Bioinformatics

Lecture 23
Protein folding

• The goal is to determine the three-dimensional structure of a protein based on its amino acid sequence

• Assumption: amino acid sequence completely and uniquely determines the folding
  – unfold a protein and then release it
  – it immediately folds back to the three-dimensional structure it had before, its “native” structure

• Protein secondary structures
  – $\alpha$-helices
  – $\beta$-sheets
  – Neither helices nor sheets, called loops
α-helix

• An α-helix is a simple helix having on average 10 residues (3 turns of the helix)

• Some helices can have as many as 40 residues

• Some amino acids appear more frequently on helices than other amino acids, not a strong enough fact to allow accurate prediction
β-sheet

- A β-sheet consists of binding between several sections of the amino acid sequence

- Each participating section is called a β-strand and has generally 5 to 10 residues

- The strands become adjacent to each other forming a kind of twisted sheet

- Certain amino acids show a preference for being in a β-sheet, but again these preferences are not so positive to allow accurate prediction
Loop

• A loop is a section of the sequence that connects the other two kinds of secondary structure

• Loops are not regular structures both in shape and size

• In general, loops are outside a folded protein, whereas the other structures form the protein core
A folded protein

- α-helix
- β-sheet
- loop
Motifs and Domains

• A motif is a simple combination of a few secondary structure that appear in several different proteins
  
  – e.g. helix-loop-helix

• A motif might serve as a binding site to other molecules or may have no role at all.

• A domain is a more complex combination of secondary structures that have a very specific function; therefore, it contains a binding site (called active site)
Protein folding

Given the amino acid sequence of a protein, determine:

– where exactly all of its α-helices, β-sheets, and loops are, and

– how they arrange themselves in motifs and domains
Greedy Approach

- Given enough chemical and physical information about each amino acid it should be possible to compute the free energy of a folding.

- Enumerate all possible foldings, compute the free energy of each.

- Choose the folding with the minimum free energy (assuming that such a folding is the protein’s native structure).
Feasibility of greedy

• Recall that proteins fold thanks in large to the angles $\phi$ and $\psi$ between the carbon and the neighboring atoms.

• These angles can assume only a few values independently of each other.

• Therefore each residue can have a configuration given by a pair of values for $\phi$ and $\psi$.

• Assume both $\phi$ and $\psi$ can assume 3 values and the protein is 100 residues long, then we have to examine $9^{100}$ foldings!
Other problems with greedy

• No agreement on how to compute the free energy of a folding, too many factors to consider
  – Shape
  – Size
  – Polarity of molecules
  – Strength of interactions of molecules, etc…

• Because of all these difficulties, other techniques have been developed, e.g. Protein Threading
Similarity of protein sequences

- An early method for secondary structure prediction was based on the idea that similar sequences should have similar structures
  - folding of A known
  - B’s sequence is similar to A’s sequence
  - Folding of B is similar to A’s

Not generally true!

- Similar proteins at the sequence level may have different secondary structures

- On the other hand, certain proteins that are very different at the sequence level are structurally related: different loops, similar cores

- Protein threading: fit a known structure to a sequence
Protein threading

- We are given
  - A protein sequence $A$ with $n$ amino acids $a_i$
  - A core structural model $C$, with $m$ core segments, we have:
    - The length $c_i$ of each core segment
    - Core segments $i$ and $i+1$ are connected by loop for which we know the maximum $l_i^{\text{max}}$ and minimum $l_i^{\text{min}}$ lengths
    - Properties of each amino acid in $C$
  - A score function $f$ to evaluate a threading

- We want to find a best scoring set $T = \{t_1, \ldots, t_m\}$ of integers such that each $t_i$ indicates what amino acid from $A$ occupies the first position from core segment $i$
If we allow pairwise interactions, the problem becomes NP-hard.
Approach

• We will allow pairwise interactions of amino acids

• We will solve the problem exactly with a standard technique used for handling NP-hard problems

• This technique is known as branch-and-bound
Branch-and-Bound

• Assume we want to find the maximum $f(s)$ among many solutions $s$ in the solution space

• First, it should be possible to separate the solution space into subspaces according to some constraints [branch]
  
  – e.g. given a solution space $X$ we could partition it into $X_1$ (all solutions that have a certain property) and $X_2$ (all solutions that do not)

• Second, for every partition $X$ obtain an upper bound on the value of $f(s)$ for every solution $s \in X$ [bound]

• Assume we have $f(s)$ for a given $s$. Now consider a subset $X$ of the solution space with an upper bound $u$. If $f(s) > u$, then we can discard $X$. 
Some considerations

• The upper bound should be as close to the actual function value as possible, so that we can find the maximum faster than with a weaker upper bound

• The upper bound should also be efficient to compute

• Even if we speedup the process by discarding some subsets of the solution space, the worst case running time is still exponential
Finding the solution space

Every solution must satisfy:

\[ 1 + \sum_{j<i}(c_j+l_j^{\min}) \leq t_i \leq n+1 - \sum_{j\geq i}(c_j+l_j^{\min}) \]

\[ t_i + c_i + l_i^{\min} \leq t_{i+1} \leq t_i + c_i + l_i^{\max} \]

This implies that each \( t_i \in [b_i, e_i] \) in every solution
Illustration

A set of solutions is given by a collection of intervals, one for each of the $m t_i$'s.
Branch

Given a set of solutions:

- choose a segment, say $i$

- choose a position $u_i$ inside the interval $[b_i, e_i]$

- Split the set into three sets in which the intervals for $t_i$ will be:
  
  - $[b_i, u_{i-1}]$
  - $[u_{i+1}, e_i]$
  - $[u_i]$
Bound

- Given a solution \( s \), \( f(s) \) can be expressed as:

\[
f(s) = \sum_i g_1(i, t_i) + \sum_{j>i} g_2(i, j, t_i, t_j)
\]

score for each segment

score for interactions

- Given a set \( X \) of solutions, a simple upper bound is:

\[
\max_{s \in X} f(s) = \max_{s \in X} \sum_i \left[ g_1(i, t_i) + \sum_{j>i} g_2(i, j, t_i, t_j) \right]
\]

\[
\leq \sum_i \left[ \max_{b_i \leq x \leq e_i} g_1(i, x) + \sum_{j>i} \max_{b_i \leq y, z \leq e_i} g_2(i, j, y, z) \right]
\]
Algorithm

$X \leftarrow$ all possible threadings  
$ub \leftarrow$ upper bound for $X$

use a max priority queue with keys being the upper bounds  
ENQUEUE($Q$, ($ub$, $X$))

while (true)
do ($ub$, $X$) $\leftarrow$ DEQUEUE($Q$)
if $|X| = 1$ [the only remaining threading in the set]
then return $X$
else split($X$)
for each new subset $X_i$ from $X$
do $ub_i$ $\leftarrow$ upper bound for $X_i$
ENQUEUE($Q$, ($ub_i$, $X_i$))