Protein Threading

Combinatorial optimization approach

Stefan Balev

`Stefan.Balev@univ-lehavre.fr`

Laboratoire d'informatique du Havre – Université du Havre

Outline

- Biological background
- Mathematical model
- ...

Biological background

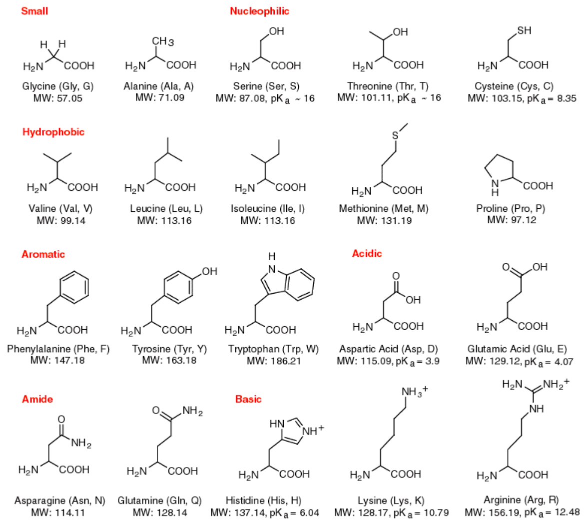
Proteins

replication

$$\text{DNA} \xrightarrow{\text{transcription}} \text{RNA} \xrightarrow{\text{translation}} \text{proteins}$$

- complex biological macromolecules composed of sequences of 20 *amino acids* (50-2500)
- key elements of many cellular functions
  - *fibrous proteins* contribute to hair, skin, bone, muscles,...
  - *membrane proteins* mediate the exchange of molecules and information across cellular boundaries
  - *globular proteins* mediate and catalyze the biochemical reactions
- The human genome codes 100,000 proteins, bacteria 500-1500

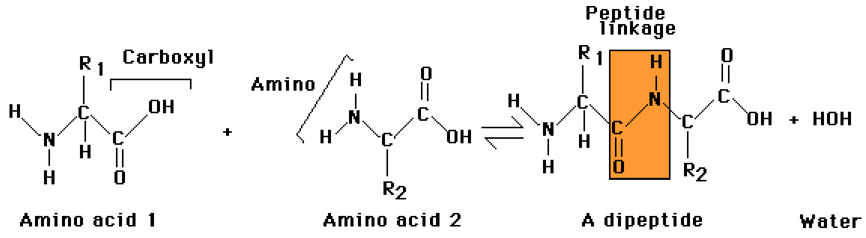
Amino acids



Genetic code

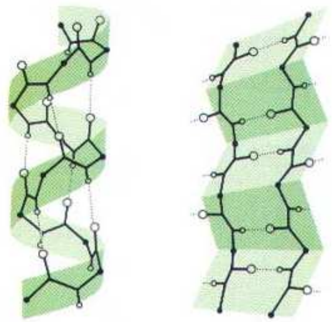
	T		C		A		G	
T	TTT	**PHE**	TCT	**SER**	TAT	**TYR**	TGT	**CYS**
	TTC	**PHE**	TCC	**SER**	TAC	**TYR**	TGC	**CYS**
	TTA	**LEU**	TCA	**SER**	TAA	**stop**	TGA	**stop**
	TTG	**LEU**	TCG	**SER**	TAG	**stop**	TGG	**TRP**
C	CTT	**LEU**	CCT	**PRO**	CAT	**HIS**	CGT	**ARG**
	CTC	**LEU**	CCC	**PRO**	CAC	**HIS**	CGC	**ARG**
	CTA	**LEU**	CCA	**PRO**	CAA	**GLN**	CGA	**ARG**
	CTG	**LEU**	CCG	**PRO**	CAG	**GLN**	CGG	**ARG**
A	ATT	**ILE**	ACT	**THR**	AAT	**ASN**	AGT	**SER**
	ATC	**ILE**	ACC	**THR**	AAC	**ASN**	AGC	**SER**
	ATA	**ILE**	ACA	**THR**	AAA	**LYS**	AGA	**ARG**
	ATG	**MET**	ACG	**THR**	AAG	**LYS**	AGG	**ARG**
G	GTT	**VAL**	GCT	**ALA**	GAT	**ASP**	GGT	**GLY**
	GTC	**VAL**	GCC	**ALA**	GAC	**ASP**	GGC	**GLY**
	GTA	**VAL**	GCA	**ALA**	GAA	**GLU**	GGA	**GLY**
	GTG	**VAL**	GCG	**ALA**	GAG	**GLU**	GGG	**GLY**

Polypeptide chains



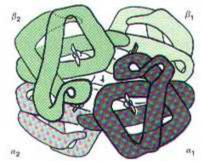
The four levels of protein structure

primary (1D) structure (amino acid sequence in the protein chain)



secondary (2D) structure
(α-helices and β-sheets)

tertiary (3D) structure



quaternary structure

The importance of protein structure

The structure determines the function (?). One cannot understand biological reactions without understanding the structure of the participating molecules.

Goals:

- **protein engineering**: mutating the gene of an existing protein in order to alter its function in a predictable way.
- **protein design**: designing *de novo* a protein to fulfill a desired function

Applications:

- pharmaceutical industry (drug design, drug docking)
- chemical industry (enzymes)
- agriculture (pesticides, herbicides, modification of composition of plant oils,...)

Protein folding problem

- **Input**: $a_1 a_2 \ldots a_N$ – a word over the 20-letter amino acid alphabet
- **Output**: $(x_j, y_j, z_j), j = 1, \ldots, N$ – the coordinates of a_j

Structure determination methods

- Experimental (*in vitro*) methods: x-ray crystallography, nuclear magnetic resonance. Slow and expensive, cannot cope with the explosion of sequences becoming available.
- Computational (*in silico*) methods. Still not accurate and reliable enough.
  - **Direct approach**: based on modeled atomic force fields and approximation from classical mechanics, seeks to minimize the free energy. Difficult to model, sensitive to approximation errors, computationally expensive.
  - **Protein threading**: seeks to assign a protein to already-known structure. Still with limited capabilities, but promising and evolving method.

Protein threading – basic assumptions

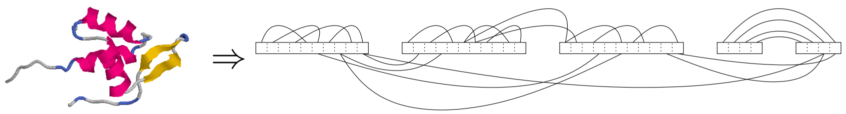
- the sequence (1D structure) determines the 3D structure
- homologous proteins have similar structure (and function)
- homologous proteins have conserved structural cores and variable loop regions
- there are only around 1,000 and 10,000 different protein structural families

Protein threading – main steps

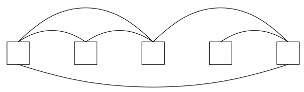
- constructing a library of potential core folds (structural templates)
- choosing an objective function (score function) to evaluate any alignment of a sequence to a structure template
- finding the best (with respect to the score function) alignment of the query sequence and each structural template in the library
- choosing the most appropriate template based on the (normalized) scores of the optimal alignments found on the previous step

Mathematical model

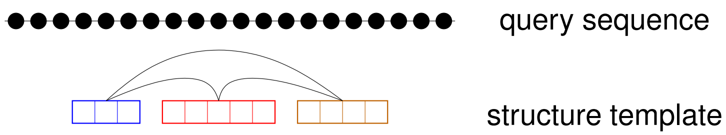
Structure templates



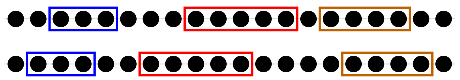
- Set of m *blocks* of length l_i, $i = 1, \ldots, m$. The blocks correspond to conserved elements of the 2D structure (α-helices and β-sheets)
- *Contact map graph* – describes the interactions between the amino acids in the blocks
- *Generalized contact map graph* – describes the interactions between the blocks



Query-to-structure alignment

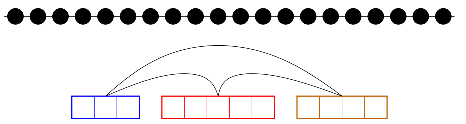


Alignment (threading): covering of segments of of the query sequence by the template blocks

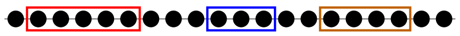


A threading is completely determined by the starting positions of the blocks

Query-to-structure alignment – rules



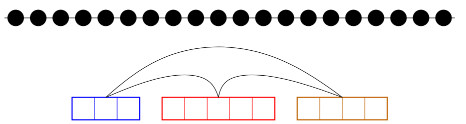
- the blocks preserve their order



- the blocks do not overlap

- no gaps in the blocks

Absolute and relative positions



abs. position	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
rel. position block 1	1	2	3	4	5	6	7	8	9											
rel. position block 2				1	2	3	4	5	6	7	8	9								
rel. position block 3									1	2	3	4	5	6	7	8	9			

If j is the absolute position of block i, then $\pi_i = j - \sum_{k=1}^{i-1} l_k$ is its *relative* position

The relative position of each block is between 1 and $n = N + 1 - \sum_{i=1}^{m} l_i$

The set of threadings is $T = \{(\pi_1, \ldots, \pi_m) \mid 1 \le \pi_1 \le \cdots \le \pi_m \le n\}$

The number of possible threadings is $|T| = \binom{m+n-1}{m}$

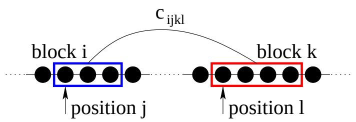
(Example: for $m = 20$ and $n = 100$, $T \approx 2.5 \times 10^{22}$)

Score function

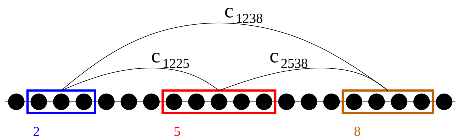
- incorporates all the biological and physical knowledge on the problem
- describes the degree of compatibility between sequence residues and their positions in the structure template
- essential for the quality of the threading
- we assume that
  - is additive
  - it can be computed considering no more than two blocks at a time

Score function

c_{ijkl}, $(i, k) \in E$, $1 \le j \le l \le n$ – score for putting block i on position j and block k on position l



$$\varphi(\pi) = \sum_{(i,k) \in E} c_{i\pi_i k\pi_k}$$



$$\varphi(2, 5, 8) = c_{1225} + c_{1238} + c_{2538}$$

The coefficients c_{ijkl} are precomputed and stored before the start of threading algorithm

Protein threading problem

$$\min\{\varphi(\pi) \mid \pi \in T\}$$

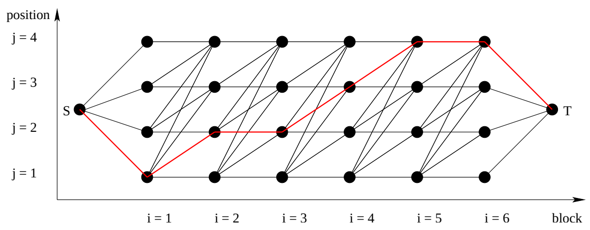
where:

$$\varphi(\pi) = \sum_{(i,k)\in E} c_{i\pi_i k\pi_k}$$

$$T = \{(\pi_1, \ldots, \pi_m) \mid 1 \le \pi_1 \le \cdots \le \pi_m \le n\}$$

The problem is known to be NP-hard and MAX-SNP-hard.

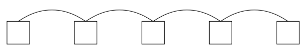
Network flow formulation



Each possible threading corresponds to a path from S to T in the graph and vice versa.

The red path correspons to the threading $(1, 2, 2, 3, 4, 4)$

Simple case – no remote links

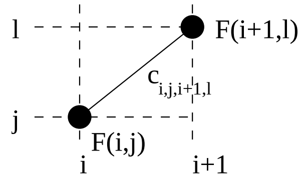


The costs $c_{i,j,i+1,l}$ are associated to the corresponding edges.

The problem reduces to finding the shortest path from S to T.

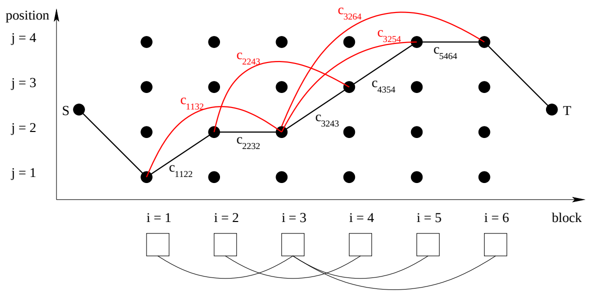
This can be done by simple dynamic programming:

$$F(i+1, l) = \min_{j=1,\ldots,l}\{F(i, j) + c_{i,j,i+1,l}\}$$



Complexity $O(mn^2)$

Taking into account the remote links



A path from S to T activates complementary edges corresponding to the remote links. We call it *augmented path*.

Protein threading problem: find the augmented path of minimal length.

Integer programming models

Non-linear model

Variables:

$$y_{ij} \in \{0,1\},\ i = 1,\dots,m,\ j = 1,\dots,n$$

$y_{ij} = 1 \Leftrightarrow$ block i is on position j

Objective function:

$$f(y) = \sum_{(i,k)\in E} \sum_{1\le j\le l\le n} c_{ijkl} y_{ij} y_{kl}$$

Constraints:

$\sum_{j=1}^{n} y_{ij} = 1$ block i is on exactly one position

$y_{i+1,l} \le \sum_{j=1}^{l} y_{ij}$ if block $i+1$ is on position l then block i is before position l

Linear model (1)

Variables:

$x_{i,j,i+1,l} = 1 \Leftrightarrow$ block i is on position j and block $i+1$ is on position l

$z_{ijkl} = 1 \Leftrightarrow$ block i is on position j and block k is on position l

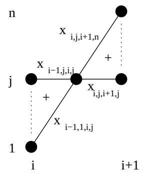
Objective function:

$$f(x,z) = \sum_{i=1}^{m-1} \sum_{1\le j\le l\le n} c_{i,j,i+1,l} x_{i,j,i+1,l} + \sum_{(i,k)\in R} \sum_{1\le j\le l\le n} c_{ijkl} z_{ijkl}$$

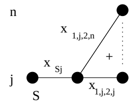
Linear model (2)

Network flow constraints:

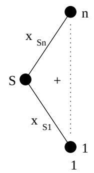
$$\sum_{l=1}^{j} x_{i-1,l,i,j} = \sum_{l=j}^{n} x_{i,j,i+1,l}$$



$$x_{Sj} = \sum_{l=j}^{n} x_{1j2l}$$



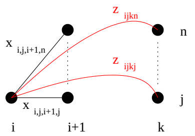
$$\sum_{j=1}^{n} x_{Sj} = 1$$



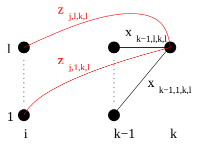
Linear model (3)

Constraints connecting x and z:

$$\sum_{l=j}^{n} x_{i,j,i+1,l} = \sum_{l=j}^{n} z_{ijkl}$$



$$\sum_{j=1}^{l} x_{k-1,j,k,l} = \sum_{j=1}^{l} z_{ijkl}$$



Other linear models

$x \in X$ – feasible threadings in terms of x variables (network flow constraints)

$y \in Y$ – feasible threadings in terms of y variables (as in the nonlinear model)

Starting from X (or Y) one can add different sets of constraints connecting z-variables to x-variables (or y variables).

Such models are proposed and compared in [Balev et al.]

Solving the IP models

LP relaxation

IP problem

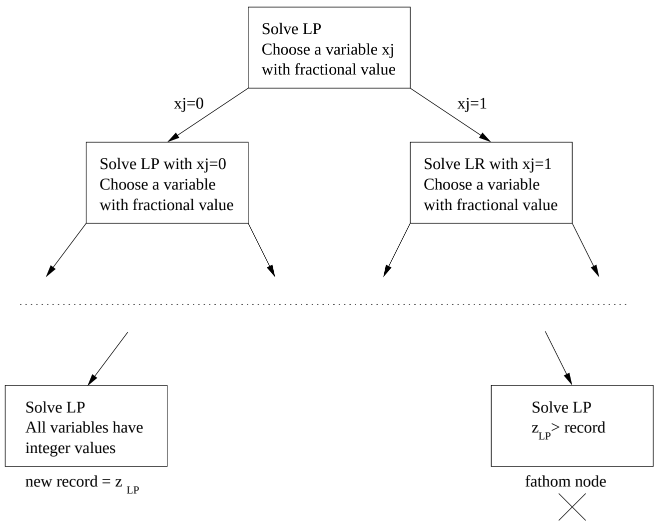
$$\begin{aligned} z_{\text{IP}} = \;& \min cx \\ \text{s.t.}\;& x \in X \text{ – linear constraints} \\ & x \in \{0,1\} \text{ – integrality constraints} \end{aligned}$$

LP relaxation

$$\begin{aligned} z_{\text{LP}} = \;& \min cx \\ \text{s.t.}\;& x \in X \\ & 0 \le x \le 1 \end{aligned}$$

- the LP relaxation is easier to solve than the IP problem (simplex method)
- the LP relaxation provides a lower bound on the optimal objective value of the IP problem (i.e. $z_{\text{LP}} \le z_{\text{IP}}$)

Branch-and-bound using LP



Experimental results

Instances are solved by CPLEX (a general purpose LP/IP solver).

Conclusions:

- much faster than the previously used methods [Lathrop et al]
- the efficiency depends on the way the model is formulated
- for more than 90% of real life instances the optimal solution is found in the root of the B&B tree (i.e. $z_{\mathsf{LP}} = z_{\mathsf{IP}}$), the rest 10% require less than 10 nodes
- this is not the case for randomly generated instances (why?)
- for large instances even solving the LP relaxation is slow because of the big number of variables and constraints in the model
- Running time: from 30s on instances of size 10^{20} to 2h on instances of size 10^{40}

Lagrangian relaxation and duality

Idea: drop a part of constraints making the problem easier to solve, introduce penalties for violating them in the objective function

IP problem:

$z_{\mathsf{IP}} = \quad \min cx$

s.t. $\quad x \in X$ – "easy" constraints

$Ax = b$ – "complicating" constraints

Lagrangian relaxation: $z_{\mathsf{LR}}(\lambda) = \min\{cx + \lambda(b - Ax) \mid x \in X\}$

Lagrangian dual: $z_{\mathsf{LD}}(\lambda) = \max_\lambda z_{\mathsf{LR}}(\lambda)$

- LR is also an IP problem, but easier to solve than IP
- LR is relaxation of IP for *any* λ (i.e. $z_{\mathsf{LR}}(\lambda) \le z_{\mathsf{IP}}$)
- LD is better than LP : $z_{\mathsf{LP}} \le z_{\mathsf{LD}} \le z_{\mathsf{IP}}$

Solving LD by subgradient optimization

Initialization: $\lambda^0 = 0, \theta_0, t = 0$

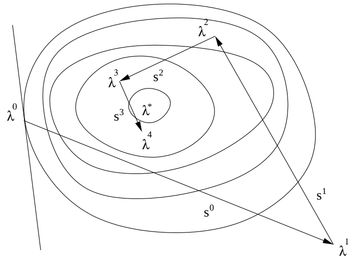
Iteration t: Solve LR(λ^t). Let x^t be the solution found.

Let $s^t = b - Ax^t$ (s^t is subgradient of $z_{\mathsf{LR}}(\lambda)$ for $\lambda = \lambda^t$).

If $s^t = 0$ then stop (x^t is optimal solution of IP).

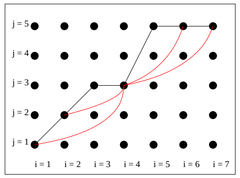
Otherwise let $\theta_{t+1} = \theta_t \rho$. If $\theta_{t+1} < \varepsilon$ then stop.

Otherwise let $\lambda^{t+1} = \lambda^t + \theta_{t+1} s^t$, $t = t + 1$

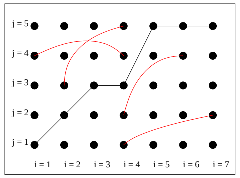


LR for protein threding

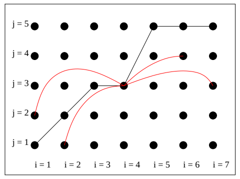
The "complicating" constraints are those connecting x- and z-variables



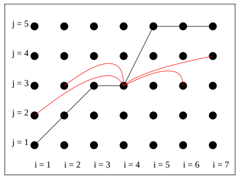
(a) original problem



(b) all connecting constraints are relaxed



(c) the constraints corresponding to one of the ends of each link are relaxed



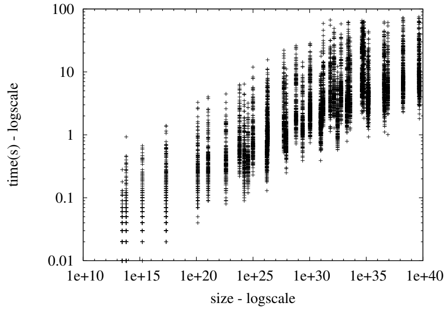
(d) like (c) but order on the free ends of the links is imposed

Solving protein threading problem

- LR is solved using dynamic programming algorithm similar to the one proposed by Lathrop et al. Complexity $O((m+r)n^2)$, where r is the number of remote links between the blocks. In the worst case this is $O(m^2n^2)$, but for real-life instances it is $O(mn^2)$.
- LD is computed using subgradient optimization limited to 500 iterations.
- Protein threading problem is solved by branch-and-bound algorithm using the LD bounds.

Experimental results

Running times for 10,000 instances



- much faster than LP relaxation
- the optimal solution is found in the root in more than 90% of the cases
- the algorithm is less sensitive to perturbations in the score coefficients

FROST

FROST (Fold Recognition Oriented Search Tool) – a threading tool developed by MIG, INRA (J.-F. Gibrat, A. Marin et al)

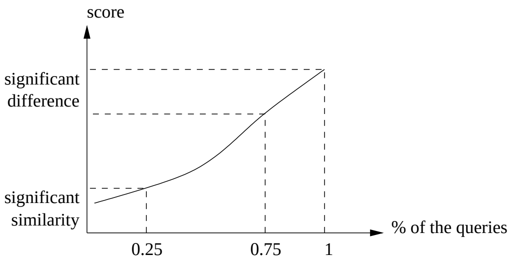
- contains a library of about 1,200 structure templates.
- optimization algorithms in FROST
  - branch-and-bound (Lathrop et al.)
  - a steepest descent heuristic (Zimmermann)
  - branch-and-bound using LP relaxation (Balev et al)
  - branch-and-bound using Lagrangian relaxation (Balev)

Score normalization in FROST (1)

The scores of alignment of a query to two templates are not directly comparable.

To normalize the scores FROST aligns each template from the database to 1,000 query sequences. The score distribution is approximated by this empirical data.

When a “real” query is threaded to the template the score is interpreted according to the distribution



Score normalization in FROST (2)

- The computing of score distributions involves about 1,200,000 threadings
- It must be repeated after each modification in the score scheme
- Using the other algorithms it takes about **3 months** on a cluster of 16 PCs
- Using Lagrangian relaxation algorithm it takes less than a **week**