – see Figure 1 from the original paper.

The entire SAT instance is encoded by concatenating all the variable and clause gadgets together, such that every clause gadget is between the two variable gadgets matching the outer literals of the clause. The entire structure can be viewed as a tree, where every subexpression inside a pair of parentheses is a child of that pair. The authors claim as obvious that such an ordering can be found in polynomial time.

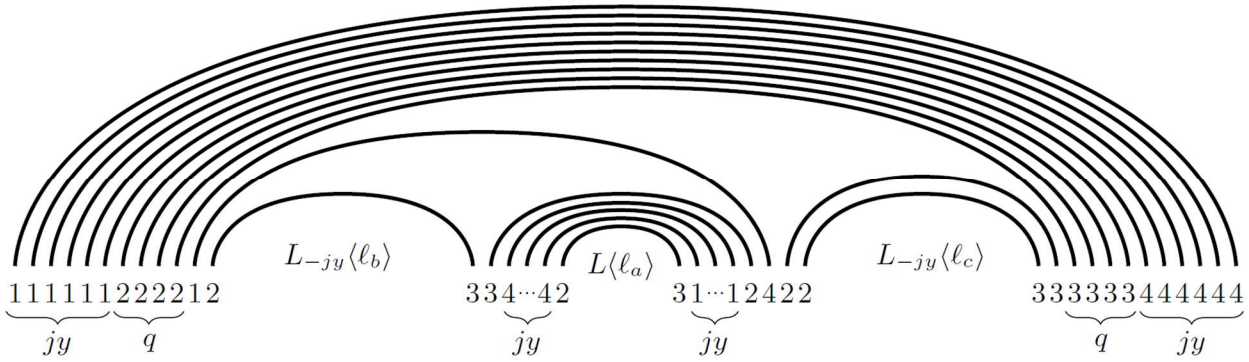

Figure 1: A Clause Gadget

The first half of the proof argues that if the SAT instance is unsatisfiable, then the labeling of the corresponding RDE instance does not have a design extension. Call the structure of the RDE instance S. If the SAT instance is not satisfiable, then there will be some clause C_j which is unsatisfied. If we remove the *q* parentheses around the outer literals, then it becomes possible to build a new structure S', identical to S except that the literals can now pair with the surrounding variables they map to. Careful counting of the parentheses we removed (just over *q*) and the pairings we add reveals that S' has more pairings than S. This means that the RDE solution implied by the labeling of S cannot be a design extension, as there exists a structure that is also compatible with it that has more base pairs.

The proof in the other direction, that if the SAT instance is satisfied then the labeling of the corresponding RDE instance can be extended to a design, is similar but with more edge cases. This half of the proof is by contradiction. Assume there is a satisfying assignment for the SAT instance. Also assume that there is some structure S', also compatible with the RDE instance labeling, that matches more base pairs. Without loss of generality we can take S' to be the structure that matches the most base pairs. The paper proves, via a counting argument, a bound on a the number of unpaired letters in any subtree (R) (i.e. any expression R inside parentheses) of S', and use that argument to prove that dots associated with a variable can only match with other dots for that variable (or their literals). A similar argument is used to show that the arches around clauses must only match with themselves. Both these

cases would cause more than *y* unpaired letters, and therefore S' would not be better than S. Finally, with these restrictions in place, the authors show a construction of a structure S'', which has more pairs than S', contradicting the existence of S' as a maximal matching better than S. Therefore S' cannot exist and S, having the maximal number of matched pairs, can be extended to a design.

Results

A naïve solution to RNA-DESIGN is $O^*(4^n)$. In the last section of their paper, the authors show how the tree structures used in the proof can be used to design a more efficient algorithm. The same properties of the structure used in the proof to show that a structure can or cannot be rematched to a better solution can be used to prove that a configuration in the search space can never yield a design. This allows early pruning of the search tree, greatly reducing the exponent base. Their proposed algorithm has the bounds

1. $\sqrt{3}^n \cdot n^{O(1)}$, n being the length of the input structure, and
2. $2^s \cdot n^{O(1)}$, s being the number of unlabeled elements in the input structure

This is still necessarily exponential, but far better than 4^n.

The authors also note that [7] gives a polynomial algorithm for RNA-DESIGN of saturated structures (i.e. structures with no unpaired bases) and claim that dynamic programing can, using the structures in the proof, extend this polynomial time solution to the RNA-DESIGN-EXTENSION problem.

Final thoughts

To summarize, the proof maps a version of SAT to a well-parenthesized, tree-like expression, and then uses that structure to prove something about the labeling of the expression. Tree-like structures and matching problems occur frequently in computer science, so it's likely that this kind of mapping could be used for problems outside the scope of bioinformatics. The authors do not mention this possibility.

It is inarguable that RNA-DESIGN-EXTENSION is an important problem. At the time of this writing, EteRNA's focus is a folding problem related to the COVID-19 pandemic [9]. Proving a long-suspected bound on the problem (per the authors, this has been an open problem for 20 years), while giving insight into the problem's structure, will help guide future research into finding better heuristics and not spending time looking for a polynomial-time solution that likely does not exist.

The paper is very well presented, with precise language and little ambiguity. The authors make their case well for both the importance of the problem and the real-world applicability of their formulation; their results depend solely on the design problem itself, not added complexities such as a more complex energy model or pseudo-knots. The high-level sketch of the proof makes the details of the proof itself (and this is a non-trivial proof) understandable after only two or three readings. The applicability of the proof structures to improved algorithms are a nice bonus in the conclusion. The only thing lacking is perhaps a section on future directions and work.

References

[1] É. Bonnet, P. Rzążewski and F. Sikora, "Designing RNA Structures is Hard," Arvix, 2018.

[2] J. W. Kalat, Biological Psychology, Fourth Edition, Pacific Grove, CA: Brooks/Cole Publishing Company, 1992.

[3] R. Nussinov and A. Jacobson, "Fast Algorithm for Predicting the Secondary Structure of Single-Stranded RNA," *Proceedings of the National Academy of Sciences of the United States of America,* vol. 77, no. 11, pp. 6309-6313, 1980.

[4] M. Zuker and P. Stiegler, "Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information.," *Nucleic acids research,* vol. 9, no. 1, pp. 133-148, 1981.

[5] K. Bringmann, F. Grandoni, B. Saha and V. V. Williams, "Truly Sub-cubic Algorithms for Language Edit Distance and RNA-Folding via Fast Bounded-Difference Min-Plus Product," in *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016*, 2016.

[6] A. Churkin, M. Retwitzer, V. Reinharz, Y. Ponty, J. Waldispuhl and D. Barash, "Design of RNAs: comparing programs for inverse RNA folding," *Briefings in Bioinformatics,* vol. 19, no. 2, pp. 250-358, 2018.

[7] J. Hales, A. Heliou, J. Manuch, Y. Ponty and L. Stacho, "Combaintorial RNA Design: Designability and Structure-Approximating Algorithms in Watson-Crick and Nussinov-Jacobson Enery Models," *Algorithmica,* vol. 79, no. 3, pp. 835-856, 2017.

[8] T. Brown, Genomes, 2nd edition, Oxford: Wiley-Liss, 2002.

[9] "https://eternagame.org/home/".

[10] R. Lorenz, S. Bernhart, C. Höner zu Siederdissen, H. Tafer, C. Flamm, P. Stadler and I. Hofacker, "ViennaRNA Package 2.0," *Algorithms for Molecular Biology,* vol. 6, 2011.