Protein Threading

BMI/CS 776
www.biostat.wisc.edu/bmi776/
Spring 2017
Anthony Gitter
gitter@biostat.wisc.edu
Goals for Lecture

Key concepts

• threading prediction task
• threading search task
• template models
• branch and bound search for threading
Protein Threading

• Generalization of homology modeling
  – homology modeling: align sequence to sequence
  – threading: align sequence to *structure* (templates)

• Key ideas
  – limited number of basic folds found in nature
  – amino acid preferences for different structural environments provide sufficient information to choose among folds
A Core Template

protein A threaded on template

core secondary structure segments

loops

protein B threaded on template

Figure from R. Lathrop et al. Analysis and Algorithms for Protein Sequence-Structure Alignment, in Computational Methods in Molecular Biology, Salzberg et al. editors, 1998.
Components of a Threading Approach

• Library of core fold templates
• Objective function to evaluate any particular placement of a sequence in a core template
• Method for searching over space of alignments between sequence and each core template
• Method for choosing the best template given alignments
Task Definition: Prediction Via Threading

• Given:
  – a protein sequence
  – a library of core templates

• Return: the best alignment of the sequence to a template
Task Definition: Threading Search

• Given:
  – a protein sequence
  – a single template

Return: the best alignment of the sequence to the template
Threading Objective Functions

• Possible sequence-template alignments are scored using a specified objective function

• Objective function scores the sequence-structure compatibility between
  – sequence amino acids
  – their corresponding positions in a core template

• It takes into account factors such as
  – amino acid preferences for solvent accessibility
  – amino acid preferences for particular secondary structures
  – interactions among spatially neighboring amino acids
Core Template with Interactions

Figure from R. Lathrop et al. Analysis and Algorithms for Protein Sequence-Structure Alignment.

- Small circles represent amino acid positions
- Thin lines indicate interactions represented in model

first amino acid in L interacts with last amino acid in L
last amino acid in K interacts with first amino acid in L
A threading can be represented as a vector $\vec{t}$, where each element indicates the index of the amino acid placed in the first position of each core segment.
A set of potential threadings can be represented by bounds on the first position of each core segment.

\[
T = \left\{ \vec{t} \mid b_i \leq t_i \leq d_i, \ b_j \leq t_j \leq d_j, \ b_k \leq t_k \leq d_k, \ b_l \leq t_l \leq d_l \right\}
\]
Possible Threadings

- Finding the optimal alignment is NP-hard in the general case where
  - there are variable length gaps between the core segments, and
  - the objective function includes interactions between neighboring (in 3D) amino acids

Figure from R. Lathrop et al. Analysis and Algorithms for Protein Sequence-Structure Alignment.
A General Pairwise Objective Function

• General objective function with pairwise interactions is:

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

scores for individual segments

scores for segment interactions

• We wish to minimize this function
Searching the Space of Alignments

• If interaction terms between amino acids are not allowed
  – dynamic programming
    • will find optimal alignment efficiently

• If interaction terms allowed
  – heuristic methods
    • fast
    • might not find the optimal alignment
  – exact methods (e.g. branch & bound)
    • if they return an alignment, it will be optimal
    • might take exponential time
    • might fail due to time or space limits
Branch and Bound Abstractly

• 3 components
  – A data structure for compactly representing a set of potential solutions
    • May be a very large set
  – An algorithm for computing a lower bound on the score obtained by any member of a set
    • In general, should not explicitly examine all members of the set
  – An algorithm for splitting a set into subsets
Branch and Bound Search

initialize $Q$ with one entry representing the set of all threadings
repeat
    $l \leftarrow$ set in $Q$ with lowest lower bound
    if $l$ contains only 1 threading
        return $l$
    else
        split $l$ into smaller subsets
        compute lower bound for each subset
        put subsets in $Q$ sorted by lower bound
Branch and Bound Illustrated

- A hypothetical branch and bound search
  - each circle illustrates the space of possible threadings
  - solid lines indicate splits made in previous steps
  - dashed lines indicate splits made in current step
  - numbers indicate lower bounds for newly created subsets
  - arrows show the set that has been split

Branch and Bound Search

• Two key issues
  – how to compute the lower bound for a set of threadings
  – how to split a threading set into subsets

• These determine the expected efficiency
A Simple Lower Bound

\[
\min_{\tilde{t} \in T} f(\tilde{t}) = \min_{\tilde{t} \in T} \sum_i \left[ g_1(i, t_i) + \sum_{j > i} g_2(i, j, t_i, t_j) \right]
\]

\[
\geq \sum_i \min_{b_i \leq x \leq d_i} g_1(i, x) + \sum_{j > i} \min_{b_i \leq y \leq d_i, b_j \leq z \leq d_j} g_2(i, j, y, z)
\]

- Calculate minimum over each term separately
- Can choose different starting positions for the same segment (e.g. x and y for segment \(i\))
A Better Lower Bound

$$\min_{\vec{t} \in T} f(\vec{t}) \geq$$

$$\min_{\vec{t} \in T} \sum_i \left[ g_1(i, t_i) + g_2(i - 1, i, t_{i-1}, t_i) + \min_{\vec{u} \in T} \sum_{j>i+1}^{j<i-1} \frac{1}{2} g_2(i, j, t_i, u_j) \right]$$

- Lower bound is closer approximation to actual score
- Constrain more of the starting indices to be shared instead of choosing them all independently
- Same $t_i$ appears in multiple terms
Splitting a Threading Set

• A threading set is split by choosing
  – a single core segment
  – a split point \( s_i \) in the segment

• A simple method
  – split the segment having the widest interval,
    i.e. \( \max_i [d_i - b_i] \)

  – choose the split point \( s_i \) as the value that results in
    the lower bound for the set
Branch and Bound: Splitting a Set

Figure from R. Lathrop et al. Analysis and Algorithms for Protein Sequence-Structure Alignment.

\[
T = \left\{ \tilde{t} \mid b_i \leq t_i \leq d_i, \ b_j \leq t_j \leq d_j, \ b_k \leq t_k \leq d_k, \ b_l \leq t_l \leq d_l \right\}
\]

1. \( T = \left\{ \tilde{t} \mid b_i \leq t_i < s_i, \ \cdots \right\} \)
2. \( T = \left\{ \tilde{t} \mid t_i = s_i, \ \cdots \right\} \)
3. \( T = \left\{ \tilde{t} \mid s_i < t_i \leq d_i, \ \cdots \right\} \)

choose segment

split into three sets
Threading Example

Suppose we have three segments \((i, j, k)\), each of which includes three amino acids. For a given sequence there are three possible starting positions for each segment. Suppose that you are given the following values for the scores of the individual segments and the scores for segment interactions.

\[
\begin{align*}
g_1(i,2) &= 5 & g_1(j,8) &= 9 & g_1(k,13) &= 3 \\
g_1(i,3) &= 2 & g_1(j,9) &= 7 & g_1(k,14) &= 4 \\
g_1(i,4) &= 8 & g_1(j,10) &= 6 & g_1(k,15) &= 1 \\
\end{align*}
\]

\[
\begin{align*}
g_2(i,j,2,8) &= 1 & g_2(j,k,8,13) &= 7 & g_2(i,k,2,13) &= 1 \\
g_2(i,j,2,9) &= 2 & g_2(j,k,8,14) &= 8 & g_2(i,k,2,14) &= 2 \\
g_2(i,j,2,10) &= 2 & g_2(j,k,8,15) &= 7 & g_2(i,k,2,15) &= 5 \\
g_2(i,j,3,8) &= 5 & g_2(j,k,9,13) &= 1 & g_2(i,k,3,13) &= 5 \\
g_2(i,j,3,9) &= 6 & g_2(j,k,9,14) &= 6 & g_2(i,k,3,14) &= 6 \\
g_2(i,j,3,10) &= 4 & g_2(j,k,9,15) &= 8 & g_2(i,k,3,15) &= 4 \\
g_2(i,j,4,8) &= 7 & g_2(j,k,10,13) &= 11 & g_2(i,k,4,13) &= 1 \\
g_2(i,j,4,9) &= 3 & g_2(j,k,10,14) &= 12 & g_2(i,k,4,14) &= 2 \\
g_2(i,j,4,10) &= 4 & g_2(j,k,10,15) &= 13 & g_2(i,k,4,15) &= 4 \\
\end{align*}
\]

We’ll find the optimal threading using the "simple lower bound" and splitting a set on the segment with the minimal \(g_1\) value. When splitting the selected segment, we’ll divide it into three intervals of length one.
Threading Example

T=[2,4], [8,10], [13,15]

split segment k

T=[2,4], [8,10], [13]
LB = g1(i,3) + g2(i,j,2,8) + g2(i,k,2,13) +
g1(j,10) + g2(j,k,9,13) +
g1(k,13)
= 14

T=[2,4], [8,10], [14]
LB = g1(i,3) + g2(i,j,2,8) + g2(i,k,2,14) +
g1(j,10) + g2(j,k,9,14) +
g1(k,14)
= 21

T=[2,4], [8,10], [15]
LB = g1(i,3) + g2(i,j,2,8) + g2(i,k,3,15) +
g1(j,10) + g2(j,k,8,15) +
g1(k,15)
= 21

split segment i

T=[2], [8,10], [13]
LB = g1(i,2) + g2(i,j,2,8) + g2(i,k,2,13) +
g1(j,8) + g2(j,k,8,13) +
g1(k,13)
= 26

T=[2], [9], [13]
LB = g1(i,2) + g2(i,j,2,9) + g2(i,k,2,13) +
g1(j,9) + g2(j,k,9,13) +
g1(k,13)
= 19

T=[2], [10], [13]
LB = g1(i,2) + g2(i,j,2,10) + g2(i,k,2,13) +
g1(j,10) + g2(j,k,10,13) +
g1(k,13)
= 28

split segment j

optimal threading
Branch and Bound Efficiency

- 58 proteins threaded against their “native” (i.e. correct) models

<table>
<thead>
<tr>
<th>Protein number</th>
<th>PDB code</th>
<th>Protein length</th>
<th>Number of core segments</th>
<th>Search Space Size</th>
<th>Number of search iterations</th>
<th>Total (search-only) seconds</th>
<th>Equivalent threadings per iteration</th>
<th>Equivalent threadings per second</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>256b</td>
<td>106</td>
<td>5</td>
<td>6.19e + 3</td>
<td>6</td>
<td>1 (1)</td>
<td>1.03e + 3</td>
<td>6.19e + 3</td>
</tr>
<tr>
<td>2</td>
<td>1end</td>
<td>137</td>
<td>3</td>
<td>4.79e + 4</td>
<td>6</td>
<td>1 (1)</td>
<td>7.98e + 3</td>
<td>4.79e + 4</td>
</tr>
<tr>
<td>3</td>
<td>1rcb</td>
<td>129</td>
<td>4</td>
<td>5.89e + 4</td>
<td>7</td>
<td>1 (1)</td>
<td>8.41e + 3</td>
<td>5.89e + 4</td>
</tr>
<tr>
<td>4</td>
<td>2mhr</td>
<td>118</td>
<td>4</td>
<td>9.14e + 4</td>
<td>7</td>
<td>1 (1)</td>
<td>1.31e + 4</td>
<td>9.14e + 4</td>
</tr>
<tr>
<td>5</td>
<td>351c</td>
<td>82</td>
<td>4</td>
<td>1.12e + 5</td>
<td>5</td>
<td>1 (1)</td>
<td>2.24e + 4</td>
<td>1.12e + 5</td>
</tr>
<tr>
<td>6</td>
<td>1brc</td>
<td>174</td>
<td>4</td>
<td>1.63e + 5</td>
<td>6</td>
<td>1 (1)</td>
<td>2.72e + 4</td>
<td>1.63e + 5</td>
</tr>
<tr>
<td>7</td>
<td>1ubq</td>
<td>76</td>
<td>5</td>
<td>1.70e + 5</td>
<td>6</td>
<td>1 (1)</td>
<td>2.83e + 4</td>
<td>1.70e + 5</td>
</tr>
<tr>
<td>8</td>
<td>1mbd</td>
<td>153</td>
<td>8</td>
<td>1.77e + 5</td>
<td>10</td>
<td>1 (1)</td>
<td>1.77e + 4</td>
<td>1.77e + 5</td>
</tr>
<tr>
<td>9</td>
<td>1lis</td>
<td>136</td>
<td>5</td>
<td>5.02e + 5</td>
<td>7</td>
<td>1 (1)</td>
<td>7.17e + 4</td>
<td>5.02e + 5</td>
</tr>
<tr>
<td>10</td>
<td>1aepl</td>
<td>161</td>
<td>5</td>
<td>5.76e + 5</td>
<td>13</td>
<td>1 (1)</td>
<td>4.43e + 4</td>
<td>5.78e + 5</td>
</tr>
<tr>
<td>50</td>
<td>5tmn</td>
<td>316</td>
<td>14</td>
<td>6.51e + 18</td>
<td>164</td>
<td>28 (7)</td>
<td>3.97e + 16</td>
<td>2.32e + 17</td>
</tr>
<tr>
<td>51</td>
<td>1lecb</td>
<td>242</td>
<td>15</td>
<td>7.01e + 18</td>
<td>320</td>
<td>26 (12)</td>
<td>2.19e + 16</td>
<td>2.70e + 17</td>
</tr>
<tr>
<td>52</td>
<td>1nar</td>
<td>290</td>
<td>17</td>
<td>2.33e + 19</td>
<td>3984</td>
<td>208 (183)</td>
<td>5.85e + 15</td>
<td>1.12e + 17</td>
</tr>
<tr>
<td>53</td>
<td>1s0l</td>
<td>275</td>
<td>15</td>
<td>4.36e + 19</td>
<td>541</td>
<td>32 (13)</td>
<td>8.05e + 16</td>
<td>1.36e + 18</td>
</tr>
<tr>
<td>54</td>
<td>5cpa</td>
<td>307</td>
<td>16</td>
<td>1.22e + 20</td>
<td>1089</td>
<td>72 (50)</td>
<td>1.12e + 17</td>
<td>1.69e + 18</td>
</tr>
<tr>
<td>55</td>
<td>9api</td>
<td>384</td>
<td>17</td>
<td>1.95e + 22</td>
<td>290</td>
<td>57 (25)</td>
<td>6.71e + 19</td>
<td>3.41e + 20</td>
</tr>
<tr>
<td>56</td>
<td>2had</td>
<td>310</td>
<td>19</td>
<td>2.57e + 22</td>
<td>4027</td>
<td>201 (179)</td>
<td>6.39e + 18</td>
<td>1.28e + 20</td>
</tr>
<tr>
<td>57</td>
<td>2cpp</td>
<td>414</td>
<td>20</td>
<td>6.37e + 24</td>
<td>3068</td>
<td>205 (164)</td>
<td>2.08e + 21</td>
<td>3.11e + 22</td>
</tr>
<tr>
<td>58</td>
<td>6taa</td>
<td>478</td>
<td>23</td>
<td>9.63e + 31</td>
<td>4917</td>
<td>1409 (1267)</td>
<td>1.96e + 28</td>
<td>6.83e + 28</td>
</tr>
</tbody>
</table>