The Threading Problem

adapted from a lecture by
Dr. Axel Mosig

CAS-MPG Partner Institute for Computational Biology (PICB) Shanghai, China

http://www.picb.ac.cn/patterns

May 30, 2011
Predicting Protein Tertiary Structures

- approx. 650–10000 different tertiary structures
- even sequences no obvious sequence similarity can fold into similar tertiary structures
- **Idea of threading:** utilize a known tertiary structure and “thread” the unknown structure into it
• **Idea:** Essential for tertiary structures are often structurally highly conserved, e.g. those parts that fold into $\alpha$-helices or $\beta$-strands.

• Transitions between these conserved parts are less relevant.

• Secondary structure of a sequence $s$ with $m$ components ($\alpha$-Helices, $\beta$-Strands) as abstract model:
Threadings-Models

• Length of transitions between sequence parts ($\lambda_i$) underly certain conditions:

$$\ell_i \leq \lambda_i \leq L_i.$$ 

Definition
A Core Model $M$ is a 5-tupel $M = (m, c, \lambda, \ell, L)$, where

- $m \equiv$ number of sec. struct. elements
- $c = (c_1, \ldots, c_m) \equiv$ length of the segments
- $\lambda = (\lambda_0, \ldots, \lambda_m) \equiv$ length of the transitions
- $\ell = (\ell_0, \ldots, \ell_m) \equiv$ lower
- $L = (L_0, \ldots, L_m) \equiv$ upper bounds for transition lengths
Threading a sequence into a model

- structure $s$ with model $M$; thread sequence $s'$ into $M$.
- Goal of threading: sec. struct. elements are mapped onto subsequences of same length in $s'$ length of transitions may vary (within bounds)
- threading representable as a sequence $t_1, \ldots, t_m$
Formal Definition of a Threading

Definition
Let $s'$ be sequence of length $n'$ and $M$ a core-Model. A sequence $t = (t_1, \ldots, t_m)$ is called a threading of $s'$ through $M$, if

(T1) $1 + \ell_0 \leq t_1 \leq 1 + L_0$
(T2) $t_i + c_i + \ell_i \leq t_{i+1} \leq t_i + c_i + L_i$ for $1 < i < m$ and
(T3) $t_m + c_m + \ell_m \leq n' + 1 \leq t_m + c_m + L_m$

- In general, given model $M$ and sequence $s$, there are many threadings satisfying (T1)–(T3).
- Which of those is best? $\rightsquigarrow$ scoring-function
Scoring-Functions: Structure

- Scoring function $f$ has two ingredients:
  - How well “matches” a segment of $s'$ into a segment $C_i$?
    $\Rightarrow g_1(i, t_i)$
  - Extendable to higher-order interactions e.g. of triplets of elements
    $g_3(i, j, k, t_i, t_j, t_k)$ ...

- $g_1, g_2$ are based on knowledge-based approaches
- $g_2$ e.g. through pairwise potentials $\Rightarrow$ Sippl (1990/1995)
Scoring-functions: interaction graphs

- Segments $C_i$ and $C_j$ from model $M$ do not interact
  $\Rightarrow g_2(i, j, k, k') = 0$ for all $k, k'$

- interaction graph: Graph $G_I$ with vertices $V_I = \{1, \ldots, m\}$ and nodes
  $$E_I = \{(i,j) \mid \exists k, k': g_2(i, j, k, k') \neq 0\}.$$

- Scoring-function for $t = (t_1, \ldots, t_m)$ formally:
  $$f(t) = \sum_{i \in [1:m]} g_1(i, t_i) + \sum_{(i,j) \in E_I} g_2(i, j, t_i, t_j)$$
Threading as Optimization problem

• Given Core-Model $M$ for sequence $s$ and sequence $s'$ with unknown tertiary structure
• Wanted: $\min_t f(t)$
• Computing $\min_t f(t)$ is (MAX-S)NP-hard: Akutsu/Miyano (1999)
  $\Rightarrow$ Backtracking-algorithm ("brute-force")
  $\Rightarrow$ Branch-and-Bound-algorithm by Lathrop and Smith (1996)
• Without $g_2$ solvable in polynomial time (dynamic programming)
Relative Threading

- **Goal:** “Address” all possible threadings $T_M(s')$ for sequence $s'$ into a model $M$ for traversing $T_M(s')$ systematically
- Let $t = (t_1, \ldots, t_m)$ a threading of $s'$ through $M$.
- Relative threading $t' = (t'_1, \ldots, t'_m)$ to $t$ is defined as

$$t'_i := t_i - \sum_{j<i} (c_i + \ell_i).$$
Scaffold for B-&-B-Algorithms

branch-and-bound($X$)
    $S$.push($X$);
    $x_{opt} := \infty$;
    while (!$S$.empty())
        $Y = S$.pop();
        if ($B(Y) < x_{opt}$) then
            if ($Y == \{t'\}$) then
                if ($f(t') < x_{opt}$) then $x_{opt} := f(t')$;
            else
                split $Y$ into $Y_L$ and $Y_R$
                $S$.push($Y_L$);
                $S$.push($Y_R$);
        end.


Threading using Branch-and-Bound

- Branch-and-Bound-algorithm traverses a spanning tree of sets of solutions
- Cutting-bounds allow to drop parts of the solution tree
- We need:
  - Sets of threadings that can be decomposed into parts
  - Lower bounds for sets of threadings that can be easily computed
Threading-Sets

- Define intervals $[b_i : d_i]$ (for $1 \leq i \leq m$)
- $\Rightarrow$ vectors $b = (b_1, \ldots, b_m)$ and $d = (d_1, \ldots, d_m)$.
- Yields set

$$T_M(b, d) = \{ t' = (t'_1, \ldots, t'_m) \mid b_i \leq t'_i \leq d_i, \quad t' \text{ is rel. threading} \}$$

of (relative) threadings.
- $T_M(1, n') = T_M(s')$
Splitting Threading sets ("Branch")

- Choose $i$ such that $b_i < d_i$.
- Divide Intervals $[b_i : d_i]$ into $[b_i : v]$ and $[v + 1 : d_i]$.
- Define analogous vectors $b', d'$ and $b'', d''$.
- $T_L := T_M(b', d')$ and $T_R := T_M(b'', d'')$ yield split of $T_M(b, d)$. 
Lower Bounds for Threading-Sets

- **Wanted:** Lower bound $B_M(b, d)$ with properties
  - $B_M(b, d) \leq \min_{t' \in T_M(b, d)} f(t')$
  - $B_M(b, d)$ should be computable fast

- **Choose**

\[
B(b, d) := \sum_i \left( (\min_{x \in [b_i : d_i]} g_1^i(i, x) + \sum_{j < i} \min_{x, y} g_2(i, j, x, y) ) \right)
\]
B-&-B-Threading-algorithm

\begin{verbatim}
thread(s, M)
    S.push(1, n);
    x_{opt} := \infty;
    while (!S.empty())
        (b, d) = S.pop();
        if (B(b, d) < x_{opt}) then
            if (\text{T}_M(b, d) == \{t'\}) then
                if (f(t') < x_{opt}) then x_{opt} := f(t');
            else
                split (b, d) into (b', d') and (b'', d'')
                if (\text{T}_M(b', d') \neq \emptyset) then
                    S.push((b', d'));
                if (\text{T}_M(b'', d'') \neq \emptyset) then
                    S.push((b'', d''));
        end
\end{verbatim}
How complex is Protein-Threading?

• B-&-B-algorithm is faster than naive Bachtracking, but still exponentiel worst-case running time
• threading-Problem is MAX-SNP-complete
• **Means:** we won’t even get good approximate solutions in polynomial time unless $P \neq NP$!
• How “complex” is the interaction graph?
• Diverse successful structure predictions (CASP)

