Welcome to the Webinar

Integer Linear Programming in Computational and Systems Biology

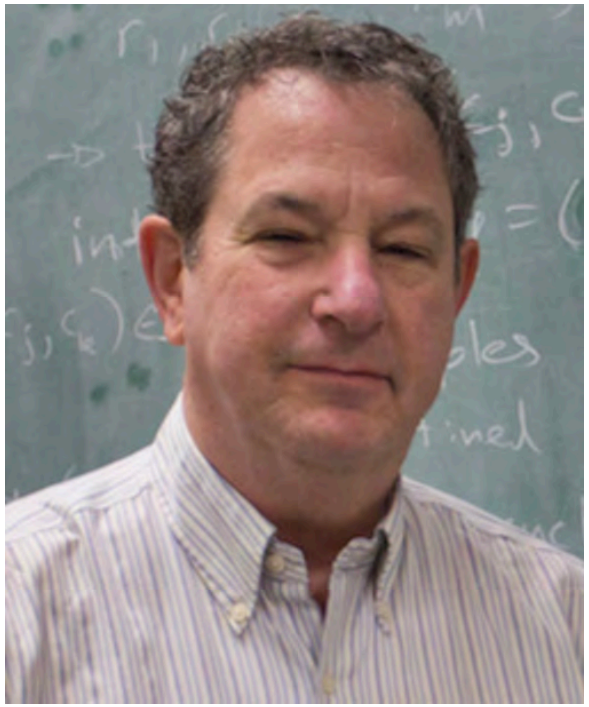
GUROBI
OPTIMIZATION

The World's Fastest Solver

Speaker Introduction

Dan Gusfield

- Distinguished Professor Emeritus in the department of computer science at the University of California, Davis
- Fellow of the IEEE, the ACM, and the International Society of Computational Biology
- Author of “Integer Linear Programming in Computational and Systems Biology: An entry-level text and course", recently published by Cambridge University Press.



Integer Linear Programming in Computational and Systems Biology

Dan Gusfield

Gurobi Webinar March, 2020

This webinar is adapted from a chapter in the book *Integer Linear Programming in Computational and Systems Biology: An entry-level text and course*, published by Cambridge University Press, 2019

I will assume that most attendees are familiar with ILP

Why Integer Programming in Computational Biology?

ILP is Increasingly used in *computational and systems biology* in *non-traditional* ways.

ILP is very often (but not always) effective in solving **instances** of hard biological problems.

Moreover, it is simpler to develop ILP formulations than to develop a special algorithm for each problem type.

Why ILP?

ILP is now used to address a wide range of problems in biological **science** - Distinct from biological **management**.
My interest is in ILP used for biological science.

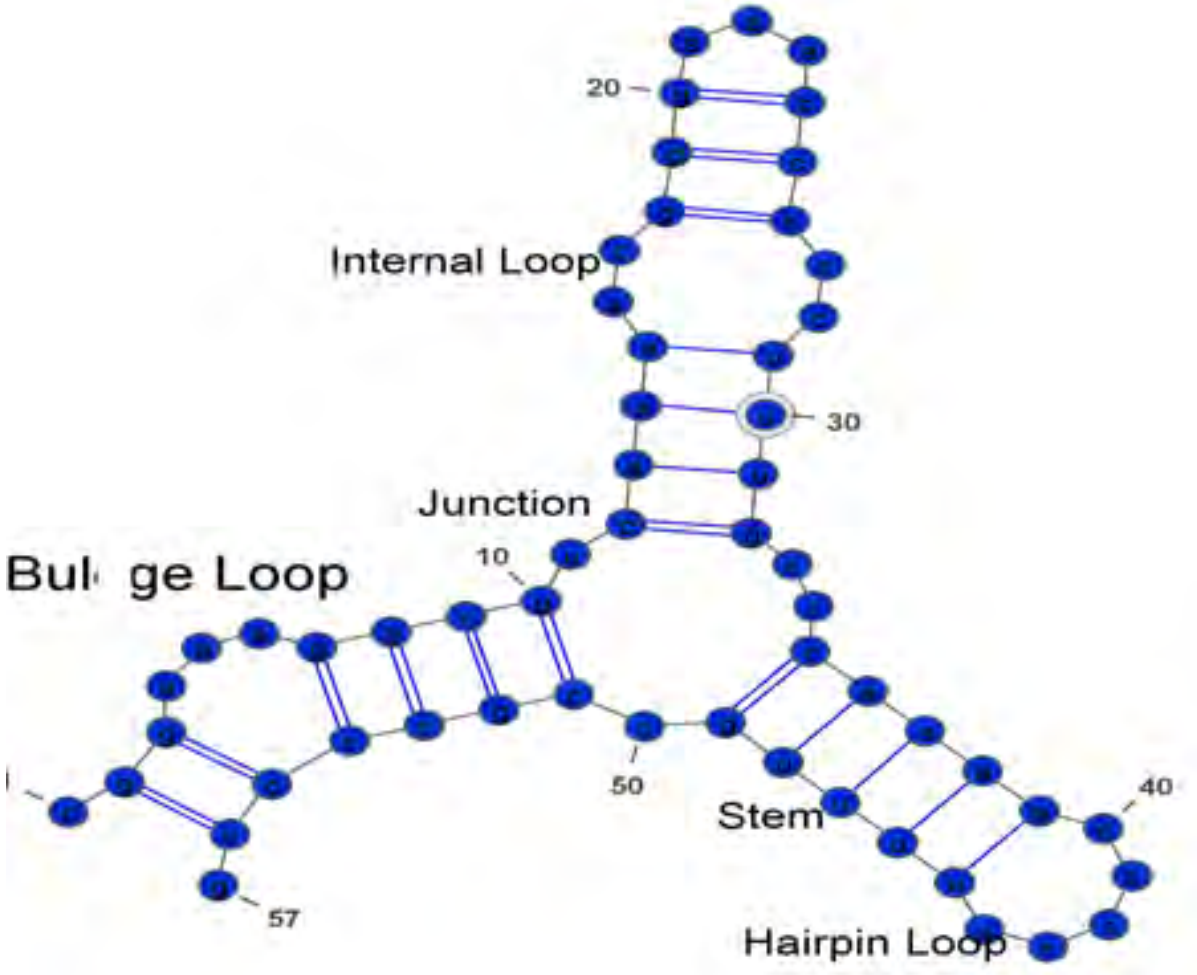
Exploiting optimization (via ILP), some biological problems can be modeled in a way that allows a solution in seconds on a laptop, while more common *(statistically-based)* models require days, weeks or months of computation on large clusters.

There is a growing literature (a few hundred papers over the last 15 years) using ILP in computational biology.

The 2-D RNA Folding Problem

I will illustrate the use of ILP to **model** and **solve** a particular class of problems - **Predicting RNA 2-D folding**, given only the RNA **sequence**.

A 2-D RNA Structure



RNA folding via ILP

Predicting 2-D structure is an important, classic problem in computational biology, that is often solved with variants of *dynamic programming*.

However, *integer linear programming* is easier (but slower).

More important, ILP can be extended to model more **complex** versions of the folding problem, that are difficult to model and solve with dynamic programming.

We start with an ILP formulation for a **simplified** version of the RNA problem.

A Crude First Model of RNA Folding

We are given a sequence S of characters (nucleotides) from the RNA alphabet $\{A, C, U, G\}$. For example, $S = ACGUGCCACGAU$.

A **pairing** is set of *disjoint* pairs of *positions* in S. A pair in a pairing is called a **matched pair**.

A position can be in at most one matched pair.

Note that some positions might not be in any matched pair in a pairing.

A Pairing

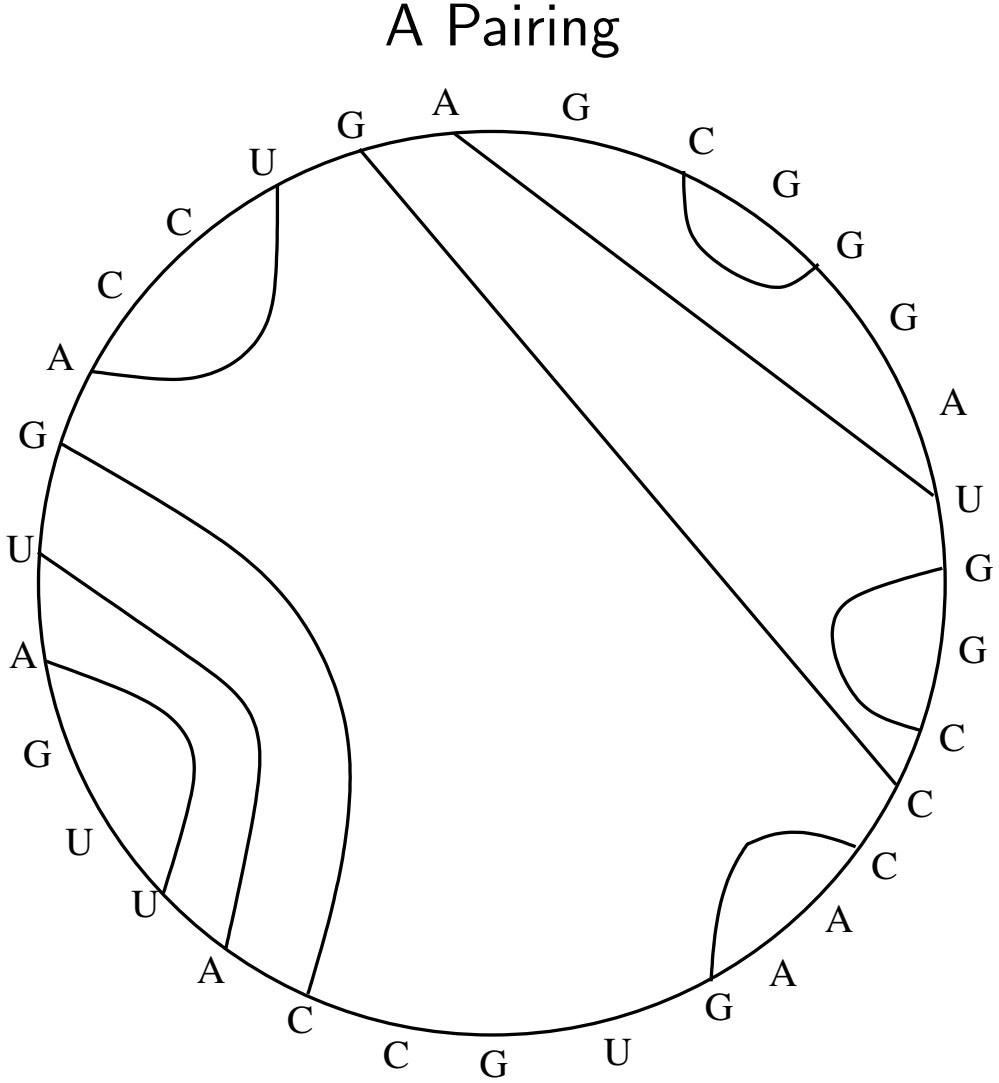


Figure : The lines show a pairing.

Crude First Model

Two characters are called **complementary** if they are: $\{A, U\}$ or $\{C, G\}$. In our first model, all matched pairs must have complementary characters.

A **nested pairing** (also called **non-crossing**) drawn on a circle: A pairing of *complementary* characters where each matched pair is connected by a line *inside* the circle, and where *none* of the lines cross.

A Nested Pairing

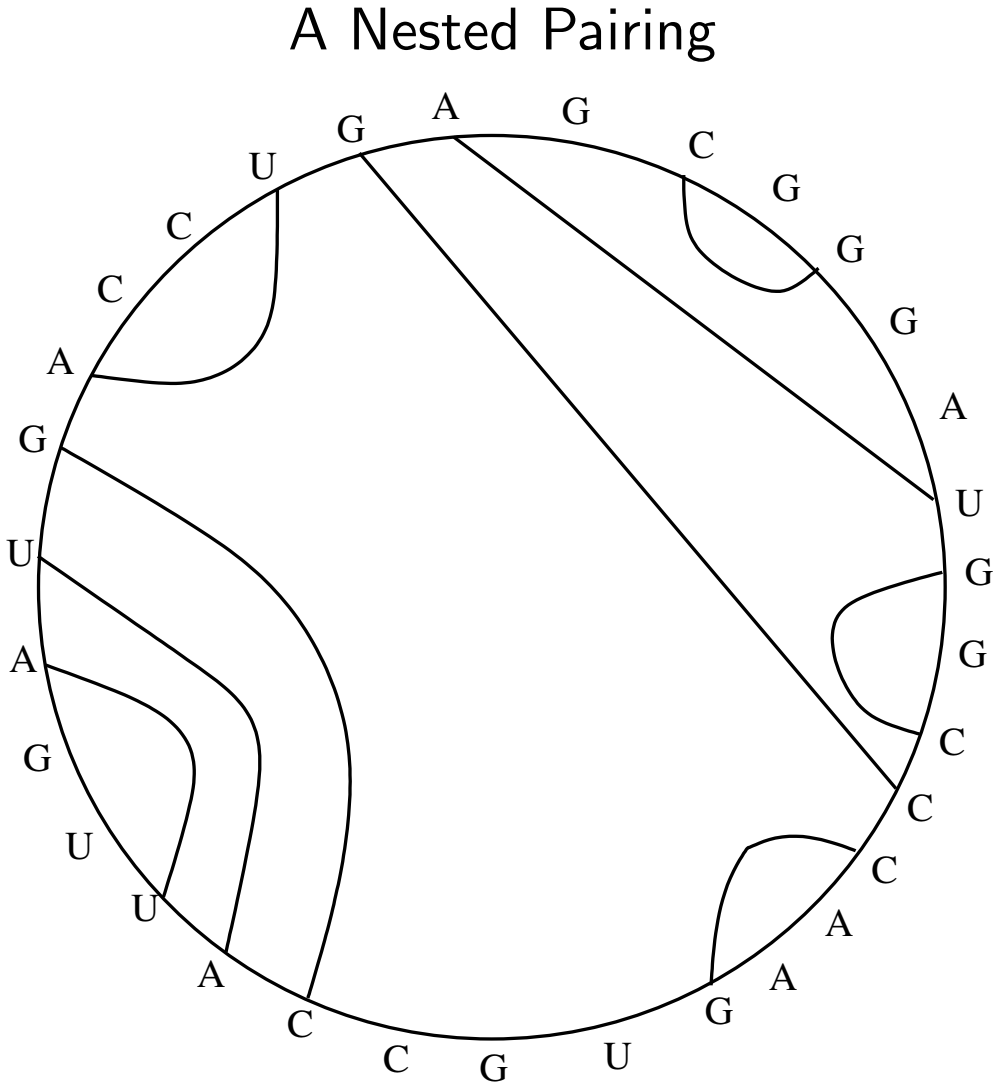
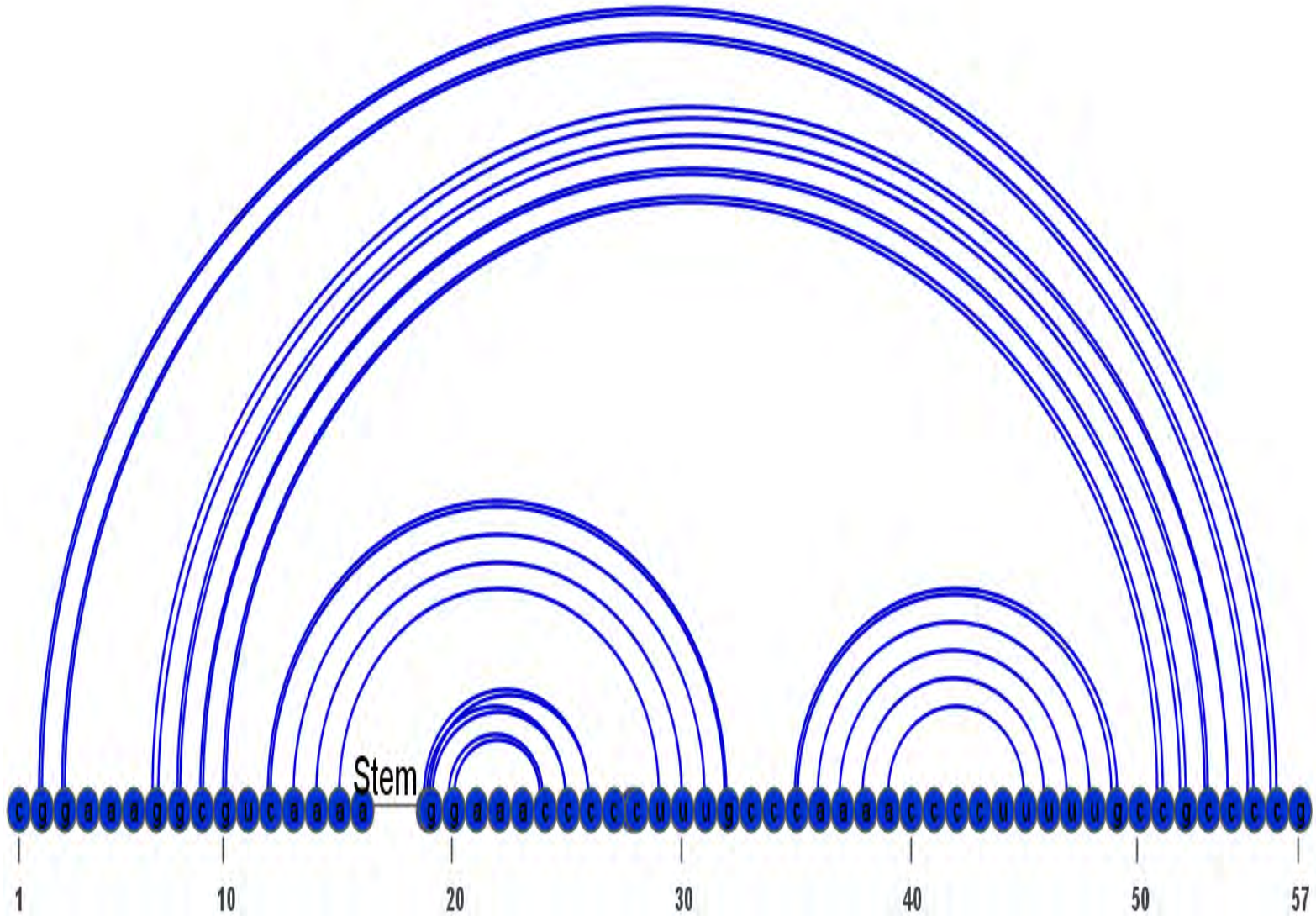


Figure : The lines show a nested pairing.

We Can Display a Nested Pairing on a Line



A Forbidden Non-Nested (Crossing) Pairing

A Simple RNA Structure

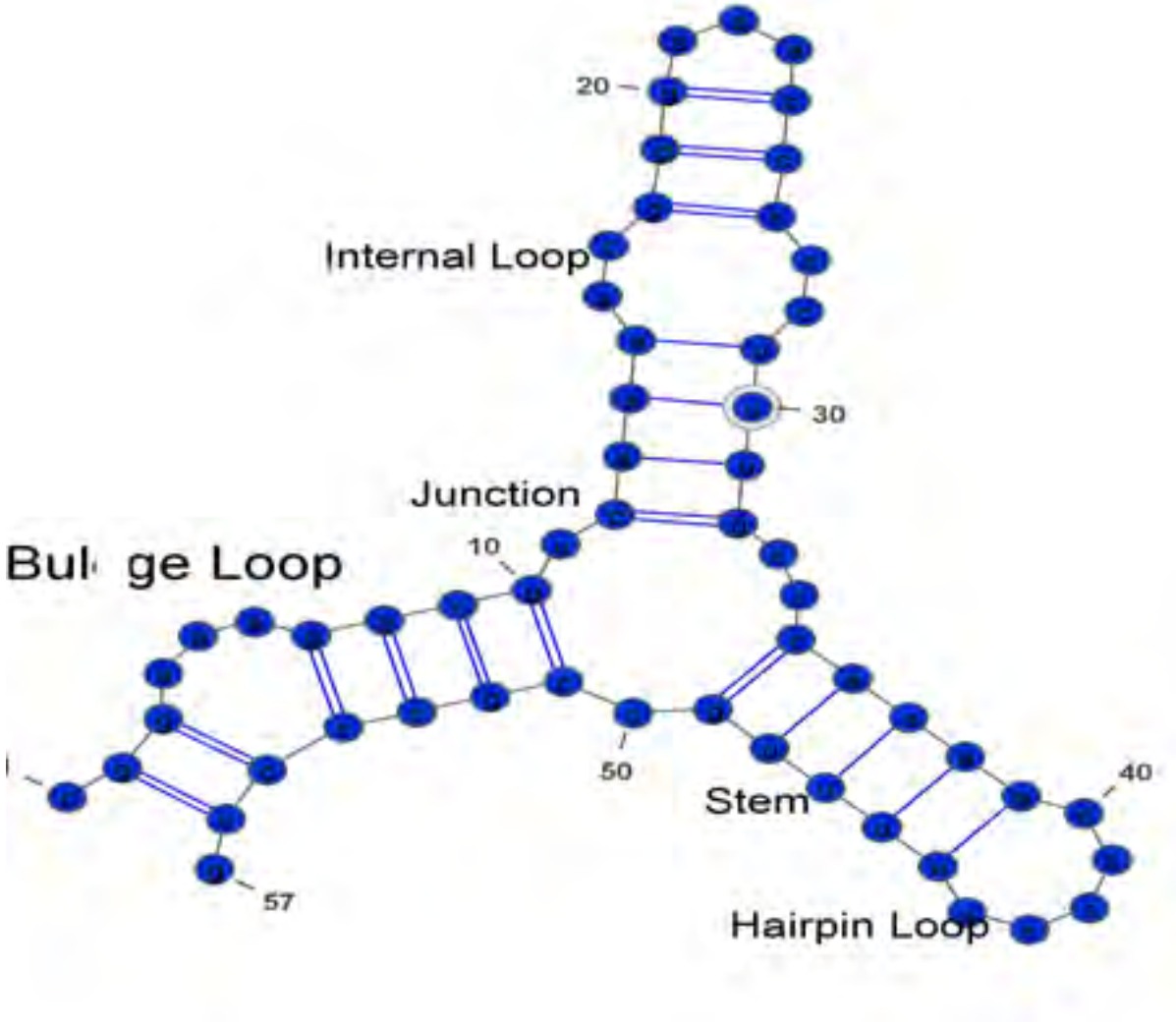


Figure : A Nested Pairing on the RNA Structure

Fold Stability

As a first approximation, it is generally accepted that the true 2-D fold of an RNA molecule *corresponds* to the **most stable** nested pairing.

In the simple model, we measure the stability of a nested pairing by the **number** of matched pairs it has. Then, the most stable nested pairing is the one with the *largest number* of matched pairs. So, we have

> **The Simple RNA Folding Problem:** *Given the sequence S of an RNA molecule, find a nested pairing with the maximum number of matched pairs.*

Formulating and Solving the Simple RNA Folding Problem via ILP

We create one **binary** ILP variable, $P(i, j)$, for each pair (i, j) of positions in S, where $i < j$.
$P(i, j)$ will be set to 1 if and only if position i is paired with position j.

If the ordered pair of characters in any positions i and j are *not* complementary, i.e., not (A, U) or (U, A) or (C, G) or (G, C), then we create and include the equality:

$$P(i, j) = 0,$$

to disallow the pairing of the characters in positions i and j.

Each site in at most one matched pair

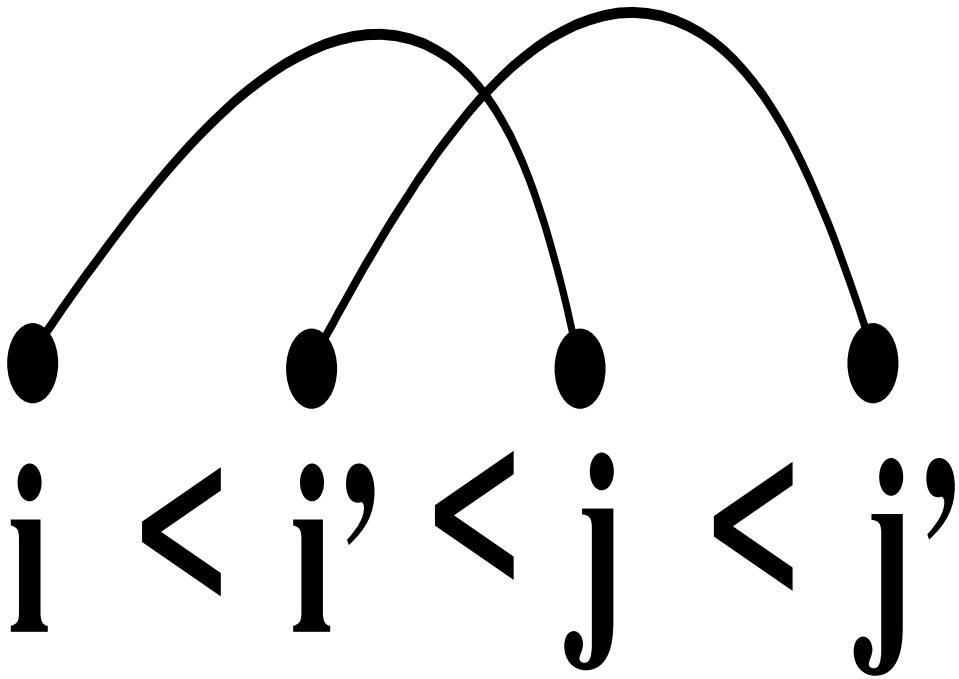
The requirement that each character can be in *at most* one matched pair:

$$\text{for each j} \quad \sum_{k>j} P(j,k) + \sum_{k<j} P(k,j) \leq 1. \tag{1}$$

Note that this is an *inequality*, not an *equality*, meaning that it is permissible for a position j to *not* be in any matched pair.

Implementing Nested Pairing

To implement the requirement that the pairing be *nested*, the key is to note that a pairing that is **not** nested, **must contain** some matched pairs (i, j) and (i', j') where $i < i' < j < j'$.



Crossing matched pairs

To Forbid Crossing Pairs

We have the inequalities:

For every four positions $i < i' < j < j'$,

$$P(i,j) + P(i',j') \leq 1 \tag{2}$$

Finally, the **objective function** is:

$$\text{Maximize} \quad \sum_{i<j} P(i,j)$$

A Toy Example

$S = ACUGU$. Then the ILP is:
Maximize $P(1,2) + P(1,3) + P(1,4) + P(1,5) + P(2,3) + P(2,4) + P(2,5) + P(3,4) + P(3,5) + P(4,5)$ s.t.

```
P(1,2) = 0
P(1,4) = 0
P(2,3) = 0
P(2,5) = 0
P(3,4) = 0
P(3,5) = 0
P(4,5) = 0

P(1,2) + P(1,3) + P(1,4) + P(1,5) <= 1
P(1,2) + P(2,3) + P(2,4) + P(2,5) <= 1
P(1,3) + P(2,3) + P(3,4) + P(3,5) <= 1
P(1,4) + P(2,4) + P(3,4) + P(4,5) <= 1
P(1,5) + P(2,5) + P(3,5) + P(4,5) <= 1
```

Continuing

```
P(1,3) + P(2,4) <= 1
P(1,3) + P(2,5) <= 1
P(1,4) + P(2,5) <= 1
P(1,4) + P(3,5) <= 1
P(2,4) + P(3,5) <= 1

All variables are binary.
```

Simple Biological Enhancements

- Minimum distance constraint between matched paired positions: Trivial to incorporate this constraint. Set $P(i, j) = 0$ if characters i and j are too close (e.g., within three positions).
- Differential binding strengths:
  A $\{C, G\}$ matched pair has three hydrogen bonds, while an $\{A, U\}$ matched pair only has two bonds.

  So, to find the most stable nested pairing, we need to *weight* matched pairs differently and find a **maximum weight** nested pairing.

  Trivial to incorporate weights into the objective function.

Simple Enhancements

Allowing non-complementary matched pairs:

In *some* models of RNA folding, a *small* number of certain **non-complementary** characters (mostly $\{G, U\}$) are allowed to form a matched pair.

If we use weights in the objective function, non-complementary matched pairs can be permitted, but discouraged, so they would appear in an optimal solution only if no nested pairing has as many matched pairs.

Simple Enhancements

Or, we can modify the ILP model to bound the number, or the percentage, of non-complementary matched pairs:

$$\sum_{(i,j)\in NCpairs} P(i,j) \leq c \times \sum_{(i,j)} P(i,j),$$

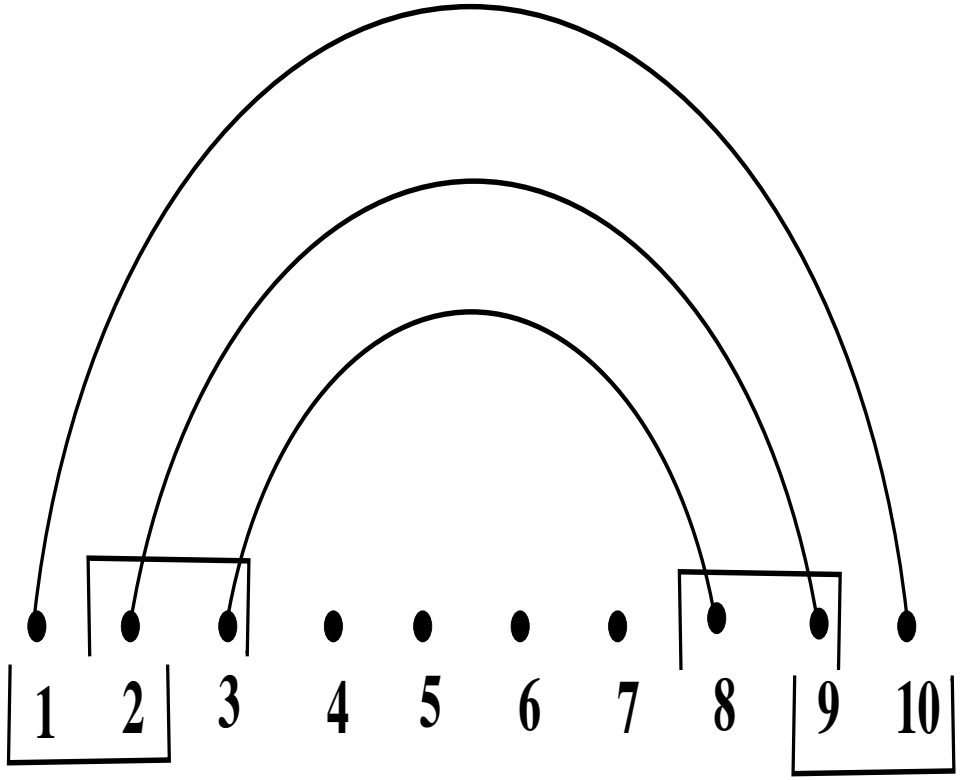
where *NCpairs* is the set of non-complementary pairs, and c is a chosen bound on the *fraction* of allowed non-complementary matched pairs.

More Complex Biological Enhancements

A matched pair (i, j) in a nested pairing is called a **stacked pair** if its neighboring positions $(i+1, j-1)$ also form a matched pair (or $(i-1, j+1)$ form a matched pair) in the nested pairing.

If (i, j) and $(i+1, j-1)$ are stacked pairs, the four positions $(i < i+1 < j-1 < j)$ form a **stacked quartet**.

A **stack** in a nested pairing consists of a **consecutive** run of two or more stacked pairs.



Stack with six positions, and two stacked quartets:

(1, 2, 9, 10) and (2, 3, 8, 9)

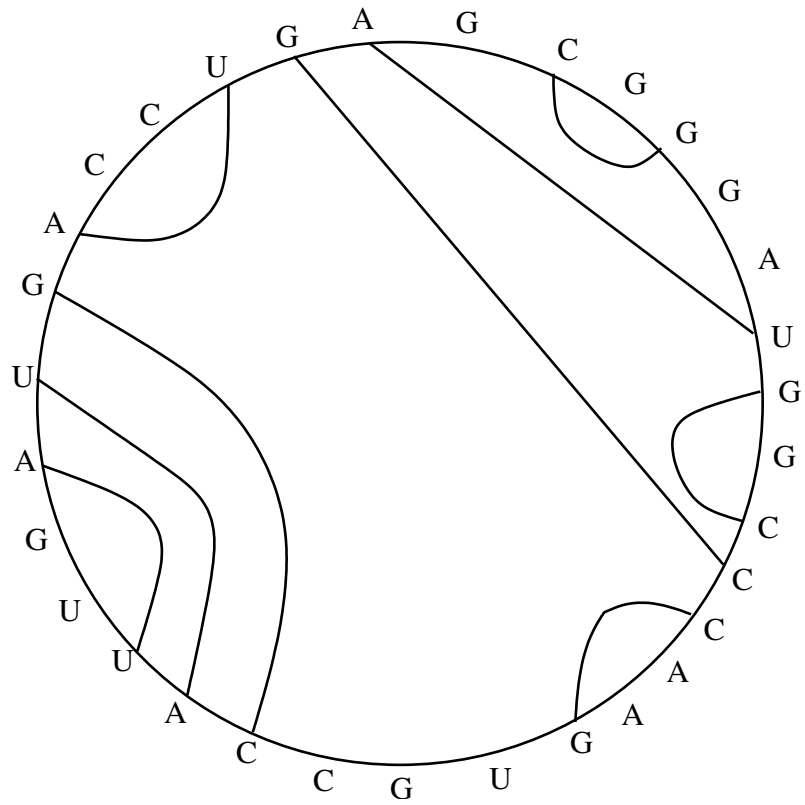
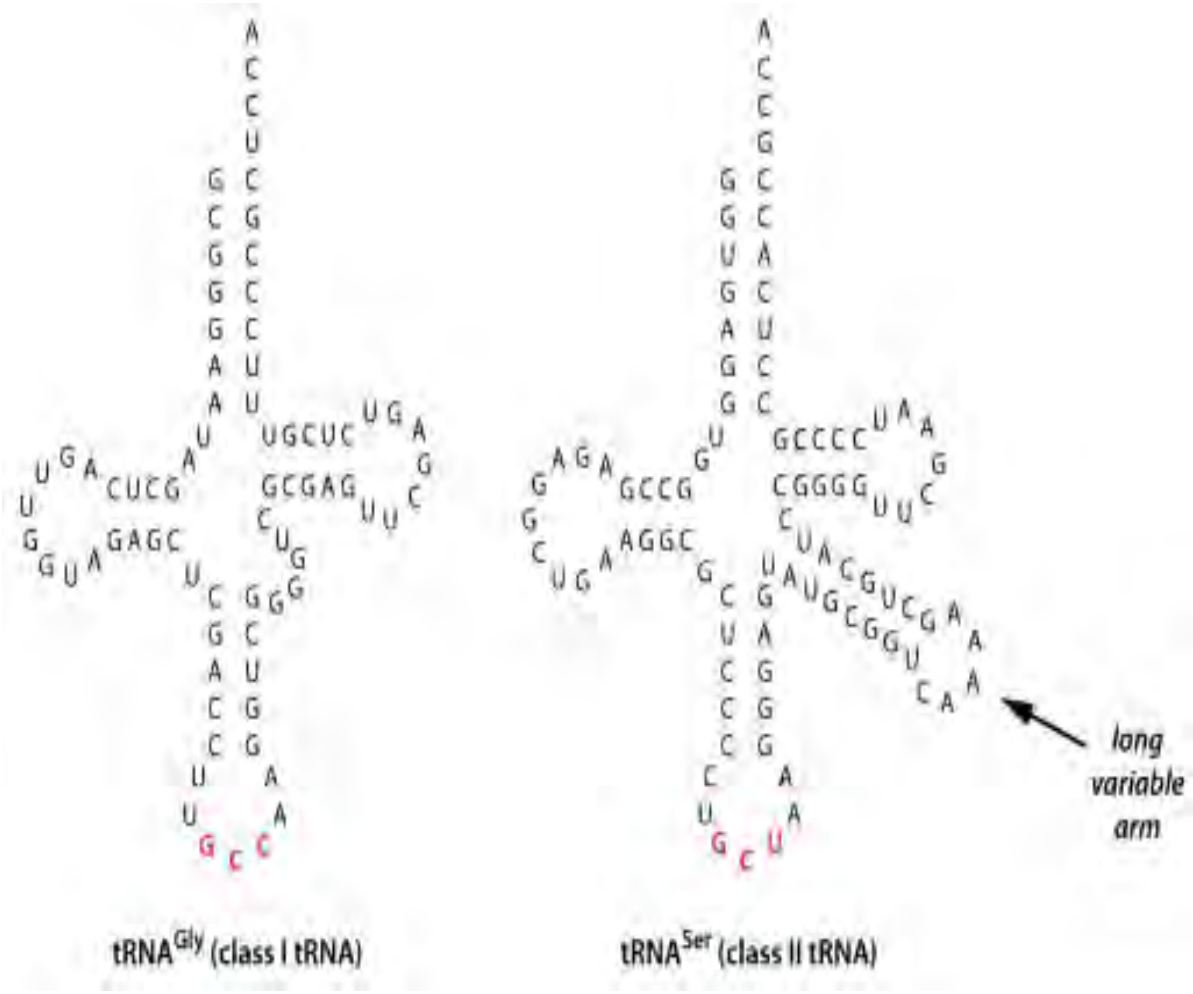


Figure : The pairing contains one stack with three matched pairs, and two stacked quartets.

Stacks are particularly evident in the folding of *transfer RNA (tRNA)*, in a distinctive secondary structure called a *cloverleaf*.



Stacks and Stability

Stacks contribute significantly to the stability of an RNA fold. So a more realistic ILP formulation for RNA folding must *encourage* matched paired characters to be organized into (long-ish) stacks.

As a simple first step, we will extend the objective function of the ILP to include a **count** of the number of **stacked quartets** in the nested pairing.

Counting Stacked Quartets

We create the binary ILP variable $Q(i,j)$ to indicate whether the matched pair (i,j) is the **first** pair in a stacked quartet. We have, for each i,j, where $j > i$:

$$P(i,j) + P(i+1,j-1) - Q(i,j) \leq 1 \tag{3}$$

$$2Q(i,j) - P(i,j) - P(i+1,j-1) \leq 0 \tag{4}$$

The first inequality enforces the condition that **if both** (i,j) and $(i+1,j-1)$ are in the nested pairing **then** the value of variable $Q(i,j)$ *must* be set to 1.

The second inequality enforces the converse condition, that $Q(i,j)$ is set to 1 **only if** $P(i,j)$ and $P(i+1,j-1)$ are **both** set to 1.

Counting Stacked Quartets

Then, to incorporate a count of the number of stacked quartets in the fold, we change the objective function from

$$\text{Maximize} \sum_{i<j} P(i,j)$$

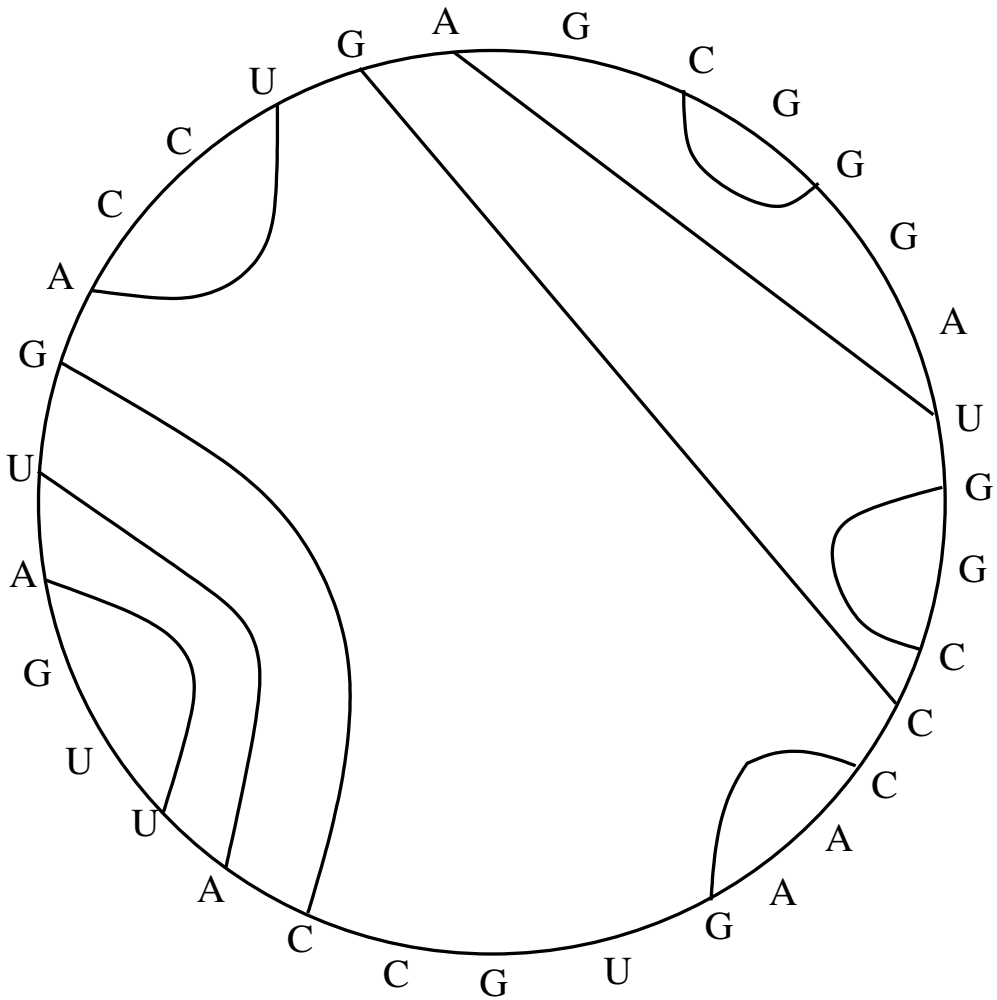
to

$$\text{Maximize} \sum_{i<j} [P(i,j) + Q(i,j)],$$

Or we can use weights to reflect the relative importance of stacked quartets compared to matched pairs.

How Do Stacked Quartets Encourage Stacks?

Stacked quartets **overlap** in a stack, so the same number of stacked quartets require *fewer* positions than if they were *separated*, i.e., not in a stack.



Weighting stacked quartets in a nested pairing

The next, and perhaps the most important, extension of the biological model is to incorporate **weights** into the objective function for *each* stacked quartet.
Then, the objective function is given as:

$$\text{Maximize} \sum_{i=1}^{i=n} [W(i,j) \times Q(i,j)],$$

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

Extensive studies have been done to determine informative weights for stacked quartets, based on **which** four characters in the stacked quartet.

Alternative Objective Functions

The objective function

$$\text{Maximize} \sum_{i<j} [P(i,j) + Q(i,j)]$$

is sometimes replaced with

$$\text{Maximize} \sum_{i<j} [Q(i,j)].$$

In fact, some biologists and biochemists think that this is the most biologically-meaningful objective function.

What is fascinating is that the ILP with this objective function solves much faster than the prior two, but the pairings are similar.

A Typical Example

In an example of length 100, with the minimum distance between matched paired characters set at five, and using the objective function with only P variables, Gurobi 7.5 took about 13 minutes and produced a nested pairing with 33 matched pairs and 14 quartets.

Then, including both P and Q variables, Gurobi took 51 seconds and produced a nested pairing with 32 matched pairs and 20 quartets.

But, using the objective function with only Q variables, Gurobi took only 2.16 seconds and produced a nested pairing with 29 matched pairs and 20 quartets.

Good Karma! The most biologically-informative objective function also solves the fastest.

Really Good Karma!

It is also fascinating to note that the RNA folding problem using the first objective function (only P variables in the objective function) has a **worst-case polynomial-time** solution. In fact, for a sequence of length n, the worst-case time for the classic dynamic-programming solution is $O(n^3)$, which is polynomial.

On the other hand, the RNA folding problem using the third objective function (only Q variables in the objective function) is **NP-hard** (proven in a 2004 paper by Rune Lyngso, LNCS 3142:919-931 July 2004). So, one might expect that the ILP using the first objective function would (empirically) run faster than the ILP with the third objective.

But that expectation is wrong - Nature Bats Last!.

More Elaborate Models of RNA Folding

Complementary pairing, *nested pairing* and *base stacking*, along with appropriate *weights* for matched pairs and stacked quartets are the most critical features of RNA folding models.

However, many other features of RNA folds have been incorporated into fold models. About *two hundred* parameter choices in one RNA folding package.

E.g., In some models of RNA folding, the weight given to stacked quartet depends both on the **specific characters** in the quartet, and on **where** it appears 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.

Location of Stacked Quartets

We extend the ILP formulation to recognize **where** a stacked quartet appears in a stack. Let $F(i,j)$ be an ILP variable that will be set to 1 if and only if the stacked quartet $(i, i+1, j-1, j)$ is the **first** stacked quartet in a stack. We use:

$$Q(i,j) - Q(i-1,j+1) - F(i,j) \leq 0$$

and

$$2F(i,j) - Q(i,j) + Q(i-1,j+1) \leq 1.$$

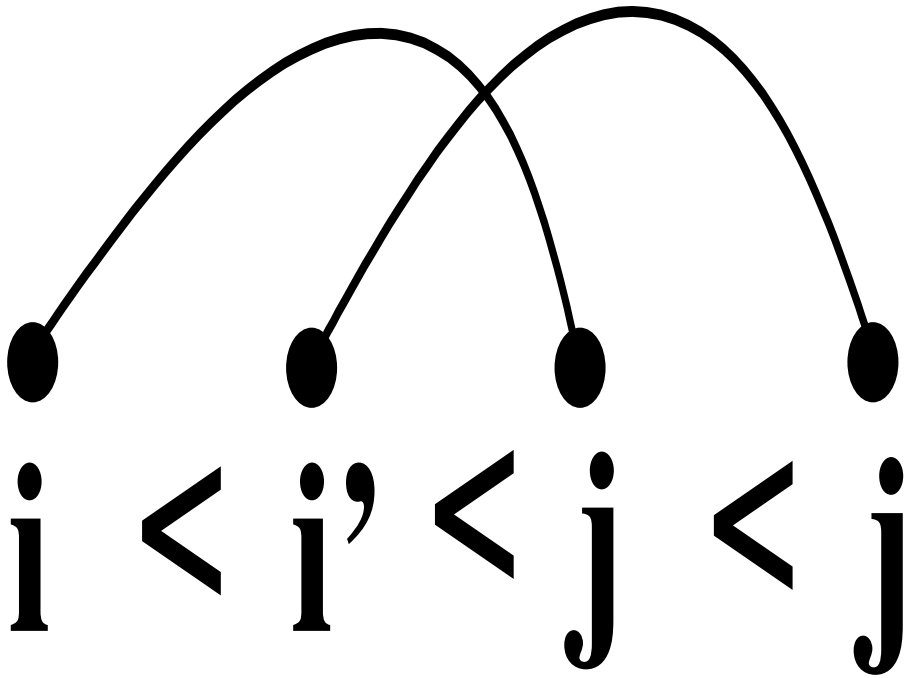
We can use the F variables to **count** the number of stacks in a pairing, or give a special weight to a matched pair that begins a stack, depending on what characters are in the matched pair (this is commonly done). We can also use the F variables and counts to encourage the matched pairs to form a **few, long** stacks. We leave these extensions to the readers.

Further Extension: Allowing Certain Crossing Pairs

Up until now, we have required that pairings be *nested, non-crossing*.

However, a limited number of **crossing** matched pairs are sometimes observed in RNA folds, particularly for RNA molecules that are not tRNA molecules.

Allowing Some Crossing Pairs



Crossing matched pairs

Allowing Crossing Pairs

Let $C(i, i', j, j')$ be an ILP variable with value 1 **if** (i, j) and (i', j') are matched pairs **that cross**.

Then, for every choice of four positions $i < i' < j < j'$, create the inequality:

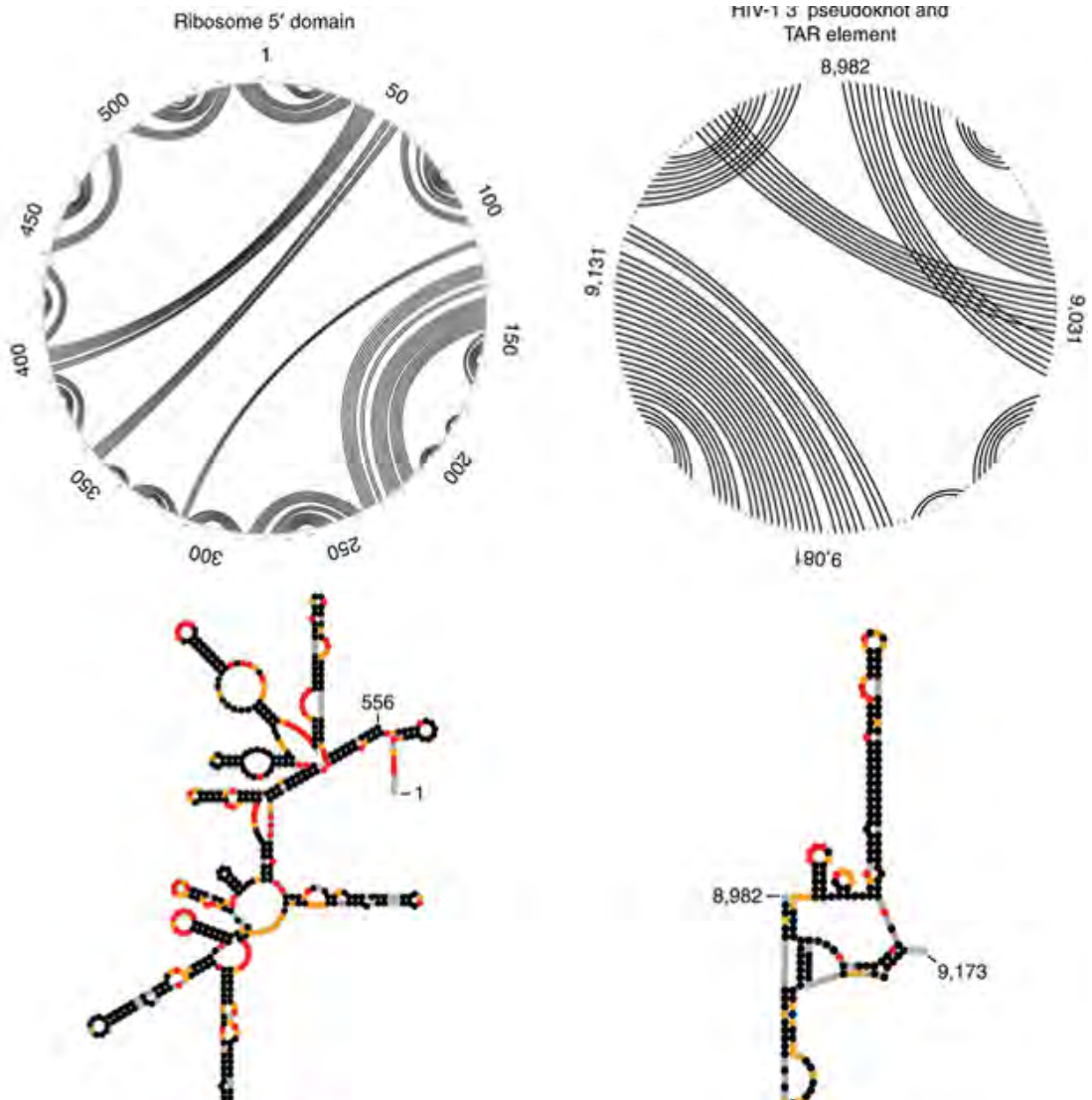
$$P(i, j) + P(i', j') - C(i, i', j, j') \leq 1 \tag{5}$$

We don't need inequalities for the *only if* direction.

With C variables, we can add an inequality that limits the number of crossing matched pairs to a fixed number, or to a fixed percentage of the number of crossing matched pairs in the pairing, etc.

Pseudo-Knots

Pseudo-knots generalize crossing pairs. If two sets of crossing matched pairs are organized into two **stacks**, they form a pseudo-knot, which is a **single** structural feature.



Modeling Pseudo-Knots in an ILP

Pseudo Knots are hard to incorporate into dynamic programming approaches to RNA folding, but **easy to model** in ILP.

We create the binary ILP variable, $PS(i, i')$, for $i < i'$, and set it to 1 if and only if there is a pseudo-knot whose two stacks **begin** at positions i and i' respectively.

Then create inequalities to set $PS(i, i')$ to 1 if and only if there are positions j and j' where

$$F(i, j) = 1 \text{ and } F(i', j') = 1 \text{ and } C(i, i', j, j') = 1.$$

That is, the matched pairs i, j and $i'j'$ are the *heads* of two stacks, and the matched pairs i, j and i', j' cross.

How to Use the $PS(i, i')$ variables

For each pseudo-knot there is exactly one $PS(i, i')$ variable set to 1, and this allows us to count the number of pseudo-knots in a pairing. Hence, we can trivially bound the number of allowed pseudo-knots, or the percentage of stacks that involved in a pseudo-knot, etc.

We can also use the Q and F variables to make the ILP formulation **count** the number characters contained in all of the stacks. Similarly, we can count the number of characters in stacks involved in a pseudo-knot, and use these count in inequalities to limit the number, or percentage, of such characters. We leave this again as an exercise.

Ending Comments

- RNA folding is just one illustration of ILP in computational biology, but it is a good representative.
- How ILPs for Computational Biology differ from ILPs for traditional applications: The native description of the problems rarely look linear, in contrast to traditional applications (for example, the classic diet problem). An ILP formulation in computational biology encodes what are called **reductions** in computer science.
- Most problems solved by ILP in computational biology are NP-hard. The virtue of NP-hardness: **Compact Expressibility!**

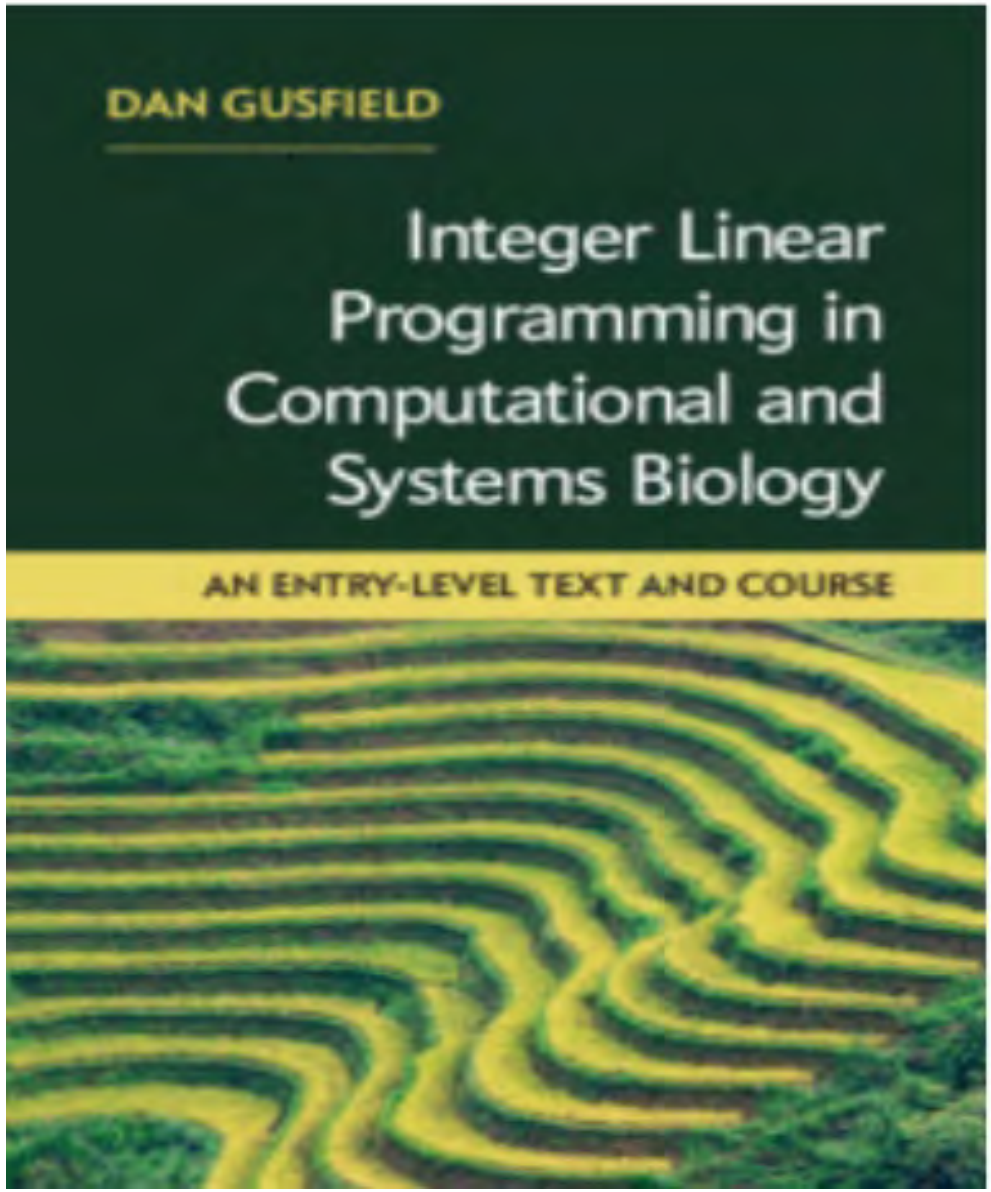
Why ILP?

If expressibility is so good, why use ILP and not some other NP-hard problem?

In *theory*, any NP-hard problem will do. But *In practice*, there is an industry which invests, maintains, and supports ILP. There is no similar industry for any other NP-hard problem.

SAT-solvers can be competitive (or even *superior*) for some pure decision problems, but they don't naturally solve **optimization** problems, especially with an extended range of values for feasible solutions.

Thank You


DAN GUSFIELD
Integer Linear Programming in Computational and Systems Biology
AN ENTRY-LEVEL TEXT AND COURSE

Thank You – Questions?

GUROBI
OPTIMIZATION

The World's Fastest Solver

Your Next Steps

- **Try Gurobi 9.0 Now!**
  - Get a 30-day commercial trial license of Gurobi at www.gurobi.com/free-trial
  - Academic and research licenses are free.
- **For questions about Gurobi pricing, please contact sales@gurobi.com or sales@gurobi.de**
- **A recording of this webinar, including the slides, will be available in roughly one week.**