Introduction

- Biology is an incredible source of challenging problems for computer science
- Problems are often hidden or vaguely defined and emerge only after several cycles of feedback with biologists, physicists, chemists, etc
Introduction

- Biology is an incredible source of challenging problems for computer science
- Problems are often hidden or vaguely defined and emerge only after several cycles of feedback with biologists, physicists, chemists, etc

- Solving one of these problems can be of unpredictable importance for life sciences and medicine
Bioinformatics deals with modeling and solving problems, analyzing and filtering data, from biology and related life sciences.

- Data availability is huge.
- Data is affected by experimental errors.
- Computer science tools should help in analyzing and filtering.
Bioinformatics applications are divided in three categories:

1) **Support infrastructure for analysis and experiments**
   Applications of computational methods for automated environments for workflow management, description and annotation of experiments, minimal reporting requirements, ...

2) **Polynomial time solvable problems**
   The input size is large: e.g. string matching problems over DNA sequences.

3) **Intractable problems**
   NP-complete or worse problems. Mainly covered by this lecture.
Areas of Bioinformatics

1. **Genomics.** Study of the genomes. Huge amount of data, fast algorithms (not always), limited to sequence analysis.

   \[
   \ldots \text{G} \text{A} \text{T} \text{C} \text{T} \text{G} \text{T} \text{A} \text{C} \text{T} \text{G} \text{A} \text{G} \text{T} \ldots
   \]

2. **Structural Bioinformatics.** Study of the folding process of bio-molecules. Less structural data than sequence data available.

3. **Systems Biology.** Study of complex interactions in biological systems. High level of representation.
Why Constraint Programming?

- Models are rarely **stable** and **static**. Constraint Programming provides the level of elaboration-tolerance to support model modifications and incremental addition of new knowledge.
- Linear Programming is not enough (in particular for modeling energy models)
- Declarative formalism is elegant and concise!
- Model execution can be later speed-up with usual CP techniques (symmetry breaking, search heuristics, constraint based local search, parallelism, developing ad-hoc global constraints, etc)
What we’ll see in more details

We’ll survey the various areas by introducing some challenging problems and showing their (high level) constraint model just to give a taste of the feasibility of the CP approach.

- **Genomics:**
  - Haplotype Inference
  - Phylogenetic trees

- **Structural Bioinformatics:**
  - RNA secondary structure prediction
  - Protein structure prediction (on lattice)

- **Systems Biology:**
  - Reasoning on Biological Networks
Some introductory references

- Nice introductory slides by Sebastian Will
  math.mit.edu/classes/18.417/Slides/intro.pdf
- A movie on DNA replication
  www.youtube.com/watch?v=bee6PWUgPo8
- A movie on DNA transcription
  www.youtube.com/watch?v=5MfSYnItYvg
- A movie on Central Dogma
  www.youtube.com/watch?v=9kOGOY7vthk
- A movie on Systems Biology
  www.youtube.com/watch?v=1mB0xoRP914
Some references focused on Constraints and Bioinformatics

- 11 (+2) Workshops on Constraint-based methods for Bioinformatics: WCB05 (Sitges)–WCB15 (Cork)
  http://clp.dimi.uniud.it/wcb/
  (workshops also in CP’97 and CP’98)
- *Algorithms for Molecular Biology (Thematic Series of AMB on Constraints and Bioinformatics), since 2012.
Haplotype inference
DNA and Genome in a nutshell

DNA (DeoxyriboNucleic Acid) is characterized by a string of nucleotides: A, C, G, and T (Adenine, Cytosine, Guanine, Thymine)

Given a sequence $s \in \{A, C, G, T\}^*$ the complementary sequence $\bar{s}$ is deterministically obtained by reversing $s$ and substituting $A \leftrightarrow T$ and $C \leftrightarrow G$

$s$ and $\bar{s}$ fold together forming the famous double helix
DNA and Genome in a nutshell

- DNA strings are long ($10^6$–$10^{10}$ nucleotides).
- Differences between the DNAs of two members of the same specie are limited (e.g., 1 in 1000 for humans).
- Some fragments of the DNA, called Genes, encode proteins (we’ll be back on that later).
- After the Human Genome Project, it is estimated that there are 16–20K protein-coding genes in human DNA.
- Differences of some nucleotides in the same gene characterize a property of an individual w.r.t. another.
- The set of all genes of an individual is called Genome.
Haplotype Inference

- Genes are packaged in bundles called chromosomes. (Chromosomes are therefore regions of DNA)

- In diploid organisms (like humans) there are almost identical chromosome pairs. Each pair is made of an inherited chromosome from the father and another from the mother.

- A haplotype is a DNA sequence that has been inherited from one parent.

- A genotype is a pairing of two corresponding haplotypes.
Haplotyping Inference

Each person inherits two haplotypes (from the mother and from the father) for most regions of the genome.

\[ \cdots \ G \ A \ T \ C \ T \ G \ T \ A \ C \ T \ G \ A \ G \ T \ \cdots \]
\[ \cdots \ G \ A \ T \ C \ T \ G \ T \ A \ C \ T \ G \ A \ A \ A \ A \ T \ \cdots \]
Haplotype Inference

Each person inherits two haplotypes (from the mother and from the father) for most regions of the genome.

```
... G A T C T G T A C T G A G T ...
... G A T C T G T A C T G A A T ...
```

⇑ ⇑⇑ ∗⇑∗

In some typical positions, the bases are subject to mutations.
Haplotypic Inference

Each person inherits two haplotypes (from the mother and from the father) for most regions of the genome.

\[
\begin{array}{cccccccccccc}
\cdots & G & A & T & C & T & G & T & A & C & T & G & A & G & T & \cdots \\
\cdots & G & A & T & C & T & G & T & A & C & T & G & A & A & T & \cdots \\
\uparrow & \uparrow & \uparrow & \uparrow & * & \uparrow & * &
\end{array}
\]

In some typical positions, the bases are subject to mutations. In the most common case, there is a Single Nucleotide Polymorphism (SNP).
Haplotype Inference

Each person inherits two haplotypes (from the mother and from the father) for most regions of the genome.

\[ \cdots \ G \ A \ T \ C \ T \ G \ T \ A \ C \ T \ G \ A \ G \ T \ \cdots \]

\[ \cdots \ G \ A \ T \ C \ T \ G \ T \ A \ C \ T \ G \ A \ A \ T \ \cdots \]

\[ \uparrow \ \uparrow \ \uparrow \ \uparrow \ \ast \ \uparrow \ \ast \]

In some typical positions, the bases are subject to mutations. In the most common case, there is a Single Nucleotide Polymorphism (SNP).

Mutations are \( C \leftrightarrow T \) and \( A \leftrightarrow G \)
Haplotype Inference

Single Nucleotide Polymorphism (SNP)

Each person has two haplotypes (from the mother and from the father) for most regions of the genome:

<table>
<thead>
<tr>
<th>G</th>
<th>A</th>
<th>A</th>
<th>T</th>
<th>C</th>
<th>T</th>
<th>T</th>
<th>T</th>
<th>C</th>
<th>G</th>
<th>T</th>
<th>A</th>
<th>C</th>
<th>T</th>
<th>G</th>
<th>A</th>
<th>G</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>G</td>
<td>A</td>
<td>A</td>
<td>T</td>
<td>C</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>C</td>
<td>G</td>
<td>T</td>
<td>A</td>
<td>C</td>
<td>T</td>
<td>G</td>
<td>A</td>
<td>A</td>
<td>T</td>
</tr>
</tbody>
</table>

Let us focus on the SNPs:

A   C   T   G
A   C   T   A

We encode SNPs according to:  

A  \leftrightarrow  0  
C  \leftrightarrow  0  
G  \leftrightarrow  1  
T  \leftrightarrow  1
Haplotype Inference

Single Nucleotide Polymorphism (SNP)

Each person has two haplotypes (from the mother and from the father) for most regions of the genome:

| G | A | A | T | C | T | T | C | G | T | A | C | T | G | A | G | T |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| G | A | A | T | C | T | T | C | G | T | A | C | T | G | A | A | T |

Let us focus on the SNPs:

- A  C  T  G
- A  C  T  A

We encode SNPs according to:  
\[ A \rightarrow 0 \quad C \rightarrow 0 \quad G \rightarrow 1 \quad T \rightarrow 1 \]

| 0 0 1 1 | 0 0 1 0 |
Haplotype Inference
Single Nucleotide Polymorphism (SNP)

Each person has two haplotypes (from the mother and from the father) for most regions of the genome:

<table>
<thead>
<tr>
<th>G</th>
<th>A</th>
<th>A</th>
<th>T</th>
<th>C</th>
<th>T</th>
<th>T</th>
<th>T</th>
<th>C</th>
<th>G</th>
<th>T</th>
<th>A</th>
<th>C</th>
<th>T</th>
<th>G</th>
<th>A</th>
<th>G</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>G</td>
<td>A</td>
<td>A</td>
<td>T</td>
<td>C</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>C</td>
<td>G</td>
<td>T</td>
<td>A</td>
<td>C</td>
<td>T</td>
<td>G</td>
<td>A</td>
<td>A</td>
<td>T</td>
</tr>
</tbody>
</table>

Let us focus on the SNPs:

<table>
<thead>
<tr>
<th>A</th>
<th>C</th>
<th>T</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>C</td>
<td>T</td>
<td>A</td>
</tr>
</tbody>
</table>

We encode SNPs according to: $A \mapsto 0$, $C \mapsto 0$, $G \mapsto 1$, $T \mapsto 1$

<table>
<thead>
<tr>
<th>0</th>
<th>0</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

But this is the situation of complete knowledge. In practice, we can detect a mismatch but not its single components.

| 0 | 0 | 1 | 2 |

The genotype is set to 2 if there is a mismatch.
Haplotype Inference
Looking for an explanation

2 1 2

1 2 1
Haplotype Inference

Looking for an explanation
Haplotype Inference
Looking for an explanation
Haplotype Inference

- A string of \( \{0, 1\}^* \) is called a *haplotype*
- A string of \( \{0, 1, 2\}^* \) is called a *genotype*
- Two equal length haplotypes generate a unique genotype
- The rules are \( 0 \oplus 0 = 0, \ 1 \oplus 1 = 1, \ 0 \oplus 1 = 2 \)
  
  E.g., \( 0010, 0101 \Rightarrow 0222 \)
- If we have a genotype, we can only conjecture (potentially exponentially many) *haplotypes* that generated it
  
  (observe that, e.g., \( 0110, 0001 \Rightarrow 0222 \))
- Biological experiments allow us to know genotypes!
- Investigating *sets* of genotypes for a population, helps in understanding the relationships between SNPs and physical features as well as medical information
- Since genotypes are introduced in evolution, it is reasonable to find minimal sets of haplotypes explaining the known genotypes.
Haplotype Inference

- Let $H$ be the set of *haplotypes* (of given length $n$) and $G$ be a set of *genotypes* (of the same length $n$).
- Given $h_1, h_2 \in H$ and $g \in G$, $\{h_1, h_2\}$ explains $g$ if and only if $|h_1| = |h_2| = |g|$ and $\forall i \in [1..n]:$
  
  \[
  g[i] \leq 1 \quad \rightarrow \quad h_1[i] = h_2[i] = g[i]
  \]
  
  \[
  g[i] = 2 \quad \rightarrow \quad h_1[i] \neq h_2[i]
  \]

- A set of haplotypes $H$ explains a set of genotypes $G$ if for all $g \in G$ there are $h_1, h_2 \in H$ such that $\{h_1, h_2\}$ explains $g$.
- Given a set of genotypes $G$ and an integer $k$, the *haplotype inference problem (HIP)* by pure parsimony is the problem of finding a set $H$ that explains $G$ and such that $|H| = k$ (decision version—NP complete).
Haplotype Inference

CP encoding

- Let us focus on the decisional version: Is there an explanation for $G$ with $k$ haplotypes?
- Generate $m = 2|G|$ vectors of 0-1 FD variables $H_1, \ldots, H_m$ of length $n$
- Add a $<$-lexicographical constraint on each pair $(H_1, H_2), (H_3, H_4), \ldots, (H_{m-1}, H_m)$ (repetitions in different pairs are allowed!)
- Build a constraint of the form:

  $$(\forall G_i \in G) \left( \langle H_{2i-1}, H_{2i} \rangle \text{ explain } G \right)$$

- Namely:

  $$\bigwedge_{j=1}^{n} \left( G_i[j] \leq 1 \rightarrow (H_{2i_1}[j] = H_{i_2}[j] = G_{2i}[j]) \wedge 
  G_i[j] = 2 \rightarrow (H_{2i_1}[j] \neq H_{2i}[j]) \right)$$

- We need to state (using constraints!) that $|\{H_1, \ldots, H_m\}| = k$. 
For $a, b \in [1..m]$ we set $F_{a,b} \leftrightarrow \bigwedge_{i=1}^{n} H_a[i] = H_b[i]$.

Namely $F_{a,b}$ is a Boolean variable that is true iff $H_a$ and $H_b$ will be equal in the solution.
Haplotype Inference

2nd CP encoding

- For $a, b \in [1..m]$ we set $F_{a,b} \iff \bigwedge_{i=1}^{n} H_a[i] = H_b[i]$.
- Namely $F_{a,b}$ is a Boolean variable that is true iff $H_a$ and $H_b$ will be equal in the solution.
- Then define $M_a \iff \bigvee_{b=a+1}^{m} F_{a,b}$
- $M_a$ is again a Boolean variable that is true if and only if there is another vector in $H_{a+1}, H_{a+2}, \ldots, H_m$ equal to $H_a$. 

Agostino Dovier (Univ. of Udine, DIMI)
**Haplotype Inference**

2nd CP encoding

For \( a, b \in [1..m] \) we set \( F_{a,b} \leftrightarrow \bigwedge_{i=1}^{n} H_a[i] = H_b[i] \).

Namely \( F_{a,b} \) is a Boolean variable that is true iff \( H_a \) and \( H_b \) will be equal in the solution.

Then define \( M_a \leftrightarrow \bigvee_{b=a+1}^{m} F_{a,b} \).

\( M_a \) is again a Boolean variable that is true if and only if there is another vector in \( H_{a+1}, H_{a+2}, \ldots, H_m \) equal to \( H_a \).

The size of \( H \) can be therefore expressed as \( \sum_{a=1}^{n} (1 - M_a) \) (viewing Boolean truth values as 0/1).
Haplotype Inference

Some References

- Gusfield and Orzack. Haplotype Inference (Survey, and ILP formulations) In CRC Handbook on Bioinformatics, 2006
- Graça, Marques-Silva, Lynce, Oliveira. Several works on SAT-based and specialized 0-1 ILP for Haplotype Inference. (e.g. WCB 08, WCB 09)
- James Cussens Maximum likelihood pedigree reconstruction using integer programming. WCB 10.
Phylogenetics
**Phylogenetic trees**

**Basics**

- A *phylogeny* describes evolutionary relationships among entities.
- Comparative biology: investigates similarities and differences
- More reliable than pattern matching
- Applied outside biology: e.g. Indo-European languages [Erdem03]
The entities a set $L$ of elementary taxonomic units, known as taxa (e.g., $L = \{\text{English}, \text{German}, \text{French}, \text{Spanish}, \text{Italian}\}$ or $L = \{\text{dog}, \text{cat}, \text{horse}, \text{chicken}\}$). A set $C$ of characters is assigned to each element of $L$ (e.g., characters “hand” and “father”, or characters “number of legs”, “length of the tail”, etc.). Characters are evaluated with FD values (e.g. $\{1$ (hand), 2 (mano/main)\} for “hand” and $\{1$ (father/padre), 2 (vater/père)\} for “father”) Each element in $L$ is assigned a value for each character. Let us focus on Boolean characters
Phylogenetic tree reconstruction

- A phylogeny $(V, E, L, C, D, f)$ for a set $L$ of taxa is a
  - finite binary tree $(V, E)$ with leaves $L \subseteq V$ (taxa=leaves, with a slight abuse of notation)
  - along with two finite sets $C$ and $D$ and a function $f : L \times C \rightarrow D$.

- $V \setminus L$ describes the ancestral units and $E$ evolutionary relationships.
- $C$ is the set of characters, and $D$ contains their domain values (also known as states).
- $f$ labels every leaf $v \in L$ by assigning a state for each character $i \in C$. 
A phylogeny \((V, E, L, C, D, f)\) where
\[ L = \{\text{English, German, French, Spanish, Italian}\} \] (taxa)
\[ C = \{\text{Hand, Father}\} \] (characters),
\[ D = \{1, 2\} \] (states).
A character $i \in C$ is **compatible** with a phylogeny if the taxa that present the same value for $i$ are connected by a subtree.

Character "Hand" is compatible with the above tree.
A character \( i \in C \) is compatible with a phylogeny if the taxa that present the same value for \( i \) are connected by a subtree.

Character Hand is compatible with the above tree.
A character $i \in C$ is **compatible** with a phylogeny if the taxa that present the same value for $i$ are connected by a subtree.

Character "Hand" is compatible with the above tree.
A character $i \in C$ is compatible with a phylogeny if the taxa that present the same value for $i$ are connected by a subtree. Otherwise it is incompatible.

Character Father is incompatible with the above tree.
A character $i \in C$ is compatible with a phylogeny if the taxa that present the same value for $i$ are connected by a subtree.

Otherwise it is incompatible.

Character Father is incompatible with the above tree.
A character $i \in C$ is compatible with a phylogeny if the taxa that present the same value for $i$ are connected by a subtree.

Otherwise it is incompatible.

Character "Father" is incompatible with the above tree.
Phylogenetic trees

$k$-incompatibility

The above subtree requirement implicitly states that when a character changes (in the evolution) it never go back to the previous value (Camin-Sokal). Moreover, that the change occurs in a unique place (Dollo).
Phylogenetic trees

$k$-incompatibility

- The above subtree requirement implicitly states that when a character changes (in the evolution) it never go back to the previous value (Camin-Sokal). Moreover, that the change occurs in a unique place (Dollo).

$k$-INCOMPATIBILITY PROBLEM

Given sets $L$ (taxa/leaves), $C$ (characters), and $D$ (states), a function $f : L \times C \rightarrow D$, and $k \in \mathbb{N}$, decide the existence of a phylogeny $(V, E, L, C, D, f)$ with at most $k$ incompatible characters.
Phylogenetic trees

\( k \)-incompatibility

- The above subtree requirement implicitly states that when a character changes (in the evolution) it never go back to the previous value (Camin-Sokal). Moreover, that the change occurs in a unique place (Dollo).

**k-INCOMPATIBILITY PROBLEM**

Given sets \( L \) (taxa/leaves), \( C \) (characters), and \( D \) (states), a function \( f : L \times C \rightarrow D \), and \( k \in \mathbb{N} \), decide the existence of a phylogeny \((V, E, L, C, D, f)\) with at most \( k \) incompatible characters.

- This problem is NP-complete (Day, Sankoff 1986).
- The number of possible phylogenies is exponential in \( L \).
- NP-complete (Day, Sankoff 1986).
Input vector $L$ of $n$ elements (taxa) each of them characterized by a $m$-tuple of (character) values.

For simplicity, let us focus on Boolean encodings.

E.g. $m = 3, n = 4$:

$$L = [[0, 1, 1], [1, 0, 0], [1, 1, 0], [1, 0, 1]]$$

(four elements/taxa with three characters)
**Encoding: Binary tree**

- The Tree can be represented by a FD vector of \( t = 2n - 1 \) elements valued in \((n+1), \ldots, t + 1\).

- Tree\([i] = j\) means that node \(i\) is a son of node \(j\). For the root \(r\), Tree\([r] = t + 1\).

- The tree is binary: for \(i = 1, \ldots, n\):
  
  \[
  \text{count}(i, \text{Tree}, \leq, 2)
  \]

**Symmetries:**

- Taxa are the leaves of the tree: nodes 1 \ldots n
- Tree\([1] = n + 1\)  
- Tree\([t] = t + 1\) (\(t\) is the root)
- For \(i, j \in \{1, \ldots, t\}\): \(i < j \rightarrow \text{Tree}[i] \leq \text{Tree}[j]\)
Each node of the tree is assigned a $m$-tuple of Boolean Values. This is stored in a vector Chars.

Chars[1]–Chars[$n$] are assigned using the input $L$. Values for internal nodes must be computed.

For $i < j$, if $\text{Tree}[i] = j$, the Hamming difference of the corresponding tuples is 1. Precisely:

$$\text{Tree}[i] = j \rightarrow \left( \sum_{\ell=1}^{m} |\text{Chars}[i][\ell] - \text{Chars}[j][\ell]| \right) = 1$$
Encoding

Hypercube tree

- Each node of the tree is assigned a \( m \)-tuple of Boolean Values. This is stored in a vector Chars.
- Chars\([1]\)–Chars\([n]\) are assigned using the input \( L \). Values for internal nodes must be computed.
- For \( i < j \), if Tree\([i]\) = \( j \), the Hamming difference of the corresponding tuples is 1. Precisely:
  \[
  \text{Tree}[i] = j \rightarrow \left( \sum_{\ell=1}^{m} |\text{Chars}[i][\ell] - \text{Chars}[j][\ell]| \right) = 1
  \]
- Actually, we can either relax the above constraint to \( \leq 1 \) (see e.g. hand/father example, italian and spanish) or (alternatively)
- Add the redundant constraint

\[\text{AllDifferentTuples(Chars)}\]
Encoding

$k$-incompatibility

- We need to state that a character changes (actually, increases) in at most one node. This makes the tree compatible with that character.
- Let Comp be a vector of $m$ elements (one per character).
- For $i < j$, let $F_{i,j} = 1$ if $\text{Tree}[i] = j$, $F_{i,j} = 0$ otherwise.
- Then, for $\ell = 1, \ldots, m$ (and $i, j = 1, \ldots, n$):

$$\text{Comp}[\ell] = \sum_{i < j} F_{i,j}(\text{Chars}[i][\ell] - \text{Chars}[j][\ell])$$

- Basically, after variable instantiation, Comp$[\ell]$ will contain the number of changes of character $\ell$ in the tree.
- The number of values different from 1 and 0 in Comp is forced to be less than or equal to $k$. 
(Some) References


\[(x > y = z) \lor (y > x = z) \lor (z > x = y) \lor (x = y = z)\]

- Le Tiep, Nguyen Hieu, Pontelli Enrico, and Cao Son Tran. ASP at Work: An ASP Implementation of PhyloWS. ICLP 2012, LIPICS vol 17. (also in WCB 12)
RNA is a sequence of nucleotides (A,C,G,U) that (often) is just an intermediary between DNA and proteins.

DNA strands are transcribed to mRNA, in order to exit the cell’s nucleus.

Nucleotides replacement: DNA T \rightarrow RNA U.
RNA Secondary Structure

RNA folds according to favorable matchings (A-U, C-G, ~ U-G)
- The **secondary structure** is the set of its base pairings
- Secondary structure determines the 3D properties
RNA Secondary Structure

- RNA folds according to favorable matchings (A-U, C-G, ~ U-G)
- The secondary structure is the set of its base pairings
- Secondary structure determines the 3D properties
Mathematically

- A RNA sequence $\tilde{s} = s_1 s_2 \cdots s_n$ is a string in $\{A, C, G, U\}^*$
- A RNA secondary structure is a (partial) injective function $P \subseteq \{1, \ldots, n\}^2$ such that
  - $(i, j) \in P \iff (j, i) \in P$
  - $(i, j) \in P$ only if $(s_i, s_j) \in \{(A, U), (U, A), (C, G), (G, C), (U, G), (G, U)\}$
Mathematically

- A RNA sequence $\tilde{s} = s_1 s_2 \cdots s_n$ is a string in $\{A, C, G, U\}^*$
- A RNA secondary structure is a (partial) injective function $P \subseteq \{1, \ldots, n\}^2$ such that
  - $(i, j) \in P \leftrightarrow (j, i) \in P$
  - $(i, j) \in P$ only if $(s_i, s_j) \in \{(A, U), (U, A), (C, G), (G, C), (U, G), (G, U)\}$
- We are interested in a solution with maximal pairings (and/or minimizing a more complex energy function)
The general problem is NP-complete [Lyngsø and Pedersen 2000].
A large sub-class has *polynomial time* complexity:
the absence of *pseudo-knots*, e.g. (8,10).
To avoid pseudo-knots, we impose a constraint:
If $i < \ell < j$ and $(i, j) \in P$, and $((\ell, k) \in P$ or $(k, \ell) \in P$), then $i < k < j$. 
A simple CP encoding

- Input $s_1, \ldots, s_n \in \{A, C, G, U\}$
- Variables $Pairs = [P_1, \ldots, P_n]$ with domain 0..n.
- Let $S_x = \{i \in \{1, \ldots, n\} \mid s_i = x\}$.
  - If $s_i = A$, then $\text{dom}(P_i) = \{0\} \cup S_U$.
  - If $s_i = C$, then $\text{dom}(P_i) = \{0\} \cup S_G$.
  - If $s_i = G$, then $\text{dom}(P_i) = \{0\} \cup S_C \cup S_U$.
  - If $s_i = U$, then $\text{dom}(P_i) = \{0\} \cup S_A \cup S_G$.
- For $i = 1, \ldots, n$, if $P_i > 0$ then $P_{P_i} = I$. If $P_i = 0$ no constraint. In CLP(FD) we can state:
  \[
  \text{element}(P + 1, [I|Pairs], I)
  \]

- Pseudo-knots: If $P_i > 0$ then $(P_{i+1} \in [i + 3..P_i - 1]) \lor (P_{i+1} = 0)$
A simple CP encoding

- As cost function we want either to maximize contacts or (as done by Dahl-Bavarian, WCB 05),
- a solution close to the statistics, namely 35% for AU, 53% for CG, 12% for GU.
- Let $NC = n - \#\text{contacts}$
- We minimize therefore a weighted sum of the form

$$
\frac{NC}{n} c_1 + \frac{\#(AU) - .35(n - NC)}{n} c_2 + \frac{\#(CG) - .53(n - NC)}{n} c_3
$$

($c_1$, $c_2$, $c_3$ constants that can be changed. The denominator $n$ can be omitted for minimization)

- Other functions can be implemented, of course.
(Some) References


Protein Structure Prediction
Proteins and Central Dogma

- The translation phase starts from a mRNA sequence and associates a protein sequence.
- Proteins are made of amino acids (20 common different types).
- Amino acids are defined by letters \( \{A, \ldots, Z\} \setminus \{B, J, O, U, X, Z\} \)
The translation selects 3 RNA basis and associates 1 amino acid.
The translation rules are encoded in the universal code.
The code contains stop symbol and some redundant RNA triplets.
Proteins

Amino acids

- Proteins are molecules made of a linear sequence of amino acids.
- Amino acids are combined through *peptide bond*.
Proteins

Amino acids

- Proteins are molecules made of a linear sequence of amino acids.
- Amino acids are combined through peptide bond.

The purple dots represent the side chains, that depend on the amino acid type.
- Side chains have different shape, size, charge, polarity, etc.
- A side chain contains from 1 (Glycine) up to 18 (Tryptophan) atoms.
There are 2 degrees of freedom (black arrows) for each amino acid.

A protein with $n$ amino acids has $2n$ degrees of freedom (plus side chains)!

Typical size range from 50 to 500 amino acids.
The structure prediction problem

- Given the **primary structure** of a protein (its amino acid sequence)
- For each amino acid, output its position in the space (**tertiary structure** of a protein)
The structure prediction problem

- Given the **primary structure** of a protein (its amino acid sequence)
- For each amino acid, output its position in the space (**tertiary structure** of a protein)

```
ALFWKLRRL...
```

- **Secondary structures** are rigid subparts (helices, sheets) that can be “easily” predicted
Proteins

Facts

- Folding is **consistent** ⇒ same protein folds in the same way [Anfinsen74]
- Folding is **fast** ⇒ 1ms – 1s
- Driven by **non covalent** forces: electrostatic interactions, volume constraints, Hydrogen Bonding, van der Waals, Salt/disulfide Bridges
- Backbone is rigid, interaction with water, ions and ligands
- There is a fixed distance (3.8Å) between the $C\alpha$ atoms of consecutive aminoacids.
- There are several statistics on (bend/torsional) angles.
The structure prediction problem

... and this is the hard part:

- In nature a protein has a unique/stable 3D conformation
- A cost function (that mimics physics laws) can be used to score each conformation
- Searching for the optimal score produces the best candidate is difficult (NP-complete even in extremely simplified modelings)
The protein structure prediction problem

- A first simplification (HP):
  - **Protein model**: only one atom per amino acid, only 2 classes of amino acids (hydrophobic and polar)
The protein structure prediction problem

- A first simplification (HP):
  - **Protein model**: only one atom per amino acid, only 2 classes of amino acids (hydrophobic and polar)

- A second simplification:
  - **Spatial model**: 2D square lattice to represent amino acid positions
The protein structure prediction problem

Model

- The input is a list $S$ of amino acids $S = s_1, \ldots, s_n$,
- where $s_i \in \{h, p\}$
- Each $s_i$ is placed on a 2D grid with integer coordinates
- Any pair of two amino acids can’t occupy the same position
- If two amino acids are at distance 1, they are in contact
The protein structure prediction problem

Model

- A folding is a function $\omega : \{1, \ldots, n\} \rightarrow \mathbb{N}^2$ where
  - $\forall i$ next($\omega(i), \omega(i + 1)$) and
  - $\forall i, j (i \neq j \rightarrow \omega(i) \neq \omega(j))$
  - next($\langle X_1, Y_1 \rangle, \langle X_2, Y_2 \rangle$) $\iff |X_1 - X_2| + |Y_1 - Y_2| = 1$. 

\[ \text{Find a folding that minimizes the (simplified) energy function:} \]
\[ E(S, \omega) = \sum_{1 \leq i \leq n-2} \sum_{2 \leq j \leq n} \text{Pot}(s_i, s_j) \cdot \text{next}(\omega(i), \omega(j)) \]
\[ \text{where Pot}(p, p') = \text{Pot}(h, p) = \text{Pot}(p, h) = 0 \text{ and Pot}(h, h) = -1. \]
The protein structure prediction problem

Model

- A folding is a function \( \omega : \{1, \ldots, n\} \rightarrow \mathbb{N}^2 \) where
- \( \forall i \) next(\( \omega(i), \omega(i + 1) \)) and
- \( \forall i, j \) (\( i \neq j \rightarrow \omega(i) \neq \omega(j) \))
- next(\( \langle X_1, Y_1 \rangle, \langle X_2, Y_2 \rangle \)) \iff |X_1 - X_2| + |Y_1 - Y_2| = 1.
- Find a folding that minimizes the (simplified) energy function:

\[
E(S, \omega) = \sum_{1 \leq i \leq n-2} \sum_{i+2 \leq j \leq n} \text{Pot}(s_i, s_j) \cdot \text{next}(\omega(i), \omega(j))
\]

where \( \text{Pot}(p, p) = \text{Pot}(h, p) = \text{Pot}(p, h) = 0 \) and \( \text{Pot}(h, h) = -1 \).
The protein structure prediction problem

Complexity

- With $N^2$ and HP, establishing whether there is a folding with energy $< k$ is NP-complete

This formulation of the problem has a nice property: you can teach it to a chidren without speaking of proteins and so on:  
*Write a folding using paper and pencil that maximizes the contacts between “H” aminoacids (black circles)*
Example of PF HP $N^2$

Yellow: H, Grey: P. All foldings have energy -6
HP on $\mathbb{N}^2$: FD encoding

- Primary $= [a_1, \ldots, a_n] = [h/p, p/p, h/p, \ldots]$
- Tertiary $x = [X_1, \ldots, X_n]$, Tertiary $y = [Y_1, \ldots, Y_n]$
HP on $\mathbb{N}^2$: FD encoding

- Primary $= [a_1, \ldots, a_n] = [h/p, p/p, h/p, \ldots]$
- Tertiary $\_x = [X_1, \ldots, X_n]$, Tertiary $\_y = [Y_1, \ldots, Y_n]$
- W.l.o.g., let $X_1 = X_2 = Y_1 = n$, $Y_2 = n + 1$. 
HP on $\mathbb{N}^2$: FD encoding

- Primary $= [a_1, \ldots, a_n] = [h/p, p/p, h/p, \ldots]$
- Tertiary $x = [X_1, \ldots, X_n]$, Tertiary $y = [Y_1, \ldots, Y_n]$
- W.l.o.g., let $X_1 = X_2 = Y_1 = n$, $Y_2 = n + 1$.
- Namely, we start with

```
  + - - - - - - - -  +
 /                   /
 n + 1               n + 1
 /                   /
  + - - - - - - - -  +
 /                   /
 n                   n
 /                   /
  + - - - - - - - -  +
 /                   /
 n - 1               n - 1
 /                   /
  + - - - - - - - -  +
```

```
n - 1  n  n + 1
```
HP on $\mathbb{N}^2$: FD encoding

- Primary $= [a_1, \ldots, a_n] = [h/p, p/p, h/p, \ldots]$
- Tertiary $x = [X_1, \ldots, X_n]$, Tertiary $y = [Y_1, \ldots, Y_n]$
- W.l.o.g., let $X_1 = X_2 = Y_1 = n$, $Y_2 = n + 1$.
- Namely, we start with

$$\text{dom}(X_1) = \cdots = \text{dom}(X_n) = \text{dom}(Y_1) = \cdots = \text{dom}(Y_n) = 1..2n$$

![Diagram showing the relationship between primary and tertiary structures.](attachment:diagram.png)
HP on $\mathbb{N}^2$: FD encoding

- $\text{Tertiary}_x = [X_1, \ldots, X_n]$, $\text{Tertiary}_y = [Y_1, \ldots, Y_n]$
- contiguous: for $i = 1, \ldots, n - 1$: $|X_i - X_{i+1}| + |Y_i - Y_{i+1}| = 1$
- no-overlap: for $i = 1, \ldots, n - 1$, for $j = i + 1, \ldots, n$: $|X_i - X_j| + |Y_i - Y_j| \geq 1$
HP on $\mathbb{N}^2$: FD encoding

- Tertiary$_x = [X_1, \ldots, X_n]$, Tertiary$_y = [Y_1, \ldots, Y_n]$
- contiguous: for $i = 1, \ldots, n-1$: $|X_i - X_{i+1}| + |Y_i - Y_{i+1}| = 1$
- no-overlap: for $i = 1, \ldots, n-1$, for $j = i + 1, \ldots, n$: $|X_i - X_j| + |Y_i - Y_j| \geq 1$
- We want to express that $(X_i, Y_i) \neq (X_j, Y_j)$. Can we use alldifferent?
HP on $\mathbb{N}^2$: FD encoding

- Tertiary $x = [X_1, \ldots, X_n]$, Tertiary $y = [Y_1, \ldots, Y_n]$
- contiguous: for $i = 1, \ldots, n - 1$: $|X_i - X_{i+1}| + |Y_i - Y_{i+1}| = 1$
- no-overlap: for $i = 1, \ldots, n - 1$, for $j = i + 1, \ldots, n$: $|X_i - X_j| + |Y_i - Y_j| \geq 1$
- We want to express that $(X_i, Y_i) \neq (X_j, Y_j)$. Can we use alldifferent?
- Let $[P_1, \ldots, P_n]$ be a list and $M$ a “big” integer (100 is ok for us).
  - for $i = 1, \ldots, n - 1$: $P_i = X_i + MY_i$. 
HP on $\mathbb{N}^2$: FD encoding

- Tertiary $x = [X_1, \ldots, X_n]$, Tertiary $y = [Y_1, \ldots, Y_n]$
- contiguous: for $i = 1, \ldots, n - 1$: $|X_i - X_{i+1}| + |Y_i - Y_{i+1}| = 1$
- no-overlap: for $i = 1, \ldots, n - 1$, for $j = i + 1, \ldots, n$: $|X_i - X_j| + |Y_i - Y_j| \geq 1$

We want to express that $(X_i, Y_i) \neq (X_j, Y_j)$. Can we use alldifferent?

- Let $[P_1, \ldots, P_n]$ be a list and $M$ a “big” integer (100 is ok for us).
- for $i = 1, \ldots, n - 1$: $P_i = X_i + MY_i$.
- We can now post: alldifferent($[P_1, \ldots, P_n]$).
HP on $\mathbb{N}^2$: FD encoding

- Primary $= [a_1, \ldots, a_n] = [h, p, p, h, p, p, h, \ldots]$
- Tertiary $_x = [X_1, \ldots, X_n]$, Tertiary $_y = [Y_1, \ldots, Y_n]$
HP on $\mathbb{N}^2$: FD encoding

- Primary $= [a_1, \ldots, a_n] = [h, p, p, h, p, p, h, \ldots]$
- Tertiary $_x = [X_1, \ldots, X_n]$, Tertiary $_y = [Y_1, \ldots, Y_n]$
- energy: for $i = 1, \ldots, n - 2$, for $j = i + 2, \ldots, n$: $c_{i,j} \in \{0, -1\}$
  \[ c_{i,j} = -1 \iff (|X_i - X_i| + |Y_i - Y_j|) = 1) \land (a_i = a_j = h) \]
- Energy $= \sum_{i=1}^{n-2} \sum_{j=i+2}^{n} c_{i,j}$
3D Lattice models: Cube, FCC, Chess Knight
The FCC lattice

- The Face Centered Cube lattice models the discrete space in which the protein can fold.
- It is proved to allow realistic conformations.
- The cube has size 2.

- Two points are *connected* (next) iff
  \[|x_i - x_j|^2 + |y_i - y_j|^2 + |z_i - z_j|^2 = 2,\]
- Each point has 12 neighbors (but 60° and 180° can be removed).
The protein folding problem
HP on FCC

- Backofen and Will fold HP-proteins up to length 200 on FCC using constraint programming
- Clever propagation, an idea of stratification and some geometrical results on the lattice.
- Drawbacks: It is only an abstraction. The solutions obtained are far from reality. For instance, helices and sheets are never obtained.
- Problems:
  - Energy function too simple.
  - Contact too strict.
The protein folding problem
A more realistic Energy function

- A $20 \times 20$ potential matrix $\text{Pot}$ storing the contribution for each pair of amino acids is used.
- Values are either positive or negative.
- The notion of contact (easy) on lattice models is slightly extended:
  - if distance $(a_i, a_j) < k$ then $\text{Pot}(a_i, a_j)$ else $\frac{\text{Pot}(a_i, a_j)}{\text{distance}^2}$
The protein folding problem
A more realistic Energy function

- A $20 \times 20$ potential matrix $\text{Pot}$ storing the contribution for each pair of aminoacids is used.
- Values are either positive or negative.
- The notion of contact (easy) on lattice models is slightly extended:
  - if distance $(a_i, a_j) < k$ then $\text{Pot}(a_i, a_j)$ else $\frac{\text{Pot}(a_i, a_j)}{\text{distance}^2}$
- COLA (COnstraint solving on LAttices) can predict on FCC proteins of length 100–120 in reasonable time
Let $X_1, \ldots, X_n$ be variables with domains $D_1, \ldots, D_n$:

$$\text{contiguous}(X_1, \ldots, X_n) = (D_1 \times \cdots \times D_n) \setminus \{(a_1, \ldots, a_n) \in (D_1 \times \cdots \times D_n) : \exists i. (1 \leq i < n \land (a_i, a_{i+1}) \notin E)\}$$

where $E$ is the set of lattice edges.

- CON (consistency checking) and GAC (generalized arc consistency filtering) are polynomial
Global constraints

contiguous

Let $X_1, \ldots, X_n$ be variables with domains $D_1, \ldots, D_n$:

$$\text{contiguous}(X_1, \ldots, X_n) = (D_1 \times \cdots \times D_n) \ \setminus \ \{(a_1, \ldots, a_n) \in (D_1 \times \cdots \times D_n) : \exists i. \ (1 \leq i < n \ \land \ \ (a_i, a_{i+1}) \notin E)\}$$

where $E$ is the set of lattice edges.

CON (consistency checking) and GAC (generalized arc consistency filtering) are polynomial
Let $X_1, \ldots, X_n$ be variables with domains $D_1, \ldots, D_n$:

$$\text{contiguous}(X_1, \ldots, X_n) = (D_1 \times \cdots \times D_n) \setminus \{(a_1, \ldots, a_n) \in (D_1 \times \cdots \times D_n) : \exists i. (1 \leq i < n \land (a_i, a_{i+1}) \notin E)\}$$

where $E$ is the set of lattice edges.

CON (consistency checking) and GAC (generalized arc consistency filtering) are polynomial
Global constraints

**alldifferent**

Let $X_1, \ldots, X_n$ be variables with domains $D_1, \ldots, D_n$:

\[
\text{alldifferent}(X_1, \ldots, X_n) = (D_1 \times \cdots \times D_n) \setminus \\
\{(a_1, \ldots, a_n) \in (D_1 \times \cdots \times D_n) : \\
\exists i, j. (1 \leq i < j \leq n \land a_i = a_j)\}
\]

CON and GAC are polynomial
Global constraints

alldifferent

Let $X_1, \ldots, X_n$ be variables with domains $D_1, \ldots, D_n$:

$$\text{alldifferent}(X_1, \ldots, X_n) = (D_1 \times \cdots \times D_n) \setminus \{(a_1, \ldots, a_n) \in (D_1 \times \cdots \times D_n) : \exists i, j. \ (1 \leq i < j \leq n \land a_i = a_j)\}$$

CON and GAC are polynomial
Global constraints

alldifferent

Let $X_1, \ldots, X_n$ be variables with domains $D_1, \ldots, D_n$:

$$\text{alldifferent}(X_1, \ldots, X_n) = (D_1 \times \cdots \times D_n) \setminus \{(a_1, \ldots, a_n) \in (D_1 \times \cdots \times D_n) : \exists i, j. (1 \leq i < j \leq n \land a_i = a_j)\}$$

- CON and GAC are polynomial
Global constraints

self avoiding walk

- Given $n$ variables $X_1, \ldots, X_n$, with domains $D_1, \ldots, D_n$, the global constraint $\text{saw}$ is the following:

$$\text{saw}(X_1, \ldots, X_n) = \text{alldifferent}(X_1, \ldots, X_n) \cap \text{contiguous}(X_1, \ldots, X_n)$$

- CON (and GAC) are NP-complete (Dal Palù, Dovier, Pontelli. IJDMB 4(1), 2010)

- Other global constraints have been studied (all distant, chain, rigid block, density maps)
Global constraints

self avoiding walk

Given $n$ variables $X_1, \ldots, X_n$, with domains $D_1, \ldots, D_n$, the global constraint $\text{saw}$ is the following:

$$\text{saw}(X_1, \ldots, X_n) = \text{alldifferent}(X_1, \ldots, X_n) \cap \text{contiguous}(X_1, \ldots, X_n)$$

**CON (and GAC) are NP-complete** (Dal Palù, Dovier, Pontelli. IJDMB 4(1), 2010)

Other global constraints have been studied (all distant, chain, rigid block, density maps)
Global constraints

self avoiding walk

Given $n$ variables $X_1, \ldots, X_n$, with domains $D_1, \ldots, D_n$, the global constraint $saw$ is the following:

$$saw(X_1, \ldots, X_n) = \text{alldifferent}(X_1, \ldots, X_n) \cap \text{contiguous}(X_1, \ldots, X_n)$$

- CON (and GAC) are NP-complete (Dal Palù, Dovier, Pontelli. IJDMB 4(1), 2010)
- Other global constraints have been studied (all distant, chain, rigid block, density maps)
Some References

- Approximated results with local search and/or LNS by Hoos et al. and by Van Hentenryck et al.
Fragment assembly

- Small number of angles allowed by a lattice models: large errors are unavoidable for long proteins.
- Difficult to reuse known information from deposited proteins (state-of-the-art methods are largely built upon this idea).
- We would like to model the PSP off-lattice, but using finite domain variables.
- The main idea is to analyze the known proteins and find some statistics between the angles formed by fragments of 4 (or more) amino acids.
- Then, using some clustering (in $\mathbb{R}^3$), assigning a set of available fragments (indexed by an integer) to subsequences of the known protein.
- The approach might be incomplete, however, we (and others) assume that if nature prefers some local shapes we should do it as well.
Preprocessing

The Protein Data Bank contains ≥ 60K protein sequences with their observed 3D structures (X-ray/NMR)
PDB: extract information

We get fragments composed of 4 consecutive amino acids and collect the corresponding shapes (indexed by sequence)
Clustering (same 4-ple, different shapes)

Clustering according to their similarity (RMSD $\leq$ threshold)
White and green form a single cluster
Clustered conformations for AAAA

Each color has a representative and frequency count
Library of fragments

For each 4 aa sequence, store the clustered representatives (RMSD \(\leq 0.5\AA\))

tupla([A,A,A,A],
[0.0,0.0,0.0,0.0,
2.5,-2.8,0.3,
1.9,-3.1,4.0,
-1.9,-3.4,3.6],
Freq, ID).
Combining the blocks

F Y V A H . . .
F Y V A
Y V A H
V A H . . .

How to assemble fragments?
Inductive step: combine the blocks

Two fragments are *compatible* only if the 3 common amino acids have a low RMSD (similar bend angle)
Inductive step: combine the blocks

Each compatible pair of fragments is stored as

\[ \text{next}(F_i, F_j, M) \]

with optimal rotation matrix \( M \) (that rotates \( F_j \) in the reference of \( F_i \))
Inductive step: combine the blocks
The assembly

Given a target sequence, pick the first 4-aa fragment. The protein is grown by attaching compatible fragments (next).
Enriching the model

- Given a $C\alpha$ 4-tuple in 3D, a small degree of freedom for the position of the side chain is allowed.
- Different amino acids have different occupation.
- A pure $C\alpha$-$C\alpha$ model does not keep into account these differences.
- We consider the positions of the centroids of the side chains.
- Roughly, a centroid is the expected center of mass of the side chain.
- We used a model with 4 (real) atoms, plus the centroid. Briefly, 5@-model.
- We skip the CP modeling. We just focus on one global constraint.
The Joined-Multibody Constraint

A *rigid block* $B$ is an ordered list of at least three (distinct) 3D points, denoted by points($B$). start($B$) and end($B$) are the lists of the first three and the last three points of points($B$). For two lists of points $\vec{p}$ and $\vec{q}$, we write $\vec{p} \bowtie \vec{q}$ if they can be perfectly overlapped by a *roto-translation*.
The Joined-Multibody Constraint

- A *rigid block* $B$ is an ordered list of at least three (distinct) 3D points, denoted by points($B$). start($B$) and end($B$) are the lists of the first three and the last three points of points($B$). For two lists of points $\vec{p}$ and $\vec{q}$, we write $\vec{p} \backsim \vec{q}$ if they can be perfectly overlapped by a *roto-translation*.

- A *multi-body* is a sequence $S_1, \ldots, S_n$ of non-empty sets of rigid blocks.
The Joined-Multibody Constraint

- A rigid block $B$ is an ordered list of at least three (distinct) 3D points, denoted by $\text{points}(B)$. $\text{start}(B)$ and $\text{end}(B)$ are the lists of the first three and the last three points of $\text{points}(B)$.
  For two lists of points $\vec{p}$ and $\vec{q}$, we write $\vec{p} \sim \vec{q}$ if they can be perfectly overlapped by a roto-translation.
- A multi-body is a sequence $S_1, \ldots, S_n$ of non-empty sets of rigid blocks.
- A sequence of rigid blocks $B_1, \ldots, B_n$, is called a rigid body if, for all $i = 1, \ldots, n - 1$, $\text{end}(B_i) \sim \text{start}(B_{i+1})$.

Basically, the JM constraint is the formalization of the problem of finding a rigid body from a multi body that fulfills a set of spatial constraints.
FIASCO: Fragment-based Interactive Assembly for protein Structure prediction with COnstraints

Constraint based local search is implemented.
Some References


To conclude, I suggest to: Play with Foldit http://fold.it/portal/
Protein Docking

- Standard methods (ClusPro) rely on a-posteriori filtering of good results (and of an idea of using FFT)
- BiGGER (Barahona and Kripphal) use constraint propagation and symmetry breaking (see Krippahl and Barahona contribution to WCB 15 — and many other publication of the group)
We want to find a primary sequence that will fold in a desired way.

Usually, a simplification is made. Fix some parts (e.g., secondary structures) and replace some of the other amino acids in all possible ways: choose those that minimize the overall energy.

Viricel, Simoncini, Allouche, de Givry, Barbe, and Schiex contribution to WCB 15 — and previous (many) works of the group.

Hugo Bazille and Jacques Nicolas (WCB 14, with ASP)
Systems Biology
A cell contains complex systems of interacting components
E.g. small molecules, DNA, proteins
Each system can be modeled by means of networks

Molecules
- DNA
- mRNA
- Protein
- Metabolite
- Heterogeneous components

Networks
- Transcriptional regulatory network
- Gene regulatory network
- Protein interaction network
- Metabolic network
- Signaling network
Biological Networks

- The problem is to model a network from biological knowledge
- The model has to be validated w.r.t. experimental data
- Data is incomplete, sometimes unreliable
- Models need to be modified, repaired and/or extended
- Models can guide the design of new experiments
Influence Graph

Operon Lactose in E. coli (example from Gebser, Schaub, Thiele, Veber, 2011)

- Simplest type of Gene Regulatory Network
- Edges show how a gene influence other genes
- The influence can be positive or negative
Influence Graphs

- An **influence graph** is a directed graph $G = \langle N, E, \sigma \rangle$ s.t. $\sigma : E \rightarrow \{+, -\}$ is a labeling of the edges.

- $\sigma$ can be partial. We consider it as total in this presentation.

- $i \rightarrow j$ where $\sigma(i, j) = +$ means that $i$ influences positively $j$ (e.g. a positive (negative) variation of the level of $i$ causes a positive (negative) variation of the level of $j$).

- $i \rightarrow j$ where $\sigma(i, j) = -$ means that $i$ influences negatively $j$ (e.g. a positive (negative) variation of the level of $i$ causes a negative (positive) variation of the level of $j$). It is often denoted as $i \longrightarrow j$. 
Influence Graphs

- Among the nodes there are input nodes, where we can increase or decrease the level of some substances.
- From experimental results one builds a set of observations, namely, some partial assignments $\mu : N \rightarrow \{-, +\}$ for the “level” of the nodes.
- One of the first problems is understanding if these partial observations are “consistent”
- $G = (N, E, \sigma)$ and $\mu$ are consistent whether there is a total extension $\mu'$ of $\mu$ (defined for all nodes in $N$) such that for each non-input node $n \in N$ there is an edge $(m, n) \in E$ such that

$$\sigma(m, n)\mu'(m) = \mu'(n)$$

(i.e. $++ = -- = +$, $+-- = --+$, using the rule of sign)
Operon Lactose in E. coli

- \( L_e \)
- \( L_i \)
- \( G \)
- \( \text{LacY} \)
- \( \text{LacI} \)
- \( \text{cAMP-CRP} \)
- \( \text{LacZ} \)
- \( A \)
Operon Lactose in E. coli
Operon Lactose in E. coli
Operon Lactose in E. coli
Some examples

1 2 3 4 5 6 7 8
+ + + + + + + + NO (8)
- - - - + - + YES
- - - - + - + YES
Operon Lactose in E. coli
Some examples

1 2 3 4 5 6 7 8
+ + + + + + + + NO (8)
+ + + + - + - YES
1 2 3 4 5 6 7 8
+ + + + + + - + + + SAT
+ - - + + - + + YES

Agostino Dovier (Univ. of Udine, DIMI)
Operon Lactose in E. coli

Some examples

```
1  2  3  4  5  6  7  8
+  +  +  +  +  +  +  +  NO (8)
+  +  +  +  +  -  +  -  YES
```
Operon Lactose in E. coli

Some examples

1 2 3 4 5 6 7 8
+ + + + + + + + NO (8)
+ + + + + + - + - YES
+ - - - - - + - + YES
+ - - + + - + + YES

Constraints and Bioinformatics
Problem definition

Checking Consistency

Given an influence graph $G = \langle N, E, \sigma \rangle$ and a partial assignment $\mu$ of the nodes $N$, establish whether $G$ and $\mu$ are consistent.
Problem definition

Checking Consistency

Given an influence graph $G = \langle N, E, \sigma \rangle$ and a partial assignment $\mu$ of the nodes $N$, establish whether $G$ and $\mu$ are consistent.

If $\mu$ is total, it is just a polynomial check.
If $\mu$ is partial, it is NP-complete [Veber06]
Problem definition

Checking Consistency

Given an influence graph $G = \langle N, E, \sigma \rangle$ and a partial assignment $\mu$ of the nodes $N$, establish whether $G$ and $\mu$ are consistent.

If $\mu$ is total, it is just a polynomial check.
If $\mu$ is partial, it is NP-complete [Veber06]
We are interested in finding the minimal modifications on edges to make the network consistent.
Influence graphs

Let $G = (V, E)$, $V = \{V_1, \ldots, V_n\}$

Introduce $X_1, \ldots, X_n$ with domain $\{-1, 1\}$ ($-1$ for -, $+1$ for +)

Assign the “known” values $X_i = \sigma(V_i)$.

For $i = 1, \ldots, n$, if $V_i$ is not “input” then, let

$$(V_{i_1}, V_i, \sigma(i_1, i)), \ldots, (V_{i_k}, V_i, \sigma(i_k, i))$$

be its entering edges. Then we set the constraint:

$$V_i \in \{X_{i_1} \sigma(i_1, i), \ldots, X_{i_k} \sigma(i_k, i)\}$$
Problem definition

Once inconsistency has been detected, the biologist would receive some guess on where the error can be. There are several chances. We show one.
Problem definition

Once inconsistency has been detected, the biologist would receive some guess on where the error can be. There are several chances. We show one.

Repairing

Given an influence graph $G = \langle N, E, \sigma \rangle$ and a partial assignment $\mu$ of the nodes $N$: find $\mu'$ such that $G$ and $\mu'$ are consistent and $\mu'$ is obtained from $\mu$ by changing as few values as possible.
Problem definition

Once inconsistency has been detected, the biologist would receive some guess on where the error can be. There are several chances. We show one.

**Repairing**

Given an influence graph $G = \langle N, E, \sigma \rangle$ and a partial assignment $\mu$ of the nodes $N$: find $\mu'$ such that $G$ and $\mu'$ are consistent and $\mu'$ is obtained from $\mu$ by changing as few values as possible.

This can be used for reasoning on the network.
Problem definition

Once inconsistency has been detected, the biologist would receive some guess on where the error can be. There are several chances. We show one.

Repairing

Given an influence graph $G = \langle N, E, \sigma \rangle$ and a partial assignment $\mu$ of the nodes $N$: find $\mu'$ such that $G$ and $\mu'$ are consistent and $\mu'$ is obtained from $\mu$ by changing as few values as possible.

This can be used for reasoning on the network. Similarly, one may ask for the minimum number of edges to be labeled in a different way, or to be added, and so on.
Influence graphs

Repairing

- Let $G = (V, E)$, $V = \{V_1, \ldots, V_n\}$
- Introduce $X_1, \ldots, X_n$ and $D_1, \ldots, D_n$ valued in $\{-1, 1\}$
- Intuitively, $X_i$ is the value of the node $i$, $D_i$ is 1 (-1) if node $i$ is consistent (inconsistent).
- Assign the “known” values $X_i = \sigma(V_i)$.
- For input nodes and for nodes not assigned by $\sigma$: $D_i = 1$
- For $i = 1, \ldots, n$, if $V_i$ is not “input” then, let

$$
(V_{i_1}, V_i, \sigma(i_1, i)), \ldots, (V_{i_k}, V_i, \sigma(i_k, i))
$$

be its entering edges. Then we set the constraints:

$$
V_i D_i \in \{X_{i_1} \sigma(i_1, i), \ldots, X_{i_k} \sigma(i_k, i)\}
$$

- Maximize $D_1 + \cdots + D_n$
Biocham (the BIOCHemical Abstract Machine)

- Biocham (Fages, Soliman et al.) is a software environment for modeling biochemical systems. (e.g., WCB 06, . . . , WCB 13)
- It allows the analysis and simulation of boolean, kinetic and stochastic models (using a rule-based language) and
- the formalization of biological properties in temporal logic (LTL/CTL)
- It uses CLP, SAT and other constraint-based techniques.
- A lot of successful experiments with real data have been performed.
Some references


Conclusions

We have surveyed the three main areas of Bioinformatics, focusing on a pair of problems per area:

- **Genomics:**
  - ✓ Haplotype Inference
  - ✓ Phylogenetic trees

- **Structural Bioinformatics:**
  - ✓ RNA secondary structure prediction
  - ✓ Protein structure prediction (and docking, and engineering)

- **Systems Biology:**
  - ✓ Reasoning on Biological Networks

There’s still a lot to do for us. On the problems seen and on a lot of other problems. CP, in combination with SAT, LS can play a central role in the present (and future) of Bioinformatics.
Three constraints from bioinformatics are enlisted

- **The constraint**: `all_differ_from_at_least_k_pos` is basically an error correcting code generator, inspired by [Frutos et al, Nucleic Acids Research 25, 1997]. Given a set $S$ of vectors it enforce all pairs of distinct vectors in $S$ to differ each other from at least $k$ positions.

- **The constraint** `sequence_folding` (by Justin Pearson) is a global constraint that can be used in the encoding of the RNA secondary structure prediction problem. It explicitly avoids “pseudo knots” (in this case, however, the problem is in $P$).

- **The stable_compatibility constraint** (by Pierre Flener, inspired by [Beldiceanu et al, CPAIOR 2006]) used for supertree reconstruction. Subsequent works by Moore and Prosser [JAIR2008] improve it.

The saw and the JM constraint deserve to be added.
Conclusions

Acknowledgments

Thank you!

- CP/ICLP organizers (in particular Willem-Jan Van Hoeve and Mats Carlsson)
- My main collaborators/co-authors in Bioinformatics:
  - Ferdinando Fioretto
  - Enrico Pontelli
  - Alessandro Dal Palù
  - Federico Fogolari
  - Federico Campeotto
  - Andrea Formisano

- and the friends that helped in the organizations of WCB 05–15:
  - Rolf Backofen, Sebastian Will, Francois Fages, Nicos Angelopoulos, Simon de Givry