Nucleic Acid Folding - A Mixed Integer Linear Programming Approach

Quill Healey
ISYE 3133 - Engineering Optimization - Final Project

March 31, 2022

Abstract

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 MILP formulation for a simplified version of the Nucleic Acid problem, and then extend the biological model and the MILP formulation to incorporate more realistic biological features of the problem.

Abstract taken from project problem statement.

1 Introduction

Note: Sets that are not defined are hyper-linked to the appendix.
Deoxyribonucleic acid (DNA) and ribonucleic acid (RNA), being the two molecules which are responsible for the storage and reading of genetic information that underpins all life, have motivated many questions in molecular biology. Specifically, being able to predict the varying tertiary structures of these two molecules allows scientists to understand and exploit their functionality. However, predicting the tertiary structure of a nucleic acid molecule is rather difficult, so often scientists turn toward an easier task: predicting its secondary structure. This simpler representation describes which pairs of bases are hydrogen bound together, which in turn provides a reasonable idea of which parts of the molecule are accessible to other nucleic acid molecules and biological structures. It is this secondary structure which we aim to provide an MILP formulation for.

In our case, given a sequence of nucleotides, predicting the secondary structure of the molecule is simply a matter of determining the interaction between each nucleotide's **nucleobase**. In RNA, these nucleobases are adenine (A), cytosine (C), uracil (U), and guanine (G). Note that for the remainder of this report we will represent each nucleotide with its

respective nucleobase's corresponding letter. That is, given a nucleotide sequence S, each element of the string will belong to the set $\{A, C, U, G\}$. Further examining S, we seek a set of disjoint nucleotide pairs, known as a **pairing**, of this sequence. Our pairing must also obey some additional biological constraints. Firstly, no singular nucleotide can be in more than one pair, and some nucleotides might not be in any pair. Additionally, we will also require that our pairing is **non-crossing** (alternatively called a **nested pairing**). That is, if we draw S as a circular string and let a line between two characters denote a hydrogen bond between those two respective nucleotides, a pairing is non-crossing if none of these lines cross each other. Finally, we will start by enforcing that all pairs are **complementary**. That is, a pair must either be comprised of the nucleobases adenine and uracil $\{A, U\}$ or cytosine and guanine $\{C, G\}$.

As with all constrained optimization problems, we must also identify the objective of our model. It is generally asserted that as a first approximation, the secondary structure of a nucleic acid molecule corresponds to a nested pairing that is most stable. **Fold stability** can vary in definition, but we will begin by assuming that the most stable folding is the one with the largest number of nested pairs. This objective leads to our first computational problem.

The simple nucleic acid folding problem: Given a 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.

2 Model

Before formulating the simple nucleic acid folding problem, we will define some common mathematical objects that all the following enhanced models will also utilize.

Recall that S denotes a string of n characters made up of the nucleic acid alphabet $\{A, C, U, G\}$ (we will index characters within the nucleotide sequence with the following notation: $S[i] \in \{A, C, U, G\}$, $\forall\, i \in \{1, 2, \ldots, n\} = \Omega$). Because at the heart of each model we have to determine which nucleotides pair with one another, we need to define decision variables for all possible pairs. Therefore, we define

$$x_{ij} = \begin{cases} 0, & \text{if pair is not chosen} \\ 1 & \text{if pair is chosen} \end{cases} \quad , \quad \forall\, i \Phi j \quad \text{as the directed connection from } S[i] \text{ to } S[j]$$

where Φ is a relation on Ω, which constrains the connections to disjoint, complementary, and non-recursive connections. Note that in our model, a **pair's order does matter**. For instance, we will treat an adenine-uracil pair as different from a uracil-adenine pair, thus each of these pairs get a decision variable to represent it.

Our choice of decision variables leads to a model which grows worst-case quadratic in size as we increase the length of the input nucleotide sequence S. To see this, let's first look at a simplified case where the decision variables can also represent potential connections between non-complementary base pairings. For this example, given a sequence S of length n, we have $n \times (n - 3)$ decision variables ($n - 3$ because we do not define decision variables between a character and itself, or its neighbors). While we will also restrict the decision variables

in our model to be between solely disjoint, complementary bases, one can imagine this will reduce the number of connections from nucleotide $S[i]$, $\forall\, i \in \Omega$ by some arbitrary $m_i \in \mathbb{Z}$, resulting in $O\left((n \times n) - \sum_i m_i\right) = O(n^2)$.

2.1 The First Crude Model

We will now construct the simple nucleic acid folding problem, taking special care to define our objective and constraints as all enhanced models will simply inherit or slightly modify this base reasoning.

Objective: Maximize the number of nested pairs. As we will deal with the "nested" aspect of this objective with our constraints, our goal is simply to maximize the number of connections. A connection, recall, is simply defined by the decision variable x_{ij}. Therefore, mathematically:

$$max \quad \sum_{i:\,(i,j)\in\Phi} \sum_{j:\,(i,j)\in\Phi} x_{ij} \tag{1}$$

Singular Pair Constraint: Given any ordered pair $(S[i], S[j])$, denoted by $x_{ij} = 1$, we must ensure that each nucleotide ($S[i]$ and $S[j]$) is not used again in any other pair. Thus establishing the following constraints. All other connections from $S[i]$ must equal zero: $x_{ij} = 1 \Rightarrow x_{ik} = 0\, \forall\, k : (i,k) \in \Phi \setminus \{(i,j)\} = \Phi_2$. All other connections to $S[i]$ must equal zero (including x_{ij}'s complement x_{ji}): $x_{ij} = 1 \Rightarrow x_{ki} = 0\ \forall k : (k,i) \in \Phi$. All other connections to $S[j]$ must equal zero: $x_{ij} = 1 \Rightarrow x_{lj} = 0\ \forall l : (l,j) \in \Phi_2$. All other connections from $S[j]$ must equal zero: $x_{ij} = 1 \Rightarrow x_{jl} = 0\ \forall l : (j,l) \in \Phi$. We can formalize these constrains using Big-M notation and the corresponding "if-then" transformation $\forall (i,j) \in \Phi$.

$$\begin{aligned} &\sum_{k:(i,k)\in\Phi_2} x_{ik} + \sum_{k:(k,i)\in\Phi} x_{ki} + \sum_{l:(l,j)\in\Phi_2} x_{lj} + \sum_{l:(j,l)\in\Phi} x_{jl} \ \leq M_1 z_{ij}, \\ &x_{ij} \leq M_2(1 - z_{ij}), \\ &z_{ij} \in \{0,1\} \end{aligned} \tag{C1}$$

where M_1 can be set to two and M_2 can be set to 1.

Non-Crossing Constraint: We will now ensure that our model builds a *nested* pairing. Picturing two characters, $S[i]$ and $S[j]$, we can imagine that the non-crossing constraint dictates that if $x_{ij} = 1$ or $x_{ji} = 1$ then no character between $S[i]$ and $S[j]$ can connect to any character outside of $S[i]$ and $S[j]$, or vice versa. Therefore, defining Ψ_{ij} to be the set of characters between the arbitrary $S[i]$ and $S[j]$ characters, and $\Theta_{ij} = \Omega \setminus \Psi_{ij} \setminus \{i,j\}$, we can guarantee a nested pairing once again using Big-M notation and the corresponding "if-then" transformation $\forall (i,j) \in \Phi$.

$$\begin{aligned} &\sum_{l\in\Psi_{ij}} \sum_{k\in\Theta_{ij}\cap\{k:(l,k)\in\Phi\}} x_{lk} + \sum_{k\in\Theta_{ij}} \sum_{l\in\Psi_{ij}\cap\{l:(k,l)\in\Phi\}} x_{kl} \ \leq M_1 w_{ij}, \\ &x_{ij} \leq M_2(1 - w_{ij}), \\ &w_{ij} \in \{0,1\} \end{aligned} \tag{C2}$$

where M_2 is trivially set to one and M_1 to $\lceil n/2 \rceil$, which is a slight over-approximation of the maximum number of pairs in an ideal nucleotide sequence.

2.2 Simple Biological Enhancements

2.2.1 Minimum Distance Constraint

Recall that we previously required that pairs of nucleotides be disjoint, that is, a minimum of two positions away. To require that pairs are now a distance of three positions away requires just a few updates to our previously defined Ω_i:

$$\Omega_i = \begin{cases} \{n-1, n, 1, 2, 3\}, & \text{if } i = 1 \\ \{n, 1, 2, 3, 4\}, & \text{if } i = 2 \\ \{n-2, n-1, n, 1, 2\}, & \text{if } i = n \\ \{n-3, n-2, n-1, n, 1\}, & \text{if } i = n-1 \\ \{i-2, i-1, i, i+1, i+2\} & \text{if } i \in \{3, \ldots, n-2\} \end{cases}$$

Note that our relation $\Phi \subseteq \Omega \times \Omega$ keeps its former definition: $\Phi = \{(i, j) \in \Omega \times \Omega : j \notin \Omega_i, (S[i], S[j]) \in \curlywedge\}$. Likewise, the decision variables, objective, and constraints keep their former definitions.

2.2.2 Differential Binding Strengths

In order for our model to reflect the varying binding strengths we simply need to add a weight function to our objective. Fortunately this only requires that we utilize the data already contained with the nucleotide sequence, meaning we do not have write any formal logic with our decision variables. We will define the matrix $D \in \mathbb{Z}^{n \times n}$ where

$$D_{ij} = \begin{cases} 3, & \text{if } S([i], S[j]) \in \{(G, C), (C, G)\} \\ 2, & \text{if } S([i], S[j]) \in \{(A, U), (U, A)\} \\ 0, & \text{o.w.} \end{cases}$$

From here were merely have to update our objective to finish updating the model.

$$max \quad \sum_{i:\,(i,j)\in\Phi} \sum_{j:\,(i,j)\in\Phi} D_{ij} x_{ij} \tag{2}$$

2.2.3 Non-complementary Base Pairs

To allow some non-complementary base pairs we will once again simply update our relation. In particular, we will add the non-complementary pairs to $\curlywedge_2$ (we will create a "derivative" of the original $\curlywedge$ as we will want to still keep the solely complementary pairs for later updates to the model):

$$\curlywedge_2 = \{(A, U), (C, G), (G, C), (U, A), (G, U), (A, C)\}$$

We will also update the relation to be $\Phi = \{(i, j) \in \Omega \times \Omega : j \notin \Omega_i, (S[i], S[j]) \in \curlywedge_2\} \subseteq \Omega \times \Omega$. Our subsequent decision variables and their constraints will once again keep their previous

definitions, however, we do need to update the objective function to account for the binding strengths of the new pairs. To make this change we will simply update D_{ij} :

$$D_{ij} = \begin{cases} 3, & \text{if } S([i], S[j]) \in \{(G,C),(C,G)\} \\ 2, & \text{if } S([i], S[j]) \in \{(A,U),(U,A)\} \\ 0.1, & \text{if } S([i], S[j]) \in \{(G,U)\} \\ 0.05, & \text{if } S([i], S[j]) \in \{(A,C)\} \\ 0 & \text{o.w.} \end{cases}$$

2.3 More Complex Biological Enhancements

Reading through the following enhancements we can foresee problems arising due to the fact that for some x_{ij}, $(i,j) \in \Phi$ the decision variables $x_{i+1,j-1}$ or $x_{i-1,j+1}$ might not exist due to our relation Φ. Consequently, for the following enhanced models we will redefine the decision variables as

$$x_{ij} = \begin{cases} 0, & \text{if pair is not chosen} \\ 1 & \text{if pair is chosen} \end{cases} \quad , \quad \forall\ (i,j) \in \Omega \times \Omega$$

Of course by eliminating the relation we have also eliminated many of the constraints necessary for our base folding problem. However, this is an easy fix. Define a new constraint to be:

$$x_{ij} = 0 \quad \forall\ (i,j) \notin \Phi \tag{C3}$$

This of course is valid logic as our relation is still built only with data, thus does not require any rigorous MIP formulation.

2.3.1 Base Stacking - Stems - Helices

In order to maximize the number of stacked quartets we are trying to maximize the number of cases where $x_{ij} = 1$ and $x_{i+1,j-1} = 1$. Therefore, we will let $Q \in \mathbf{B}^{n \times n}$ where $x_{ij} = 1$ and $x_{i+1,j-1} = 1$ implies that $Q_{ij} = 1$ (initially define all terms in the matrix to be zero and only change them if the linear constraints enforce a change). We can represent this statement using the following "if-then" transformation: $f(x) = x_{ij} + x_{i+1,j-1} - 1 > 0$ then $g(x) = Q_{ij} - 1 \geq 0$. To update our model we then simply change our objective to:

$$max \quad \sum_{i \in \Omega} \sum_{j \in \Omega} D_{ij} x_{ij} + \quad \sum_{i \in \Omega} \sum_{j \in \Omega} Q_{ij} \tag{3}$$

and add the formal "if-then" constraints

- $\forall (i,j) \in \Omega : i \neq n, j \neq 1$:

$$\begin{aligned} & 1 - Q_{ij} \leq M_1 y_{ij}, \\ & x_{ij} + x_{i+1,j-1} - 1 \leq M_2(1 - y_{ij}), \\ & Q_{ij} \leq 1, \\ & y_{ij} \in \{0,1\} \end{aligned} \tag{C4}$$

- $\forall (i,j) \in \Omega : i = n, j \neq 1$:

$$\begin{aligned} & 1 - Q_{ij} \leq M_1 y_{ij}, \\ & x_{ij} + x_{1,j-1} - 1 \leq M_2(1 - y_{ij}), \\ & Q_{ij} \leq 1, \\ & y_{ij} \in \{0,1\} \end{aligned} \tag{C5}$$

- $\forall (i,j) \in \Omega : i \neq n, j = 1$:

$$\begin{aligned} & 1 - Q_{ij} \leq M_1 y_{ij}, \\ & x_{ij} + x_{i+1,n} - 1 \leq M_2(1 - y_{ij}), \\ & Q_{ij} \leq 1, \\ & y_{ij} \in \{0,1\} \end{aligned} \tag{C6}$$

where we can set $M_1 = M_2 = 1$.

2.3.2 Weighting Stacked Quartets in a Nested Pairing

Updating our quartet objective to include weights is rather trivial as the weight function itself is purely dependent on the nucleotide sequence that we are given. Or in other words, we do not have to write any formal logic. Nonetheless, there still is some work to be done. Firstly, define a new nucleotide sequence S' to help with the circular nature of our string:

$$S' = \begin{cases} S[i], & \text{if } i \in \Omega \\ S[n], & \text{if } i = 0 \\ S[1], & \text{if } i = n+1 \end{cases}$$

Now, let T_{lk} represent entries in the project statement's stacked quartet weight table, where $(l,k) \in \urcorner \times \urcorner$. Finally, we define $W \in \mathbb{Z}^{n \times n}$ where

$$W_{ij} = \begin{cases} T_{(S[i],S[j]),(S'[i+1],S'[j-1])}, & \text{if } ((S[i],S[j]),(S'[i+1],S'[j-1])) \in \urcorner \times \urcorner \\ 0 & o.w. \end{cases}$$

$\forall (i,j) \in \Omega \times \Omega$. Thus we can define our new objective function as

$$max \quad \sum_{i \in \Omega} \sum_{j \in \Omega} D_{ij} x_{ij} + \quad \sum_{i \in \Omega} \sum_{j \in \Omega} W_{ij} Q_{ij} \tag{4}$$

2.3.3 Position of Quartet

Our final update to the objective will be to reward quartet positioning (**Important Note:** Because there was no clear specifications I'm making the following assumptions about first, middle, and last. When there is only one quartet it will be considered a first. When there are two stacked quartets there is a top and bottom. Any stack with more than two quartets will have one or more middle quartets). In order to do so, we will introduce two new matrices:

$L_{ij}, F_{ij} \in \mathbb{Z}^{n\times n}$. Once again set each entry in both matrices equal to 0 and then only update them if the constraints enforce the change. The former will be used to detect if a quartet is the last quartet while the latter will be used to determine if a quartet is the first quartet (note that Q now detects middle quartets).
To detect and reward a quartet for being the first in a stack the following logic will be used: if $x_{ij} = 1$ and $x_{i+1,j-1} = 1$ and $x_{i-1,j+1} = 0$ then $F_{ij} = 2$.
Let's write this formally with an "if-then" MIP transform (note that I adjust for circularity for $i = n$ and $j = 1$, but not for the other cases. This logic can easily be inferred and implemented in Python - writing them has become too time consuming):

- $\forall (i,j) \in \Omega : i \neq n, j \neq 1$:

$$\begin{aligned} &2 - F_{ij} \leq M_1 u_{ij} \\ &1 - x_{i-1,j+1} \geq M_2(1 - u_{ij}) \\ &x_{ij} + x_{i+1,j-1} - 1 \leq M_3(1 - u_{ij}) \\ &F_{ij} \leq 2 \\ &u_{ij} \in \{0,1\} \end{aligned} \tag{C7}$$

- $\forall (i,j) \in \Omega : i = n, j \neq 1$:

$$\begin{aligned} &2 - F_{ij} \leq M_1 u_{ij} \\ &1 - x_{i-1,j+1} \geq M_2(1 - u_{ij}) \\ &x_{ij} + x_{1,j-1} - 1 \leq M_3(1 - u_{ij}) \\ &F_{ij} \leq 2 \\ &u_{ij} \in \{0,1\} \end{aligned} \tag{C8}$$

- $\forall (i,j) \in \Omega : i \neq n, j = 1$:

$$\begin{aligned} &2 - F_{ij} \leq M_1 u_{ij} \\ &1 - x_{i-1,j+1} \geq M_2(1 - u_{ij}) \\ &x_{ij} + x_{1,n} - 1 \leq M_3(1 - u_{ij}) \\ &F_{ij} \leq 2 \\ &u_{ij} \in \{0,1\} \end{aligned} \tag{C9}$$

where $M_1 = 2$, $M_2 = 1$, $M_3 = 1$.
Likewise, to detect and reward a quartet for being the last in a stack the following logic will be implemented: $x_{ij} = 1$ and $x_{i+1,j-1} = 1$ and $x_{i+2,j-2} = 0$ and $x_{i-1,j+1} = 1$ (this last constraint is necessary to ensure that we do not count a single quartet as both a first and last, but rather just a first) then $L_{ij} = 2$.
Let's write this formally using more "if-then" constraints (note that from here on out will no longer account for circular nature of string for reasons stated above). $\forall (i,j) \in \Phi : (i,j) \notin$

$\{(1,n),(n,1)\}$

$$\begin{aligned} & 2 - L_{ij} \leq M_1 v_{ij} \\ & 1 - x_{i+2,j-2} \geq M_2(1 - v_{ij}) \\ & x_{ij} + x_{i+1,j-1} + x_{i-1,j+1} - 2 \leq M_3(1 - v_{ij}) \\ & L_{ij} \leq 2 \\ & v_{ij} \in \{0,1\} \end{aligned} \tag{C10}$$

where $M_1 = 2, M_2 = 1$, and $M_3 = 1$.
Finally, we need to adjust Q_{ij} to only account for middle quartets. This can be tracked with the following logic: if $x_{ij} = 1$ and $x_{i+1,j-1} = 1$ and $x_{i-1,j+1} = 1$ and $x_{i+2,j-2} = 1$ then $Q_{ij} = 1$. Formally written, $\forall (i,j) \in \Phi : (i,j) \notin \{(1,n),(n,1)\}$

$$\begin{aligned} & 1 - Q_{ij} \leq M_1 q_{ij} \\ & x_{ij} + x_{i+1,j-1} + x_{i+2,j-2} + x_{i-1,j+1} - 3 \leq M_2(1 - q_{ij}) \\ & Q_{ij} \leq 1 \\ & q_{ij} \in \{0,1\} \end{aligned} \tag{C11}$$

where $M_1 = 1$ and $M_2 = 1$. Note that this constraint replaces constraints (C4 - C6).
Finally we will update our objective to reflect these new position treatments.

$$max \quad \sum_{i \in \Omega} \sum_{j \in \Omega} D_{ij} x_{ij} + \quad \sum_{i \in \Omega} \sum_{j \in \Omega} W_{ij} Q_{ij} + \quad \sum_{i \in \Omega} \sum_{j \in \Omega} W_{ij} F_{ij} + \quad \sum_{i \in \Omega} \sum_{j \in \Omega} W_{ij} L_{ij} \tag{5}$$

2.3.4 Crossing Pairs

The following constraint in non-linear. I feel confident that if I had started earlier (or had more time in general) then I would have been able to solve it. We will once again utilize the interior and exterior sets we created for the previous non-folding constraint, Ψ_{ij} and Θ_{ij}. Now, our aim is to follow the logic $\forall x_{ij} = 1$ the sum of crosses cannot be greater than 10. A non-linear solution can be put in place with the following constraint.

$$\sum_{i:(i,j) \in \Phi} \sum_{i:(i,j) \in \Phi} x_{ij} \left(\sum_{l \in \Psi_{ij}} \sum_{k \in \Theta_{ij}} x_{lk} + \sum_{k \in \Theta_{ij}} \sum_{l \in \Psi_{ij}} x_{kl} \right) \leq 10 \tag{C12}$$

This constraint is rather pleasing: notice how if a connection is used then the crossings are counted, but if the connection goes unused then the crossings are not counted.

A Formulation Size

Recall that at the beginning of the formulation we declared that our initial crude model's decision variables were of size $O(n^2)$. Throughout the remainder of the report (excluding the final, non-linear constraint), we never create any constraints, decision variables, or objectives that loop through the input nucleotide sequence more than once. Consequently we can claim that the worst-case time complexity remains quadratic. This is rather good as most dynamic programming solutions are cubic in size.

B Appendix

B.1 Disjoint, Complementary, Non-Recursive Relation

When solving constrained optimization problems, a general rule of thumb is that a model's complexity increases with the number of decision variables that are declared. Consequently, we seek to reduce the number that we use. For the simple folding model we know that no nucleotide can connect to a neighboring nucleotide, itself, or a nucleotide with a non-complementary base. Consequently, as S is fixed data, we can eliminate any potential decision variables that fall within the aforementioned categories.

In order for our relation to restrict neighboring and recursive connections, we must first account for the fact that the nucleotide sequence is circular. That is, we must account for the neighbor edge cases $i-1$ and $i+1$ for $i=1$ and $i=n$, respectively. To do this, we declare a new set derived from Ω:

$$\Omega_i = \begin{cases} \{n, 1, 2\}, & \text{if } i = 1 \\ \{n-1, n, 1\}, & \text{if } i = n \\ \{i-1, i, i+1\} & \text{if } i \in \{2, \ldots, n-1\} \end{cases}$$

and create a set of allowed base pairings

$$\urcorner = \{(A, U), (C, G), (G, C), (U, A)\}.$$

Now we define our relation Φ:

$$\Phi = \{(i, j) \in \Omega \times \Omega : j \notin \Omega_i, (S[i], S[j]) \in \urcorner\} \subseteq \Omega \times \Omega$$

B.2 Inside Nucleotide Pair Set

$$\Psi_{ij} = \begin{cases} \{i+1, \ldots, n\}, & \text{if } j = 1 \\ \{1, \ldots, j-1\}, & \text{if } i = n \\ \{i+1, \ldots, j-1\} & o.w. \end{cases}$$