Nucleic Acid Folding
ISYE 3133 Project

1 Problem Description

Nucleic Acid Folding Problem is to predict the secondary structure of an Nucleic Acid molecule, given only its nucleotide sequence. This important, classic problem in computational biology is often solved with variants of dynamic programming which have been highly-refined and engineered in several widely-used computer programs. But here, we look at how integer linear programming can be used to obtain the same results and can also be extended to model more complex versions of the folding problem, in ways that are difficult to model with dynamic programming. We start with an ILP formulation for a simplified version of the Nucleic Acid problem, and then extend the biological model and the ILP formulation to incorporate more realistic biological features of the problem.

1.1 The first crude model

We let S denote a string of n characters made up of the Nucleic Acid alphabet $\{A, C, U, G\}$. For example, $S = ACGUGCCACGAU$. A pairing is a set of disjoint pairs of characters in S. A character can be in at most one pair. Note that some characters might not be in any pair in a pairing. A pair is called complementary if the two characters in the pair are $\{A, U\}$ or $\{C, G\}$. In our first model of Nucleic Acid folding, we require that all pairs be complementary. If we draw the Nucleic Acid string S as a circular string, we define a nested pairing (alternately called non-crossing) as a pairing of complementary nucleotides, where each pair in the pairing is connected by a line inside the circle, and where none of the lines cross each other as shown in Figure 1.

Fold Stability: It is generally asserted that as a first approximation, the fold of an Nucleic Acid molecule corresponds to a nested pairing that is the most stable. In the simple model, we measure the stability of a nested pairing by the number of matched pairs it has. So, the most stable nested pairing is the one with the largest number of matched pairs. This leads to the following computational problem.

The Simple Nucleic Acid Folding Problem: Given the nucleotide sequence S of a Nucleic Acid molecule, find a nested pairing that pairs the maximum number of nucleotides, compared to any other nested pairing.

1. Formulate an integer linear program for solving the Simple Nucleic Acid Folding Problem.

1.2 Simple Biological Enhancements

Incorporate the following constraints in your model.

A nested pairing of complementary nucleotides

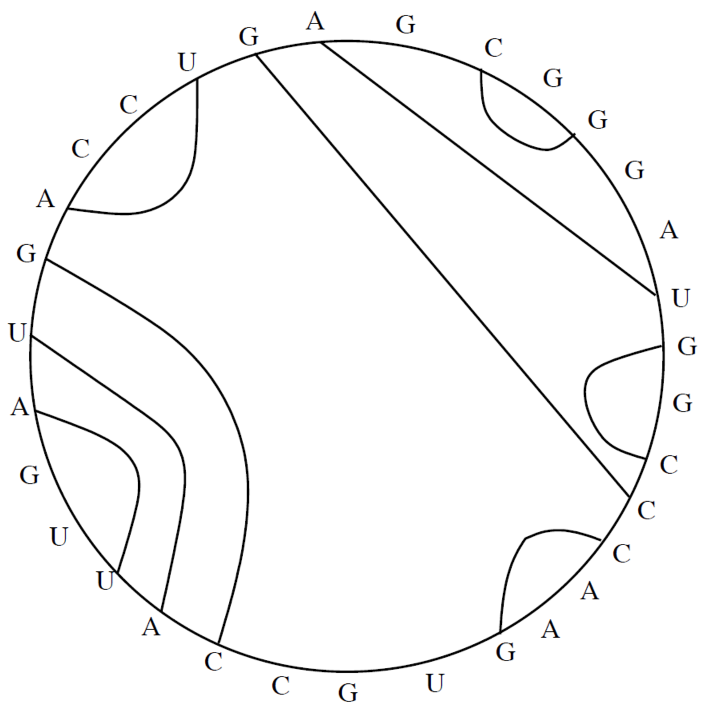

Figure 1: Non-crossing Pairing

1. Minimum distance constraint between paired positions, to avoid impossible bends. Any pair must be at least distance 3 away.

2. Differential binding strengths. The binding strength of a $\{C, G\}$ pair is larger than the binding strength of an $\{A, U\}$ pair, since a $\{C, G\}$ pair has three hydrogen bonds, while an $\{A, U\}$ pair only has two bonds. So, to find the most stable nested pairing, we need a maximum weight nested pairing. Assign the weight of three for each $\{C, G\}$ pair and weight of two for each $\{A, U\}$ pair.

3. Allowing non-complementary matched pairs. In some models of Nucleic Acid folding, certain non-complementary pairs of characters are allowed to form matching pairs, as long as appropriate weights, or multipliers, are used in the objective function for each allowed pair. The most commonly allowed non-complementary pair is $\{G, U\}$. Assign weight of 0.1 to pair $\{G, U\}$ and 0.05 to pair $\{A, C\}$ and zero to rest of the pairs. Update your model with these constraints.

1.3 More Complex Biological Enhancements

1. **Base Stacking - stems - helices.** A matched pair (i, j) in a nested pairing is called a stacked pair if either $(i+1, j-1)$ or $(i-1, j+1)$ is also a matched pair in the nested pairing. A stack in a nested pairing consists of a consecutive run of two or more stacked pairs. If (i, j) and $(i+1, j-1)$ are stacked pairs, the four positions $(i < i+1 < j-1 < j)$ is called a stacked quartet. In the above example, there is a stack of three pairings which leads to two stacked quartets.

   Stacking contributes significantly to the stability of an Nucleic Acid fold. So a more realistic ILP formulation for Nucleic Acid folding, must encourage paired nucleotides to be organized into (long-ish) stacks as much as possible. As a simple first step, extend the objective function of the ILP by including a count of the number of stacked quartets in the nested pairing. Thus the new objective is total weight of pairings as calculated above plus the number of stacked quartets. Update your model accordingly.

2. **Weighting stacked quartets in a nested pairing.** The next, and perhaps the most important, extension of the chemical model is to incorporate weights into the objective function for each stacked quartet in a stack. Then the objective function for the quartets is given as

$$\sum_{i=1}^{n} W(i,j)Q(i,j)$$

   where $W(i, j)$ is a positive constant that depends on which four nucleotides are in the stacked quartet $(i, i+1, j, j-1)$.

   Extensive chemical studies have been done to determine good weights for stacked quartets, based on the specific nucleotides that the stacked quartet contains. The following table shows the weights for stacked quartets. Update your model to incorporate these weights. The objective still involves the total weight of pairings as well.

	(A,U)	(C,G)	(G,C)	(U,A)
(A,U)	9	21	24	13
(C,G)	22	33	34	24
(G,C)	21	24	33	21
(U,A)	12	23	21	16

3. **Position of Quartet** In some models of nucleic folding, the weight given to stacked quartet depends both on the specific nucleotides in the quartet,and on where the stacked quartet is in a stack. The main distinction is whether the stacked quartet is the first quartet, the last quartet, or a middle quartet in a stack. Update your model where the weight of the first and last quartet is doubled as given in the table.

4. **Crossing Pairs.** So far we focused on pairs that do not cross. However, a limited number of crossing pairs are observed in Nucleic Acid structure. Update your model that allows at most $C = 10$ crossing pairs.

2 Report on Formulation (Due March 29)

A report should be written on describing your formulation for the above problem.

1. Introduction. Present the optimization problem as if I knew nothing about it beforehand.

2. Model. Describe the way you have chosen to model the problem. What are your variables, objective, and constraints? Do not just write out the formulation - explain each part, as if the reader was only slightly familiar with optimization. What is the size of your formulation as compared to the input size?

3 Final Report: Implementation and Analysis (Due April 21)

Solution and Analysis. Describe (in words as well as numbers) your optimal solution and the analysis of the solution (sensitivity, etc.) that you did. How does your model scale with the size of the input? Which of the constraints are the most complicated to solve? Did you update your model while implementing? Why and what updates did you do?
Your first report must be typed and at most 6 pages long. Use the answers to the questions given in the previous section to guide your write-up. Please submit your code along with the report.