Vorlesung Bioinformatik
Protein Threading

Dr. Axel Mosig

18. Mai 2004

Vorhersage der Tertiärstruktur von Proteinen

- *Ab Initio*-Methoden (z.B. via Molecular Dynamics) sind rechenintensiv; nur in Einzelfällen praktikabel.

- Gewisse Probleme in Zusammenhang mit ab-initio-Verfahren zur Strukturvorhersage von Proteinen sind NP-hart

- Homologe Primärstrukturen haben oft ähnliche Tertiärstrukturen

- Anzahl signifikant unterschiedlicher Tertiärstrukturen ist wesentlich geringer als Anzahl verschiedener Aminosäuresequenzen.

Vorhersage der Tertiärstruktur von Proteinen

- Schätzungsweise 650–10000 verschiedene Tertiärstrukturen ggü. weitaus mehr verschiedenen Primärstrukturen

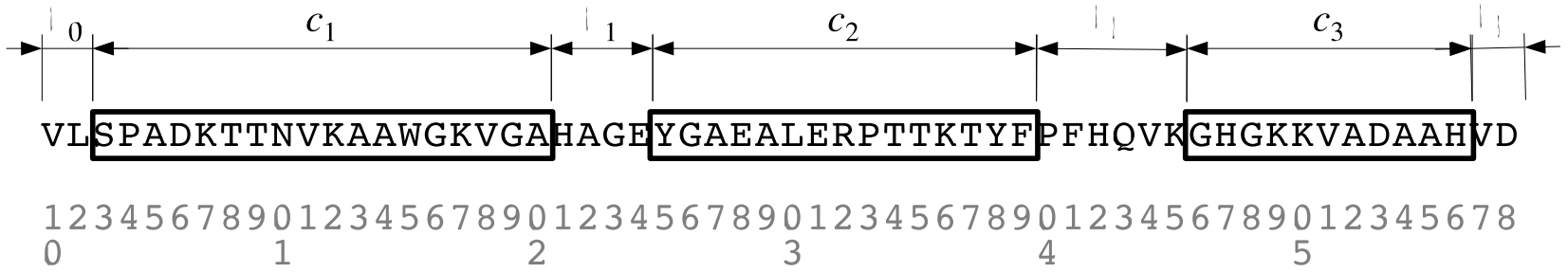
- $\leadsto$ auch Sequenzen mit nicht-offensichtlicher Sequenzähnlichkeit nehmen die gleiche Tertiärstruktur an.

- *Idee des Threadings:* verwende bekannte Sekundär- und Tertiärstruktur, um eine Primärstruktur mit unbekannter Faltung in diese Tertiärstruktur "einzufädeln".

- *Hier und heute:* Branch-and-Bound-Algorithmus von Lathrop und Smith (1996).

Threading-Modelle

- *Idee:* Essentiell für die Tertiärstruktur sind die Sequenzteile, die α-helices oder β-strands kodieren;
- Übergänge zwischen diesen sind weniger relevant für die Tertiärsrtuktur.
- Sekundärstruktur einer Sequenz s mit m Komponenten (α-Helices, β-Strands) als abstraktes Modell:



Threading-Modelle

- Länge der Übergänge zwischen den Sequenzteilen (also die λ_i) unterliegen gewissen Bedingungen:

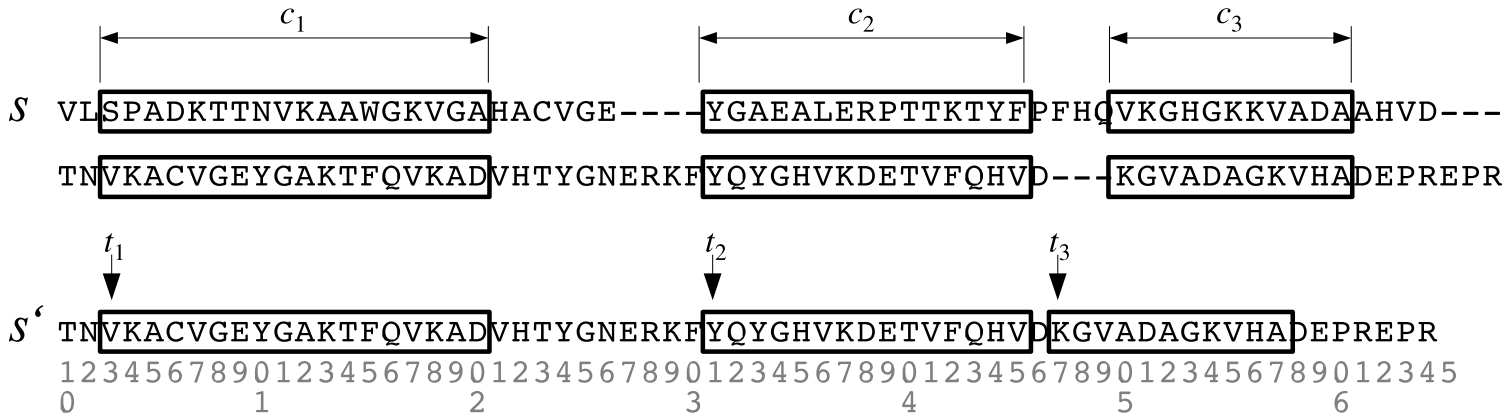
$$\ell_i \leq \lambda_i \leq L_i.$$

Definition 1. *Ein* Core Modell M *ist ein 5-Tupel* $M = (m, c, \lambda, \ell, L)$, *wobei*

- $m \equiv$ Anzahl Sek.-Strukt.-Elemente
- $c = (c_1, \ldots, c_m) \equiv$ Längen der Segmente
- $\lambda = (\lambda_0, \ldots, \lambda_m) \equiv$ Länge der Übergänge
- $\ell = (\ell_0, \ldots, \ell_m) \equiv$ Untere Schranken f. Überg.-Längen
- $L = (L_0, \ldots, L_m) \equiv$ Obere Schranken f. Überg.-Längen

Einfädeln einer Sequenz in ein Modell

- Struktur s mit Modell M; fädele nun s' in M ein.
- Ziel des Einfädelns : Sek.-Strukt.-Elemente werden Teilsequenzen gleicher Länge in s' zugeordnet; Längen der Übergänge dürfen variieren.
- Threading repräsentierbar als Folge $t_1, \ldots, t_m$



Formale Definition eines Threadings

Definition 2. *Sei s' Sequenz der Länge n' und M ein Core-Modell. Eine Folge $t = (t_1, \ldots, t_m)$ heißt* Threading *von s' durch M, wenn*

(T1) $1 + \ell_0 \leq t_1 \leq 1 + L_0$

(T2) $t_i + c_i + \ell_i \leq t_{i+1} \leq t_i + c_i + L_i$ *für* $1 < i < m$ *und*

(T3) $t_m + c_m + \ell_m \leq n' + 1 \leq t_m + c_m + L_m$

- Zu einem Modell M und einer Seq. s gibt es i.A. zahlreiche Threadings, die (T1)–(T3) erfüllen.
- Welches davon ist das beste? $\rightsquigarrow$ Scoring-Funktion

Scoring-Funtkionen: Struktur

- Scoring-Funktion f aus zwei Bestandteilen:
  - Wie gut "passt" ein Abschnitt von s' in ein Segment C_i?
    $\rightsquigarrow g_1(i, t_i)$
  - Wie gut interagieren Segmentpaare C_i und C_j?
    $\rightsquigarrow g_2(i, j, t_i, t_j)$
- Erweiterbar auf Segment-Tripel usw. durch $g_3(i, j, k, t_i, t_j, t_k)$ ...
- g_1, g_2 beruhen auf wissensbasierten Ansätzen
- g_2 z.B. via pairwise potentials $\rightsquigarrow$ Sippl (1990/1995)

Scoring-Funktionen: Interaktionsgraphen

- Segmente C_i und C_j aus einem Modell M interagieren nicht
  $\rightsquigarrow g_2(i, j, k, k') = 0$ für alle k, k'

- *Interaktionsgraph*: Graph G_I mit Knoten $V_I = \{1, \ldots, m\}$ und Knoten

$$E_I = \{(i, j) \mid \exists k, k' \colon g_2(i, j, k, k') \neq 0\}.$$

- Scoring-Funktion zu $t = (t_1, \ldots, t_m)$ formal:

$$f(t) = \sum_{i \in [1:m]} g_1(i, t_i) + \sum_{(i,j) \in E_I} g_2(i, j, t_i, t_j)$$

Threading als Optimierungsproblem

- Gegeben Core-Modell M zu Sequenz s sowie eine Sequenz s' mit unbekannter Tert.-Struktur

- Gesucht ist $\min_t f(t)$

- Berechnen von $\min_t f(t)$ ist (MAX-S)NP-hart: Akutsu/Miyano (1999)
  $\leadsto$ Backtracking-Algorithmus ("brute-force")
  $\leadsto$ Branch-and-Bound-Algorithmus von Lathrop und Smith (1996)

- Wenn g_2 ausser Betracht gelassen wird, gibt es Polynomialzeit-Algorithmen (via dyn. Progr.)

Relatives Threading

- *Ziel:* "Adressiere" alle Möglichen Threadings $T_M(s')$ einer Sequenz s' in ein Modell M, um $T_M(s')$ systematisch zu durchlaufen.

- $t = (t_1, \ldots, t_m)$ sei Threading von s' durch M.

- Das *relative Threading* $t' = (t'_1, \ldots, t'_m)$ zu t ist definiert durch

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

Branch-and-Bound-Algorithmen

- *Branch-and-Bound:* Geläufige Technik aus der kombinatorischen Optimierung, um die Laufzeit von Backtracking-Algorithmen zu verbessern; (siehe z.B. Buch von Papadimitriou/Steglitz)

- *Grundidee:*
  - Während der Berechnung einer optimalen Lösung ist die bisher beste Lösung $f(t_{\text{opt}}) = x_{\text{opt}}$ bekannt
  - es gibt gewisse Lösungsbereiche $T' \subset T_M(s')$, von denen sich schnell erkennen lässt, dass dort keine bessere Lösung $t' \in T$ mit $f(t') < f(t_{\text{opt}})$ liegen kann.

Gerüst für B-&-B-Algorithmen

```
branch-and-bound(X)
    S.push(X);
    x_opt := ∞;
    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 mit Branch-and-Bound

- Branch-and-Bound-Algorithmus traversiert einen Spannbaum von Lösungsmengen
- Durch leicht zu ermittelnde untere Schranken müssen manche Verzweigungen nicht weiter verfolgt werden.
- Wir benötigen:
  - Möglichkeit, Mengen von Threadings zu definieren, die sich leicht in mehrere Teile zerlegen lassen
  - Untere Schranken für Mengen von Threadings, die sich leicht ausrechnen lassen.

Threading-Mengen

- Definiere Intervalle $[b_i : d_i]$ (für $1 \leq i \leq m$)
- $\rightsquigarrow$ Vektoren $b = (b_1, \ldots, b_m)$ und $d = (d_1, \ldots, d_m)$.
- Erhalten so die Menge

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

  von (relativen) Threadings.

- $T_M(\mathbf{1}, \mathbf{n}') = T_M(s')$

Aufsplitten von Threading-Mengen ("Branch")

- Wähle i, so dass $b_i < d_i$.
- Teile Intervall $[b_i : d_i]$ auf in $[b_i : v]$ und $[v + 1 : d_i]$
- Definiere analog Vektoren b', d' und b'', d''
- $T_L := T_M(b', d')$ und $T_R := T_M(b'', d'')$ liefern die Aufteilung von $T_M(b, d)$.

Untere Schranken für Threading-Mengen ("Bound")

- *Gesucht:* Untere Schranke $B_M(b,d)$ mit Eigenschaften
  - $B_M(b,d) \leq \min_{t' \in T_M(b,d)} f(t')$
  - $B_M(b,d)$ soll einfach zu berechnen sein

- Wähle

$$B(b,d) := \sum_i \big((\min_{x \in [b_i:d_i]} g_1'(i,x) + \sum_{j<i} \min_{x,y} g_2(i,j,x,y) \big)$$

B-&-B-Threading-Algorithmus

```
thread(s, M)
    S.push(1, n);
    x_opt := ∞;
    while (!S.empty())
        (b, d) + S.pop();
        if (B(b, d) < x_opt) then
            if (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 (T_M(b', d') ≠ ∅) then
                    S.push((b', d'));
                if (T_M(b'', d'') ≠ ∅) then
                    S.push((b'', d''));
end.
```

Wie Komplex ist Protein-Threading?

- B-&-B-Algorithmus ist zwar schneller als naives Bachtrching, aber immer noch exponentielle Laufzeit im worst-case
- Threading-Problem ist MAX-SNP-vollständig
- *Bedeutet:* auch das Berechnen approximativer Lösungen geht nicht in poly.-Zeit, sofern P$\neq$NP!
- Wie "komplex" ist der Interaktionsgraph?
- Diverse erfolgreiche Strukturvorhersagen erzielt. (CASP)

Literatur

- Akutsu T, Miyano S, On the approximation of protein threading *Theoretical Computer Science* **210, 261-275** (1999)

- Lathrop, R.H. and Smith, T.F. Global Optimum Protein Threading with Gapped Alignment and Empirical Pair Potentials, *J. Mol. Biol.*, **255**, pp. 641-665, Feb., 1996.

- Sippl, M.J., Knowledge-Based potentials for Proteins. *Curr. Op. Struct. Biology*, **5**, 1995, pp.229–235

- P. Clote, R. Backofen, Computational Molecular Biology: An Introduction. John Wiley and Sons.