Protein Structure Alignment

Ramji Srinivasan

ramji@cs.stanford.edu
Papers


Motivation
Motivation

• Evolution conserves protein structure significantly more than protein sequence
Motivation

• Evolution conserves protein structure significantly more than protein sequence

• Structural similarity often implies functional similarity
Motivation

- Structural similarity often implies functional similarity
Motivation
Motivation
Motivation
Protein A
Motivation

Protein A

Protein B
Assume we know the structure of two proteins, A and B, and we know the function of A.
Motivation

- Assume we know the structure of two proteins, A and B, and we know the function of A
- If we can produce an alignment between A and B, we might be able to infer B’s function
Assume we know the structure of two proteins, A and B, and we know the function of A.

If we can produce an alignment between A and B, we might be able to infer B’s function.
Motivation

Assume we know the structure of two proteins, A and B, and we know the function of A.

If we can produce an alignment between A and B, we might be able to infer B’s function.

Structural Alignment may imply functional similarity.
Problem

• Take 2 protein structures in R3, A and B, and output a pair of maximal substructures--one from each protein--that have the highest degree of similarity

• Like protein sequence alignment, just harder ;)

Preliminaries and Definitions
Preliminaries and Definitions

- Each protein is a chain of atoms in $\mathbb{R}^3$
- Assume one atom per residue to simplify the model
- Let $A$ be a protein of $n$ atoms, so $A = (a_1, \ldots, a_n)$ with $a_i \in \mathbb{R}^3$
- Define a $k$-long subchain $P = (p_1, p_2, \ldots, p_k)$ where $1 \leq p_1 \leq p_2 \leq \cdots \leq p_n$, by $A(P) = (a_{p_1}, a_{p_2}, \ldots, a_{p_k})$
- A gap is two consecutive indices, $p_i, p_{i+1}$, such that $p_i + 1 < p_{i+1}$
Preliminaries and Definitions

- Each protein is a chain of atoms in $\mathbb{R}^3$
- Assume one atom per residue to simplify the model
- Let $A$ be a protein of $n$ atoms, so $A = (a_1, \ldots, a_n)$ with $a_i \in \mathbb{R}^3$
- Define a $k$-long subchain $P = (p_1, p_2, \ldots, p_k)$ where $1 \leq p_1 \leq p_2 \leq \cdots \leq p_n$, by $A(P) = (a_{p_1}, a_{p_2}, \ldots, a_{p_k})$
- A gap is two consecutive indices, $p_i, p_{i+1}$, such that $p_i + 1 < p_{i+1}$
Preliminaries and Definitions

- Each protein is a chain of atoms in $\mathbb{R}^3$
- Assume one atom per residue to simplify the model
- Let $A$ be a protein of $n$ atoms, so $A = (a_1, \ldots, a_n)$ with $a_i \in \mathbb{R}^3$
- Define a $k$-long subchain $P = (p_1, p_2, \ldots, p_k)$ where $1 \leq p_1 \leq p_2 \leq \cdots \leq p_n$, by $A(P) = (a_{p_1}, a_{p_2}, \ldots, a_{p_k})$
- A gap is two consecutive indices, $p_i, p_{i+1}$, such that $p_i + 1 < p_{i+1}$

Say $P = (1, 2, 3, 5, 6)$

Protein A
Preliminaries and Definitions

- Each protein is a chain of atoms in $\mathbb{R}^3$
- Assume one atom per residue to simplify the model
- Let $A$ be a protein of $n$ atoms, so $A = (a_1, \ldots, a_n)$ with $a_i \in \mathbb{R}^3$
- Define a $k$-long subchain $P = (p_1, p_2, \ldots, p_k)$ where $1 \leq p_1 \leq p_2 \leq \cdots \leq p_n$, by $A(P) = (a_{p_1}, a_{p_2}, \ldots, a_{p_k})$
- A gap is two consecutive indices, $p_i, p_{i+1}$, such that $p_i + 1 < p_{i+1}$

```
Protein A

Say P = (1, 2, 3, 5, 6)

Subchain P
```
Preliminaries and Definitions

- Each protein is a chain of atoms in $\mathbb{R}^3$.
- Assume one atom per residue to simplify the model.
- Let $A$ be a protein of $n$ atoms, so $A = (a_1, \ldots, a_n)$ with $a_i \in \mathbb{R}^3$.
- Define a $k$-long subchain $P = (p_1, p_2, \ldots, p_k)$ where $1 \leq p_1 \leq p_2 \leq \cdots \leq p_n$, by $A(P) = (a_{p_1}, a_{p_2}, \ldots, a_{p_k})$.
- A gap is two consecutive indices, $p_i, p_{i+1}$, such that $p_i + 1 < p_{i+1}$.
Preliminaries and Definitions

• Consider 2 proteins, A of n atoms and B of m atoms, with 2 subchains, P of protein A and Q of protein B, with \( n \geq m \)

• **Correspondence:** two subchains P and Q of equal length \(|P| = |Q|\); a correspondence associates pairs of atoms from two proteins that appear in the same position in their respective subchains

• Number of gaps in a correspondence denoted by \( G_{P,Q} \)

• We keep A fixed, and apply a **rigid** transformation to B
Preliminaries and Definitions

- Consider 2 proteins, A of n atoms and B of m atoms, with 2 subchains, P of protein A and Q of protein B, with \( n \geq m \)

- **Correspondence**: two subchains P and Q of equal length \(|P| = |Q|\); a correspondence associates pairs of atoms from two proteins that appear in the same position in their respective subchains

- Number of gaps in a correspondence denoted by \( G_{P,Q} \)

- We keep A fixed, and apply a **rigid** transformation to B

![Diagram of Protein A with atoms labeled a1 to a8]
Preliminaries and Definitions

- Consider 2 proteins, A of n atoms and B of m atoms, with 2 subchains, P of protein A and Q of protein B, with \( n \geq m \)
- **Correspondence:** two subchains P and Q of equal length \(|P| = |Q|\); a correspondence associates pairs of atoms from two proteins that appear in the same position in their respective subchains
- Number of gaps in a correspondence denoted by \( G_{P,Q} \)
- We keep A fixed, and apply a **rigid** transformation to B

![Diagram of Protein A and Protein B with subchains and correspondence](image-url)
Preliminaries and Definitions

• Consider 2 proteins, A of n atoms and B of m atoms, with 2 subchains, P of protein A and Q of protein B, with \( n \geq m \)

• **Correspondence:** two subchains P and Q of equal length \(|P| = |Q|\); a correspondence associates pairs of atoms from two proteins that appear in the same position in their respective subchains

• Number of gaps in a correspondence denoted by \( G_{P,Q} \)

• We keep A fixed, and apply a **rigid** transformation to B
Preliminaries and Definitions

- Consider 2 proteins, A of n atoms and B of m atoms, with 2 subchains, P of protein A and Q of protein B, with \( n \geq m \)
- **Correspondence:** two subchains P and Q of equal length \(|P| = |Q|\); a correspondence associates pairs of atoms from two proteins that appear in the same position in their respective subchains
- Number of gaps in a correspondence denoted by \( G_{P,Q} \)
- We keep A fixed, and apply a **rigid** transformation to B

![Diagram](https://example.com/diagram.png)
Preliminaries and Definitions

- Consider 2 proteins, A of n atoms and B of m atoms, with 2 subchains, P of protein A and Q of protein B, with $n \geq m$

- **Correspondence**: two subchains P and Q of equal length $|P| = |Q|$; a correspondence associates pairs of atoms from two proteins that appear in the same position in their respective subchains

- Number of gaps in a correspondence denoted by $G_{P,Q}$

- We keep A fixed, and apply a rigid transformation to B

![Diagram of proteins and subchains](image)
Preliminaries and Definitions

- Consider 2 proteins, A of n atoms and B of m atoms, with 2 subchains, P of protein A and Q of protein B, with \( n \geq m \)
- **Correspondence:** two subchains P and Q of equal length \(|P| = |Q|\); a correspondence associates pairs of atoms from two proteins that appear in the same position in their respective subchains
- Number of gaps in a correspondence denoted by \( G_{P,Q} \)
- We keep A fixed, and apply a **rigid** transformation to B
Preliminaries and Definitions

- Consider 2 proteins, A of n atoms and B of m atoms, with 2 subchains, P of protein A and Q of protein B, with $n \geq m$
- **Correspondence:** two subchains P and Q of equal length $|P| = |Q|$; a correspondence associates pairs of atoms from two proteins that appear in the same position in their respective subchains
- Number of gaps in a correspondence denoted by $G_{P,Q}$
- We keep A fixed, and apply a **rigid** transformation to B
Consider 2 proteins, A of n atoms and B of m atoms, with 2 subchains, P of protein A and Q of protein B, with $n \geq m$.

**Correspondence:** two subchains P and Q of equal length $|P| = |Q|$; a correspondence associates pairs of atoms from two proteins that appear in the same position in their respective subchains.

Number of gaps in a correspondence denoted by $G_{P,Q}$.

We keep A fixed, and apply a **rigid** transformation to B.

Translation
Preliminaries and Definitions

• Consider 2 proteins, A of $n$ atoms and B of $m$ atoms, with 2 subchains, P of protein A and Q of protein B, with $n \geq m$

• **Correspondence:** two subchains P and Q of equal length $|P| = |Q|$; a correspondence associates pairs of atoms from two proteins that appear in the same position in their respective subchains

• Number of gaps in a correspondence denoted by $G_{P,Q}$

• We keep A fixed, and apply a **rigid** transformation to B
Preliminaries and Definitions

- Consider 2 proteins, A of \( n \) atoms and B of \( m \) atoms, with 2 subchains, P of protein A and Q of protein B, with \( n \geq m \)
- **Correspondence:** two subchains P and Q of equal length \( |P| = |Q| \); a correspondence associates pairs of atoms from two proteins that appear in the same position in their respective subchains
- Number of gaps in a correspondence denoted by \( G_{P,Q} \)
- We keep A fixed, and apply a **rigid** transformation to B
Preliminaries and Definitions

- Consider 2 proteins, A of n atoms and B of m atoms, with 2 subchains, P of protein A and Q of protein B, with \( n \geq m \)
- **Correspondence:** two subchains P and Q of equal length \( |P| = |Q| \); a correspondence associates pairs of atoms from two proteins that appear in the same position in their respective subchains
- Number of gaps in a correspondence denoted by \( G_{P,Q} \)
- We keep A fixed, and apply a **rigid** transformation to B

![Diagram](image)
• Consider 2 proteins, A of n atoms and B of m atoms, with 2 subchains, P of protein A and Q of protein B, with \( n \geq m \)

• **Correspondence:** two subchains P and Q of equal length \(|P| = |Q|\); a correspondence associates pairs of atoms from two proteins that appear in the same position in their respective subchains

• Number of gaps in a correspondence denoted by \( G_{P,Q} \)

• We keep A fixed, and apply a **rigid** transformation to B

\[ \text{Protein A} \]
\[ \text{Protein B} \]

\[ \text{Rotation} \]
Preliminaries and Definitions

- Consider 2 proteins, A of n atoms and B of m atoms, with 2 subchains, P of protein A and Q of protein B, with \( n \geq m \)
- **Correspondence:** two subchains P and Q of equal length \( |P| = |Q| \); a correspondence associates pairs of atoms from two proteins that appear in the same position in their respective subchains
- Number of gaps in a correspondence denoted by \( G_{P,Q} \)
- We keep A fixed, and apply a **rigid** transformation to B
Structural Alignment Problem

- Take two proteins, A and B, and find subchains P and Q of equal length such that
  1. A(P) and B(Q) are similar
  2. Correspondence length |P| = |Q| is maximal under condition 1
Measures of Similarity

- **Distance root mean squared**
  
  \[ dRMS = \left[ \frac{2}{k^2 - k} \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} (\|a_{p_i} - a_{p_j}\| - \|b_{q_i} - b_{q_j}\|)^2 \right]^{\frac{1}{2}} \]

- **Coordinate root mean squared**
  
  \[ cRMS = \min_{\hat{B}} \left[ \frac{1}{k} \sum_{i=1}^{k} (\|a_{p_i} - \hat{b}_{q_i}\|)^2 \right]^{\frac{1}{2}} \]

- **Structural score - penalizes gaps**
  
  \[ STRUCTURAL \ \text{score}_{P,Q} = \max_{\hat{B}} \sum_{i=1}^{k} \frac{20}{1 + \|a_{p_i} - \hat{b}_{q_i}\|^2 / 5} - 10 \ast G_{P,Q} \]
Conditions for Polynomial Time
Conditions for Polynomial Time

- A polynomial time algorithm works on only a “few” transformations, and for each transformation, aligns to the other protein quickly
Conditions for Polynomial Time

- A polynomial time algorithm works on only a “few” transformations, and for each transformation, aligns to the other protein quickly
  
  1. Number of rigid transformations under consideration is bounded by a polynomial
Conditions for Polynomial Time

• A polynomial time algorithm works on only a "few" transformations, and for each transformation, aligns to the other protein quickly

1. Number of rigid transformations under consideration is bounded by a polynomial

2. Given a rigid transformation, should be able to find a correspondence in polynomial time
Conditions for Polynomial Time

- A polynomial time algorithm works on only a “few” transformations, and for each transformation, aligns to the other protein quickly
  
  1. Number of rigid transformations under consideration is bounded by a polynomial
  
  2. Given a rigid transformation, should be able to find a correspondence in polynomial time

Protein B

\[
\begin{align*}
  &b_2 \quad b_4 \\
  &b_3 \quad b_5 \\
  &b_1 \quad b_6
\end{align*}
\]
Conditions for Polynomial Time

- A polynomial time algorithm works on only a “few” transformations, and for each transformation, aligns to the other protein quickly

1. Number of rigid transformations under consideration is bounded by a polynomial

2. Given a rigid transformation, should be able to find a correspondence in polynomial time

Protein B

Searching across only a “few” transformations, and each transformation is aligned quickly
Conditions for Polynomial Time

- A polynomial time algorithm works on only a “few” transformations, and for each transformation, aligns to the other protein quickly.

1. Number of rigid transformations under consideration is bounded by a polynomial

2. Given a rigid transformation, should be able to find a correspondence in polynomial time

Searching across only a “few” transformations, and each transformation is aligned quickly.
Conditions for Polynomial Time

- A polynomial time algorithm works on only a “few” transformations, and for each transformation, aligns to the other protein quickly.

1. Number of rigid transformations under consideration is bounded by a polynomial.

2. Given a rigid transformation, should be able to find a correspondence in polynomial time.

Searching across only a “few” transformations, and each transformation is aligned quickly.
Conditions for Polynomial Time

- A polynomial time algorithm works on only a “few” transformations, and for each transformation, aligns to the other protein quickly

1. Number of rigid transformations under consideration is bounded by a polynomial

2. Given a rigid transformation, should be able to find a correspondence in polynomial time

Protein B

Searching across only a “few” transformations, and each transformation is aligned quickly
Do the easy part first: find a correspondence
Do the easy part first: find a correspondence

- Say we’ve got a rigid transformation of B, now what?
Do the easy part first: find a correspondence

- Say we’ve got a rigid transformation of B, now what?
- We can find the best correspondence in polynomial time as long as our scoring function is separable:
Do the easy part first: find a correspondence

- Say we’ve got a rigid transformation of B, now what?
- We can find the best correspondence in polynomial time as long as our scoring function is separable:
  1. Optimal substructure: restriction of an optimal correspondence to any substructure is itself an optimal correspondence of the substructure
Do the easy part first: find a correspondence

- Say we’ve got a rigid transformation of B, now what?
- We can find the best correspondence in polynomial time as long as our scoring function is separable:
  1. Optimal substructure: restriction of an optimal correspondence to any substructure is itself an optimal correspondence of the substructure
  2. Space of relevant subproblems is small (polynomial)
Do the easy part first: find a correspondence

- Say we’ve got a rigid transformation of B, now what?
- We can find the best correspondence in polynomial time as long as our scoring function is separable:
  1. Optimal substructure: restriction of an optimal correspondence to any substructure is itself an optimal correspondence of the substructure
  2. Space of relevant subproblems is small (polynomial)
- These two conditions make the problem amenable to dynamic programming, in $O(n^2)$ time and space
Bounding the Number of Transformations

- Still, we need to show that the number of rigid transformations must be polynomially bounded
- With certain conditions on the scoring function, that scoring function can be approximated by evaluating it only polynomially many times
Rigid Transformations

- Rigid transformation consists of a translation and rotation
- Translation and rotations parameterized by vectors
Rigid Transformations

- Rigid transformation consists of a translation and rotation

- Translation and rotations parameterized by vectors

\[
\text{Translation} \quad (t_x, t_y, t_z) \in \mathbb{R}^3
\]

\[
t_x \in \left[ \frac{- (X^A + X^B)}{2}, \frac{(X^A + X^B)}{2} \right]
\]

\[
t_y \in \left[ \frac{- (Y^A + Y^B)}{2}, \frac{(Y^A + Y^B)}{2} \right]
\]

\[
t_z \in \left[ \frac{- (Z^A + Z^B)}{2}, \frac{(Z^A + Z^B)}{2} \right]
\]
Rigid Transformations

- Rigid transformation consists of a translation and rotation
- Translation and rotations parameterized by vectors

Translation

\((t_x, t_y, t_z) \in \mathbb{R}^3\)

- \(t_x \in \left[ \frac{-(X^A + X^B)}{2}, \frac{(X^A + X^B)}{2} \right]\)
- \(t_y \in \left[ \frac{-(Y^A + Y^B)}{2}, \frac{(Y^A + Y^B)}{2} \right]\)
- \(t_z \in \left[ \frac{-(Z^A + Z^B)}{2}, \frac{(Z^A + Z^B)}{2} \right]\)

\(X_a = 8, X_b = 4\)

Range = ±6
Rigid Transformations

- Rigid transformation consists of a translation and rotation
- Translation and rotations parameterized by vectors

\[
\begin{align*}
 t_x &\in \left[ \frac{-X^A + X^B}{2}, \frac{X^A + X^B}{2} \right], \\
 t_y &\in \left[ \frac{-Y^A + Y^B}{2}, \frac{Y^A + Y^B}{2} \right], \\
 t_z &\in \left[ \frac{-Z^A + Z^B}{2}, \frac{Z^A + Z^B}{2} \right],
\end{align*}
\]

\[X_a = 8, X_b = 4\]

Range = ±6
Rigid Transformations

- Rigid transformation consists of a translation and rotation

- Translation and rotations parameterized by vectors

\[
(t_x, t_y, t_z) \in \mathbb{R}^3
\]

\[
t_x \in \left[ \frac{-(X^A + X^B)}{2}, \frac{(X^A + X^B)}{2} \right]
\]

\[
t_y \in \left[ \frac{-(Y^A + Y^B)}{2}, \frac{(Y^A + Y^B)}{2} \right]
\]

\[
t_z \in \left[ \frac{-(Z^A + Z^B)}{2}, \frac{(Z^A + Z^B)}{2} \right]
\]

- Range = ±6

Xa = 8, Xb = 4

\[
\begin{align*}
Xa & = 8, & Xb & = 4 \\
\text{Range} & = \pm 6
\end{align*}
\]
Rigid Transformations

- Rigid transformation consists of a translation and rotation
- Translation and rotations parameterized by vectors

Translation

\[(t_x, t_y, t_z) \in \mathbb{R}^3\]

\[
t_x \in \left[\frac{-(X^A + X^B)}{2}, \frac{(X^A + X^B)}{2}\right]
\]

\[
t_y \in \left[\frac{-(Y^A + Y^B)}{2}, \frac{(Y^A + Y^B)}{2}\right]
\]

\[
t_z \in \left[\frac{-(Z^A + Z^B)}{2}, \frac{(Z^A + Z^B)}{2}\right]
\]

\[
X_a = 8, X_b = 4
\]

Range = ±6
Rigid Transformations

- Rigid transformation consists of a translation and rotation

- Translation and rotations parameterized by vectors

\[
\begin{align*}
    t_x &\in \left[ \frac{- (X^A + X^B)}{2}, \frac{(X^A + X^B)}{2} \right] \\
    t_y &\in \left[ \frac{- (Y^A + Y^B)}{2}, \frac{(Y^A + Y^B)}{2} \right] \\
    t_z &\in \left[ \frac{- (Z^A + Z^B)}{2}, \frac{(Z^A + Z^B)}{2} \right]
\end{align*}
\]

Range: \(\pm 6\)

\(X_a = 8, X_b = 4\)
Rigid Transformations

• Rigid transformation consists of a translation and rotation

• Translation and rotations parameterized by vectors

Translation

\[(t_x, t_y, t_z) \in \mathbb{R}^3\]

\[t_x \in \left[ \frac{-(X^A + X^B)}{2}, \frac{(X^A + X^B)}{2} \right] \]

\[t_y \in \left[ \frac{-(Y^A + Y^B)}{2}, \frac{(Y^A + Y^B)}{2} \right] \]

\[t_z \in \left[ \frac{-(Z^A + Z^B)}{2}, \frac{(Z^A + Z^B)}{2} \right] \]

\[X_a = 8, X_b = 4\]

Range = ±6
Rigid Transformations

- Rigid transformation consists of a translation and rotation

- Translation and rotations parameterized by vectors

Translation
\[
(t_x, t_y, t_z) \in \mathbb{R}^3
\]

\[
t_x \in \left[ -\frac{(X^A + X^B)}{2}, \frac{(X^A + X^B)}{2} \right]
\]

\[
t_y \in \left[ -\frac{(Y^A + Y^B)}{2}, \frac{(Y^A + Y^B)}{2} \right]
\]

\[
t_z \in \left[ -\frac{(Z^A + Z^B)}{2}, \frac{(Z^A + Z^B)}{2} \right]
\]

\(X_a = 8, X_b = 4\)

Range = ±6
Rigid Transformations

- Rigid transformation consists of a translation and rotation

- Translation and rotations parameterized by vectors

\[
(t_x, t_y, t_z) \in \mathbb{R}^3
\]

\[
t_x \in \left[ \frac{-(X^A + X^B)}{2}, \frac{(X^A + X^B)}{2} \right]
\]

\[
t_y \in \left[ \frac{-(Y^A + Y^B)}{2}, \frac{(Y^A + Y^B)}{2} \right]
\]

\[
t_z \in \left[ \frac{-(Z^A + Z^B)}{2}, \frac{(Z^A + Z^B)}{2} \right]
\]

\[
X_a = 8, X_b = 4
\]

Range = ±6
Rigid Transformations

- Rigid transformation consists of a translation and rotation

- Translation and rotations parameterized by vectors

Translation

\[(t_x, t_y, t_z) \in \mathbb{R}^3\]

\[t_x \in \left[ \frac{-(X^A + X^B)}{2}, \frac{(X^A + X^B)}{2} \right]\]

\[t_y \in \left[ \frac{-(Y^A + Y^B)}{2}, \frac{(Y^A + Y^B)}{2} \right]\]

\[t_z \in \left[ \frac{-(Z^A + Z^B)}{2}, \frac{(Z^A + Z^B)}{2} \right]\]

\[X_a = 8, X_b = 4\]

Range = ±6
Rigid Transformations

- Rigid transformation consists of a translation and rotation
- Translation and rotations parameterized by vectors

Translation

\[(t_x, t_y, t_z) \in \mathbb{R}^3\]

\[t_x \in \left[\frac{- (X^A + X^B)}{2}, \frac{(X^A + X^B)}{2}\right]\]

\[t_y \in \left[\frac{- (Y^A + Y^B)}{2}, \frac{(Y^A + Y^B)}{2}\right]\]

\[t_z \in \left[\frac{- (Z^A + Z^B)}{2}, \frac{(Z^A + Z^B)}{2}\right]\]
Rigid Transformations

- Rigid transformation consists of a translation and rotation
- Translation and rotations parameterized by vectors

Translation

\[(t_x, t_y, t_z) \in \mathbb{R}^3\]

\[
t_x \in \left[\frac{-(X^A + X^B)}{2}, \frac{(X^A + X^B)}{2}\right]
\]
\[
t_y \in \left[\frac{-(Y^A + Y^B)}{2}, \frac{(Y^A + Y^B)}{2}\right]
\]
\[
t_z \in \left[\frac{-(Z^A + Z^B)}{2}, \frac{(Z^A + Z^B)}{2}\right]
\]

Rotation

\[(r_1, r_2, r_3) \text{ with } r_i \in [0, 2\pi]\]
CDS Function

- Assume a correspondence between subchain P of A and Q of B; with A fixed

- A CDS is a Correspondence Dependent Scoring Function

- For each rigid transformation of B, compute the CDS function, denoted by $Sc_{P,Q}$, by using the distances between corresponding atom pairs in space,

- There are exponentially many correspondences and exponentially many CDS functions; we only care about scoring functions of the form:

$$F(r_1, r_2, r_3, t_x, t_y, t_z) = \max_{|P|=|Q|} Sc_{P,Q}(r_1, r_2, r_3, t_x, t_y, t_z)$$
CDS Function

- Assume a correspondence between subchain P of A and Q of B; with A fixed
- A CDS is a Correspondence Dependent Scoring Function
- For each rigid transformation of B, compute the CDS function, denoted by $Sc_{PQ}$, by using the distances between corresponding atom pairs in space,
- There are exponentially many correspondences and exponentially many CDS functions; we only care about scoring functions of the form:

$$F(r_1, r_2, r_3, t_x, t_y, t_z) = \max_{|P|=|Q|} Sc_{P,Q}(r_1, r_2, r_3, t_x, t_y, t_z)$$
CDS Function

- Assume a correspondence between subchain P of A and Q of B; with A fixed
- A CDS is a Correspondence Dependent Scoring Function
- For each rigid transformation of B, compute the CDS function, denoted by $Sc_{PQ}$, by using the distances between corresponding atom pairs in space,
- There are exponentially many correspondences and exponentially many CDS functions; we only care about scoring functions of the form:

$$F(r_1, r_2, r_3, t_x, t_y, t_z) = \max_{|P|=|Q|} Sc_{PQ}(r_1, r_2, r_3, t_x, t_y, t_z)$$
CDS Function

- Assume a correspondence between subchain P of A and Q of B; with A fixed.
- A CDS is a Correspondence Dependent Scoring Function.
- For each rigid transformation of B, compute the CDS function, denoted by $S_{C_P}$, by using the distances between corresponding atom pairs in space.
- There are exponentially many correspondences and exponentially many CDS functions; we only care about scoring functions of the form:

$$F(r_1, r_2, r_3, t_x, t_y, t_z) = \max_{|P|=|Q|} S_{C_P, \mathbf{Q}}(r_1, r_2, r_3, t_x, t_y, t_z)$$
CDS Function

- Assume a correspondence between subchain P of A and Q of B; with A fixed
- A CDS is a Correspondence Dependent Scoring Function
- For each rigid transformation of B, compute the CDS function, denoted by $S_{PQ}$, by using the distances between corresponding atom pairs in space,
- There are exponentially many correspondences and exponentially many CDS functions; we only care about scoring functions of the form:

$$F(r_1, r_2, r_3, t_x, t_y, t_z) = \max_{|P|=|Q|} S_{PQ}(r_1, r_2, r_3, t_x, t_y, t_z)$$
Lipschitz conditions for the CDS function

- We’re going to impose some conditions on our CDS function that allow us to compute that same function at only a “few” points to find a near-optimal value.

- A CDS function $S_c$ satisfies coordinate-wise Lipschitz conditions with values $c_r$ and $c_t$, if:

  For all rigid transformations $\vec{p} = (r_1, r_2, r_3, t_x, t_y, t_z)$ and for all $\delta > 0$

  \[
  |S_{c_P,Q}(\vec{p} + \delta \vec{e}_r) - S_{c_P,Q}(\vec{p})| \leq c_r \delta \\
  |S_{c_P,Q}(\vec{p} + \delta \vec{e}_t) - S_{c_P,Q}(\vec{p})| \leq c_t \delta
  \]

where $\vec{e}_1, \ldots, \vec{e}_6$ are the standard basis vectors in $\mathbb{R}^6$, $\vec{e}_r \in \{\vec{e}_1, \vec{e}_2, \vec{e}_3\}$, and $\vec{e}_t \in \{\vec{e}_4, \vec{e}_5, \vec{e}_6\}$.
Lipschitz conditions for the CDS function

- We’re going to impose some conditions on our CDS function that allow us to compute that same function at only a “few” points to find a near-optimal value.

- A CDS function $S_c$ satisfies coordinate-wise Lipschitz conditions with values $c_r$ and $c_t$, if:

  For all rigid transformations $\vec{p} = (r_1, r_2, r_3, t_x, t_y, t_z)$ and for all $\delta > 0$

  $$|S_{c_P,Q}(\vec{p} + \delta \vec{e}_r) - S_{c_P,Q}(\vec{p})| \leq c_r \delta$$

  $$|S_{c_P,Q}(\vec{p} + \delta \vec{e}_t) - S_{c_P,Q}(\vec{p})| \leq c_t \delta$$

where $\vec{e}_1, \ldots, \vec{e}_6$ are the standard basis vectors in $\mathbb{R}^6$, $\vec{e}_r \in \{\vec{e}_1, \vec{e}_2, \vec{e}_3\}$, and $\vec{e}_t \in \{\vec{e}_4, \vec{e}_5, \vec{e}_6\}$.
Lipschitz conditions for the CDS function

- We’re going to impose some conditions on our CDS function that allow us to compute that same function at only a “few” points to find a near-optimal value.

- A CDS function $S_c$ satisfies coordinate-wise Lipschitz conditions with values $c_r$ and $c_t$, if:

  For all rigid transformations $\vec{p} = (r_1, r_2, r_3, t_x, t_y, t_z)$ and for all $\delta > 0$

  \[
  |S_{c_P,Q}(\vec{p} + \delta \vec{e}_r) - S_{c_P,Q}(\vec{p})| \leq c_r \delta
  \]

  \[
  |S_{c_P,Q}(\vec{p} + \delta \vec{e}_t) - S_{c_P,Q}(\vec{p})| \leq c_t \delta
  \]

  where $\vec{e}_1, \ldots, \vec{e}_6$ are the standard basis vectors in $\mathbb{R}^6$, $\vec{e}_r \in \{\vec{e}_1, \vec{e}_2, \vec{e}_3\}$, and $\vec{e}_t \in \{\vec{e}_4, \vec{e}_5, \vec{e}_6\}$
Lipschitz conditions for the CDS function

• We’re going to impose some conditions on our CDS function that allow us to compute that same function at only a “few” points to find a near-optimal value

• A CDS function $S_c$ satisfies coordinate-wise Lipschitz conditions with values $c_r$ and $c_t$, if:

For all rigid transformations $\vec{p} = (r_1, r_2, r_3, t_x, t_y, t_z)$ and for all $\delta > 0$

$$|S_{c_P, Q}(\vec{p} + \delta \vec{e}_r) - S_{c_P, Q}(\vec{p})| \leq c_r \delta$$

$$|S_{c_P, Q}(\vec{p} + \delta \vec{e}_t) - S_{c_P, Q}(\vec{p})| \leq c_t \delta$$

where $\vec{e}_1, \ldots, \vec{e}_6$ are the standard basis vectors in $\mathbb{R}^6$, $\vec{e}_r \in \{\vec{e}_1, \vec{e}_2, \vec{e}_3\}$, and $\vec{e}_t \in \{\vec{e}_4, \vec{e}_5, \vec{e}_6\}$

Small perturbations in the transformation affect the scoring function by a user-defined constant
Lipschitz-satisfying CDS functions yield finite sets

- You give me a CDS function that satisfies Lipschitz conditions for some $c_r$ and $c_t$ that you specify

- I’ll give you a finite set of rotations and translations that you need to evaluate your function on to get a good alignment score

- For every $\epsilon > 0$, there exists a finite set $G = G(\epsilon)$ of rotations and translations such that

  1. $|G| = O\left( \frac{n c_r^3 c_t^3}{\epsilon^6} \right)$

  2. For every choice of a translation and a rotation $\vec{p}$, there is a point $\vec{p}_G \in G$ with $|F(\vec{p}) - F(\vec{p}_G)| \leq \epsilon$. 
Lipschitz-satisfying CDS functions yield finite sets

- You give me a CDS function that satisfies Lipschitz conditions for some $c_r$ and $c_t$ that you specify
- I’ll give you a finite set of rotations and translations that you need to evaluate your function on to get a good alignment score
- For every $\epsilon > 0$, there exists a finite set $G = G(\epsilon)$ of rotations and translations such that
  1. $|G| = O\left(\frac{nc^3c^3}{\epsilon^6}\right)$
  2. For every choice of a translation and a rotation $\vec{p}$, there is a point $\vec{p}_G \in G$ with $|F(\vec{p}) - F(\vec{p}_G)| \leq \epsilon$. 

\[
\text{Diagram:}
\begin{align*}
  F(\vec{p}) \quad &\quad \vec{p} \\
\end{align*}
\]
Lipschitz-satisfying CDS functions yield finite sets

• You give me a CDS function that satisfies Lipschitz conditions for some $c_r$ and $c_t$ that you specify

• I’ll give you a finite set of rotations and translations that you need to evaluate your function on to get a good alignment score

• For every $\epsilon > 0$, there exists a finite set $G = G(\epsilon)$ of rotations and translations such that

1. $|G| = O\left(\frac{n c_r^3 c_t^3}{\epsilon^6}\right)$

2. For every choice of a translation and a rotation $\vec{p}$, there is a point $\vec{p}_G \in G$ with $|F(\vec{p}) - F(\vec{p}_G)| \leq \epsilon$. 
Lipschitz-satisfying CDS functions yield finite sets

- You give me a CDS function that satisfies Lipschitz conditions for some $c_r$ and $c_t$ that you specify.

- I’ll give you a finite set of rotations and translations that you need to evaluate your function on to get a good alignment score.

- For every $\epsilon > 0$, there exists a finite set $G = G(\epsilon)$ of rotations and translations such that
  
  1. $|G| = O\left(\frac{nc^3c^3}{\epsilon^6}\right)$
  
  2. For every choice of a translation and a rotation $\vec{p}$, there is a point $\vec{p}_G \in G$ with $|F(\vec{p}) - F(\vec{p}_G)| \leq \epsilon$.

I can find a transformation in $G$ whose $F$ score is within $\epsilon$ of the $F$ score of the transformation you specified.
Constructing a $G(\epsilon)$ set, in rotations

- Given an $\epsilon$, in each dimension of rotation ($x,y, and z$) space out points such that:

$$\delta_r = \frac{\epsilon}{3c_r}$$

- Size of each set in each dimension of rotation is $2\pi / (\epsilon / 3c_r)$ - so the size of the set in the rotation dimension is $3 * 2\pi * 3c_r / \epsilon$, or $O(c_r / \epsilon)$
Constructing a $G(\epsilon)$ set, in rotations

- Given an $\epsilon$, in each dimension of rotation ($x,y,\text{ and } z$) space out points such that:

$$\delta_r = \frac{\epsilon}{3c_r}$$

- Size of each set in each dimension of rotation is $2\pi / (\epsilon / 3cr)$ - so the size of the set in the rotation dimension is $3 \times 2\pi \times 3cr / \epsilon$, or $O(cr / \epsilon)$
Constructing a $\mathcal{G}(\varepsilon)$ set, in rotations

- Given an $\varepsilon$, in each dimension of rotation ($x,y,$ and $z$) space out points such that:

$$\delta_r = \frac{\varepsilon}{3c_r}$$

- Size of each set in each dimension of rotation is $2\pi / (\varepsilon / 3c_r)$ - so the size of the set in the rotation dimension is $3 \times 2\pi \times 3c_r / \varepsilon$, or $O(c_r / \varepsilon)$
Constructing a $G(\epsilon)$ set, in rotations

- Given an $\epsilon$, in each dimension of rotation (x, y, and z) space out points such that:
  \[ \delta_r = \frac{\epsilon}{3c_r} \]

- Size of each set in each dimension of rotation is $2\pi / (\epsilon / 3c_r)$ - so the size of the set in the rotation dimension is $3 \times 2\pi \times 3c_r / \epsilon$, or $O(cr / \epsilon)$
Constructing a \(G(\epsilon)\) set, in rotations

- Given an \(\epsilon\), in each dimension of rotation (x,y, and z) space out points such that:

\[
\delta_r = \frac{\epsilon}{3c_r}
\]

- Size of each set in each dimension of rotation is \(2\pi / (\epsilon / 3cr)\) - so the size of the set in the rotation dimension is \(3 * 2\pi * 3cr / \epsilon\), or \(O(cr / \epsilon)\)
Constructing a $G(\epsilon)$ set, in rotations

- Given an $\epsilon$, in each dimension of rotation ($x,y,$ and $z$) space out points such that:
  \[ \delta_r = \frac{\epsilon}{3c_r} \]

- Size of each set in each dimension of rotation is $2\pi / (\epsilon / 3c_r)$ - so the size of the set in the rotation dimension is $3 \times 2\pi \times 3c_r / \epsilon$, or $O(cr / \epsilon)$
Constructing a $\mathbb{G}(\varepsilon)$ set, in rotations

- Given an $\varepsilon$, in each dimension of rotation ($x$, $y$, and $z$) space out points such that:

  \[ \delta_r = \frac{\varepsilon}{3c_r} \]

- Size of each set in each dimension of rotation is $2\pi / (\varepsilon / 3c_r)$ - so the size of the set in the rotation dimension is $3 \times 2\pi \times 3c_r / \varepsilon$, or $O(cr / \varepsilon)$
Constructing a $G(\varepsilon)$ set, in translations

- Given an $\varepsilon$, in each dimension of translation (x,y, and z) space out points such that:

$$\delta_t = \frac{\varepsilon}{3c_t}$$

- Size of each set in each dimension of translation is:

$O[(W^A + W^B)c_t/\varepsilon]$, where $W = X, Y, Z$ respectively

\[\frac{\varepsilon}{3c_t}, \frac{2\varepsilon}{3c_t}, \frac{3\varepsilon}{3c_t}\text{ etc...}\]

\[-\frac{(X_a + X_b)}{2}, \frac{(X_a + X_b)}{2}\]
Size of the $G(\epsilon)$ set

- Given that the protein has $n$ residues, we have the total size of $G$:

$$|G| = \left( \frac{6\pi c_r^3}{\epsilon} \right)^3 \left( \frac{3c_t(W^A + W^B)}{\epsilon} \right)^3 = O\left( \frac{n c_r^3 c_t^3}{\epsilon^6} \right)$$
Size of the $G(\epsilon)$ set

- Given that the protein has $n$ residues, we have the total size of $G$:

$$|G| = \left( \frac{6\pi c_r^3}{\epsilon} \right)^3 \left( \frac{3c_t(W^A + W^B)}{\epsilon} \right)^3 = O\left( \frac{n c_r^3 c_t^3}{\epsilon^6} \right)$$

This gets bigger as $\epsilon$ gets smaller.
Points in $G(\varepsilon)$ are $\varepsilon$ close to any transformation
Points in $G(\varepsilon)$ are $\varepsilon$ close to any transformation

- Take a point $\hat{p}$ in rotation and translation space
- Nearest point $\hat{p}_G \in G$ can be reached by moving at most $\delta_r/2$ along each dimension of rotation and $\delta_t/2$ along each dimension of translation
Points in $G(\epsilon)$ are $\epsilon$ close to any transformation

- Take a point $\hat{p}$ in rotation and translation space
- Nearest point $\hat{p}_G \in G$ can be reached by moving at most $\delta_r/2$ along each dimension of rotation and $\delta_t/2$ along each dimension of translation
Points in $G(\epsilon)$ are $\epsilon$ close to any transformation

- Take a point $\hat{p}$ in rotation and translation space
- Nearest point $\hat{p}_G \in G$ can be reached by moving at most $\delta_r/2$ along each dimension of rotation and $\delta_t/2$ along each dimension of translation
Points in $G(\varepsilon)$ are $\varepsilon$ close to any transformation

- Take a point $\hat{p}$ in rotation and translation space
- Nearest point $\hat{p}_G \in G$ can be reached by moving at most $\delta_r/2$ along each dimension of rotation and $\delta_t/2$ along each dimension of translation
Points in $G(\epsilon)$ are $\epsilon$ close to any transformation
Points in $G(\epsilon)$ are $\epsilon$ close to any transformation

- Say we move $\delta_r/2$ in one axis of rotation space
- Then, because $|Sc_{P,Q}(\vec{p} + \delta\vec{e}_r) - Sc_{P,Q}(\vec{p})| \leq c_r\delta$ for any $\delta > 0$
- We have $|F_{P,Q}(\vec{p} + \delta_r/2 \cdot \vec{e}_r) - F_{P,Q}(\vec{p})| \leq c_r \cdot \delta_r/2$ for this particular rotation
- And so the overall change is at most $3c_r\delta_r/2 + 3c_r\delta_t/2 = \epsilon/2 + \epsilon/2 = \epsilon$
Finding all near-maximal scoring points of $F$

- Let $M$ be the global maximum of $F$, and call a point $\hat{p}$ $\epsilon$-maximal if $F(\hat{p}) \geq M - \epsilon$
- From our previous result, we know that there is a close point $\hat{p}_G \in G$ with $F(\hat{p}_G) \geq F(\hat{p}) - \epsilon$
- Given $\epsilon$ evaluate $F$ on all points of $G(\epsilon)$ defined above—recall that there are $O(nc^3c^3/\epsilon^6)$ such points
- Now take the subset of these points within $2\epsilon$ from the highest value found
- This will find approximations to all $\epsilon$-maximal points
Finding all near-maximal scoring points of F

- Let $M$ be the global maximum of $F$, and call a point $\hat{p}$ $\epsilon$-maximal if $F(\hat{p}) \geq M - \epsilon$

- From our previous result, we know that there is a close point $\hat{p}_G \in G$ with $F(\hat{p}_G) \geq F(\hat{p}) - \epsilon$

- Given $\epsilon$ evaluate $F$ on all points of $G(\epsilon)$ defined above—recall that there are $O(nc^3c^3/r^6)$ such points

- Now take the subset of these points within $2\epsilon$ from the highest value found

- This will find approximations to all $\epsilon$-maximal points
Finding all near-maximal scoring points of $F$

- Let $M$ be the global maximum of $F$, and call a point $\hat{p}$ $\epsilon$-maximal if $F(\hat{p}) \geq M - \epsilon$
- From our previous result, we know that there is a close point $\hat{p}_G \in G$ with $F(\hat{p}_G) \geq F(\hat{p}) - \epsilon$
- Given $\epsilon$ evaluate $F$ on all points of $G(\epsilon)$ defined above—recall that there are $O(nc_i^3c_r^3/\epsilon^6)$ such points
- Now take the subset of these points within $2\epsilon$ from the highest value found
- This will find approximations to all $\epsilon$-maximal points
Finding all near-maximal scoring points of $F$

- Let $M$ be the global maximum of $F$, and call a point $\hat{p}$ $\epsilon$-maximal if $F(\hat{p}) \geq M - \epsilon$
- From our previous result, we know that there is a close point $\hat{p}_G \in G$ with $F(\hat{p}_G) \geq F(\hat{p}) - \epsilon$
- Given $\epsilon$ evaluate $F$ on all points of $G(\epsilon)$ defined above—recall that there are $O(nc_1^3c_r^3/\epsilon^6)$ such points
- Now take the subset of these points within $2\epsilon$ from the highest value found
- This will find approximations to all $\epsilon$-maximal points
STRUCTAL-type scores are well-behaved

• All functions of the form:

\[ F(r_1, r_2, r_3, t_x, t_y, t_z) = \max_{|P|=|Q|} \sum_{i=1}^{|P|} \frac{C_1}{C_2 + \|a_{p_i} - b_{q_i}\|^2} - C_3 \cdot G_{P,Q} \]

where \( C_1, C_2, \) and \( C_3 \) are positive constants.

where \( C_1, C_2, \) and \( C_3 \) are positive constants, are well-behaved

• That is, these functions satisfy the Lipschitz conditions with

Globular Proteins \( \begin{cases} c_t = O(n) \\ c_r = O(n^{4/3}) \end{cases} \)
Nonglobular Proteins \( \begin{cases} c_r = O(n^2) \end{cases} \)
Closely related NP-hard problem

- Instead of a protein chain $A$ of $n$ atoms, we can create an $n$-by-$n$ real symmetric matrix $D$
- $D(i,j)$ is the Euclidean distance between $i$th and $j$th atoms of $A$
- Internal distance matrix that corresponds to a subchain $S$ of the protein $A$ is the submatrix of $D$ consisting of the rows and columns indexed by elements of $S$

$$D = \begin{bmatrix} D(a_1, a_1) & D(a_1, a_2) & D(a_1, a_3) \\ D(a_2, a_1) & D(a_2, a_2) & D(a_2, a_3) \\ D(a_3, a_1) & D(a_3, a_2) & D(a_3, a_3) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & \sqrt{2} \\ 1 & \sqrt{2} & 0 \end{bmatrix}.$$
Closely related NP-hard problem

- Instead of a protein chain $A$ of $n$ atoms, we can create an $n$-by-$n$ real symmetric matrix $D$

- $D(i,j)$ is the Euclidean distance between $i$th and $j$th atoms of $A$

- Internal distance matrix that corresponds to a subchain $S$ of the protein $A$ is the submatrix of $D$ consisting of the rows and columns indexed by elements of $S$

Protein $A$ with Euclidean Distances noted

\[
D = \begin{bmatrix}
D(a_1, a_1) & D(a_1, a_2) & D(a_1, a_3) \\
D(a_2, a_1) & D(a_2, a_2) & D(a_2, a_3) \\
D(a_3, a_1) & D(a_3, a_2) & D(a_3, a_3)
\end{bmatrix} = \begin{bmatrix}
0 & 1 & 1 \\
1 & 0 & \sqrt{2} \\
1 & \sqrt{2} & 0
\end{bmatrix}.
\]
Slightly generalized problem
Slightly generalized problem

• If we took generalized distance metrics, instead of Euclidean distances, the approximate structural-alignment problem would be NP-hard

• Any correct and efficient solution of the approximate structural-alignment problem exploits the fact that proteins are in 3d Euclidean space
Slightly generalized problem
Slightly generalized problem

- If you could solve the generalized distance alignment problem, you could solve CLIQUE
- CLIQUE is an NP-hard problem: the input is a graph and an integer $k$
- Output is either a $k$-clique, or, if there is no such graph “no”
Slightly generalized problem

• If you could solve the generalized distance alignment problem, you could solve CLIQUE

• CLIQUE is an NP-hard problem: the input is a graph and an integer $k$

• Output is either a $k$-clique, or, if there is no such graph “no”

If $k = 4$, there is a clique
Slightly generalized problem

- If you could solve the generalized distance alignment problem, you could solve CLIQUE.

- CLIQUE is an NP-hard problem: the input is a graph and an integer $k$.

- Output is either a $k$-clique, or, if there is no such graph "no".

If $k = 4$, there is a clique.
Slightly generalized problem

• If you could solve the generalized distance alignment problem, you could solve CLIQUE.

• CLIQUE is an NP-hard problem: the input is a graph and an integer $k$.

• Output is either a $k$-clique, or, if there is no such graph “no”.

If $k = 4$, there is a clique. If $k = 4$, there is no clique.
Slightly generalized problem

• If you could solve the generalized distance alignment problem, you could solve CLIQUE

• CLIQUE is an NP-hard problem: the input is a graph and an integer k

• Output is either a k-clique, or, if there is no such graph “no”

If $k = 4$, there is a clique

If $k = 4$, there is no clique
NP-hard Problem

Formulation

- Let $D^A$, $D^B$ be distance matrices of two metric spaces—effectively, the internal distance matrices of chains $A$ and $B$

- Let the score of two subchains $P$ and $Q$, of equal length $|P| = |Q| = k$, be
  \[ \text{Sc}_{P,Q} = \sum_{i=1}^{k} \sum_{j=1, i \neq j}^{k} 2/(1 + [D^A(p_i, p_j) - D^B(q_i, q_j)]^2) \]

- For every $0 < \epsilon < 1$, it is NP-hard to find subchains that are within $\epsilon$ from the optimal score
Create two chains and compare them

- We’ll take a graph $G = (V, E)$, with $|V| = n$, then construct two chain structures, and use the algorithm for finding correspondences of near-optimal scores.
- First structure, $S^A$ has $n$ atoms and encodes $G$—each vertex is associated with an atom (using some ordering), and the distance between two atoms is the length of a shortest path in $G$ between two corresponding vertices.
- Internal-distances matrix associated with $S^A$ is an $n \times n$ matrix $D^A$, where $D^A(i, j)$ is the length of the shortest path from $v_i$ to $v_j$.
- Second structure, $S^B$ has $k$ atoms—meaning it encodes a clique of size $k$.
- Internal-distances matrix, $D^B$, has zeroes on the diagonal and ones elsewhere.
- Now, if the score is strictly $> 2(k^2 - k) - 1$ return the subset of $S^A$ (a $k$-clique); otherwise, return “no”.
Create two chains and compare them

- We’ll take a graph \( G = (V, E) \), with \(|V| = n\), then construct two chain structures, and use the algorithm for finding correspondences of near-optimal scores
- First structure, \( S^A \) has \( n \) atoms and encodes \( G \)—each vertex is associated with an atom (using some ordering), and the distance between two atoms is the length of a shortest path in \( G \) between two corresponding vertices
- Internal-distances matrix associated with \( S^A \) is an \( n \times n \) matrix \( D^A \), where \( D^A(i, j) \) is the length of the shortest path from \( v_i \) to \( v_j \)
- Second structure, \( S^B \) has \( k \) atoms—meaning it encodes a clique of size \( k \)
- Internal-distances matrix, \( D^B \), has zeroes on the diagonal and ones elsewhere
- Now, if the score is strictly \( > 2(k^2 - k) - 1 \) return the subset of \( S^A \) (a \( k \)-clique); otherwise, return “no”
Create two chains and compare them

- We’ll take a graph $G = (V, E)$, with $|V| = n$, then construct two chain structures, and use the algorithm for finding correspondences of near-optimal scores
- First structure, $S^A$ has $n$ atoms and encodes $G$—each vertex is associated with an atom (using some ordering), and the distance between two atoms is the length of a shortest path in $G$ between two corresponding vertices
- Internal-distances matrix associated with $S^A$ is an $n \times n$ matrix $D^A$, where $D^A(i, j)$ is the length of the shortest path from $v_i$ to $v_j$
- Second structure, $S^B$ has $k$ atoms—meaning it encodes a clique of size $k$
- Internal-distances matrix, $D^B$, has zeroes on the diagonal and ones elsewhere
- Now, if the score is strictly $> 2(k^2 - k) - 1$ return the subset of $S^A$ (a $k$-clique); otherwise, return “no”

\[
D^A = \begin{bmatrix}
0 & 1 & 2 & 3 & 3 \\
1 & 0 & 1 & 2 & 2 \\
2 & 1 & 0 & 1 & 1 \\
3 & 2 & 1 & 0 & \sqrt{2} \\
3 & 1 & 1 & \sqrt{2} & 0 \\
\end{bmatrix}.
\]
Create two chains and compare them

- We’ll take a graph $G = (V, E)$, with $|V| = n$, then construct two chain structures, and use the algorithm for finding correspondences of near-optimal scores
- First structure, $S^A$ has $n$ atoms and encodes $G$—each vertex is associated with an atom (using some ordering), and the distance between two atoms is the length of a shortest path in $G$ between two corresponding vertices
- Internal-distances matrix associated with $S^A$ is an $n \times n$ matrix $D^A$, where $D^A(i, j)$ is the length of the shortest path from $v_i$ to $v_j$
- Second structure, $S^B$ has $k$ atoms—meaning it encodes a clique of size $k$
- Internal-distances matrix, $D^B$, has zeroes on the diagonal and ones elsewhere
- Now, if the score is strictly $> 2(k^2 - k) - 1$ return the subset of $S^A$ (a $k$-clique); otherwise, return “no”
Create two chains and compare them

- We’ll take a graph $G = (V, E)$, with $|V| = n$, then construct two chain structures, and use the algorithm for finding correspondences of near-optimal scores.
- First structure, $S^A$ has $n$ atoms and encodes $G$—each vertex is associated with an atom (using some ordering), and the distance between two atoms is the length of a shortest path in $G$ between two corresponding vertices.
- Internal-distances matrix associated with $S^A$ is an $n \times n$ matrix $D^A$, where $D^A(i, j)$ is the length of the shortest path from $v_i$ to $v_j$.

\[
D^A = \begin{pmatrix}
0 & 1 & 2 & 3 & 3 \\
1 & 0 & 1 & 2 & 2 \\
2 & 1 & 0 & 1 & 1 \\
3 & 2 & 1 & 0 & \sqrt{2} \\
3 & 1 & 1 & \sqrt{2} & 0
\end{pmatrix}.
\]

- Second structure, $S^B$ has $k$ atoms—meaning it encodes a clique of size $k$.
- Internal-distances matrix, $D^B$, has zeroes on the diagonal and ones elsewhere.
- Now, if the score is strictly $> 2(k^2 - k) - 1$ return the subset of $S^A$ (a $k$-clique); otherwise, return “no”.

Structure $B$ has a clique of $k$.

\[
D^B = \begin{pmatrix}
0 & 1 & 1 \\
1 & 0 & 1 \\
1 & 1 & 0
\end{pmatrix}.
\]
Create two chains and compare them

- We’ll take a graph $G = (V, E)$, with $|V| = n$, then construct two chain structures, and use the algorithm for finding correspondences of near-optimal scores.
- First structure, $S^A$ has $n$ atoms and encodes $G$—each vertex is associated with an atom (using some ordering), and the distance between two atoms is the length of a shortest path in $G$ between two corresponding vertices.
- Internal-distances matrix associated with $S^A$ is an $n \times n$ matrix $D^A$, where $D^A(i, j)$ is the length of the shortest path from $v_i$ to $v_j$.
- Second structure, $S^B$ has $k$ atoms—meaning it encodes a clique of size $k$.
- Internal-distances matrix, $D^B$, has zeroes on the diagonal and ones elsewhere.
- Now, if the score is strictly $> 2(k^2 - k) - 1$ return the subset of $S^A$ (a $k$-clique); otherwise, return “no”.

**Graph G**

$D^A = \begin{pmatrix}
0 & 1 & 2 & 3 & 3 \\
1 & 0 & 1 & 2 & 2 \\
2 & 1 & 0 & 1 & 1 \\
3 & 2 & 1 & 0 & \sqrt{2} \\
3 & 1 & 1 & \sqrt{2} & 0
\end{pmatrix}$.

**Structure B has a clique of $k$**

$D^B = \begin{pmatrix}
0 & 1 & 1 \\
1 & 0 & 1 \\
1 & 1 & 0
\end{pmatrix}$.

**Subchain P**

$P = \begin{pmatrix}
0 & 1 & 1 \\
1 & 0 & \sqrt{2} \\
1 & \sqrt{2} & 0
\end{pmatrix}$.
Create two chains and compare them

- We’ll take a graph $G = (V,E)$, with $|V| = n$, then construct two chain structures, and use the algorithm for finding correspondences of near-optimal scores
- First structure, $S^A$ has $n$ atoms and encodes $G$—each vertex is associated with an atom (using some ordering), and the distance between two atoms is the length of a shortest path in $G$ between two corresponding vertices
- Internal-distances matrix associated with $S^A$ is an $n \times n$ matrix $D^A$, where $D^A(i,j)$ is the length of the shortest path from $v_i$ to $v_j$
- Second structure, $S^B$ has $k$ atoms—meaning it encodes a clique of size $k$
- Internal-distances matrix, $D^B$, has zeroes on the diagonal and ones elsewhere
- Now, if the score is strictly $> 2(k^2 - k) - 1$ return the subset of $S^A$ (a $k$-clique); otherwise, return “no”

Graph $G$

\[
D^A = \begin{bmatrix}
0 & 1 & 2 & 3 & 3 \\
1 & 0 & 1 & 2 & 2 \\
2 & 1 & 0 & 1 & 1 \\
3 & 2 & 1 & 0 & \sqrt{2} \\
3 & 1 & 1 & \sqrt{2} & 0
\end{bmatrix}
\]

Structure B has a clique of $k$

\[
D^B = \begin{bmatrix}
0 & 1 & 1 \\
1 & 0 & 1 \\
1 & 1 & 0
\end{bmatrix}
\]

\[
Sc_{P,Q} = \sum_{i=1}^{k} \sum_{j=1, i \neq j}^{k} 2/(1 + [D^A(p_i, p_j) - D^B(q_i, q_j)]^2)
\]

Score for the subchain $P$ is $11.414 > 2(k^2 - k) - 1 = 11$
Results

- Let $R$ be a net for the space of rotations, $T$ a net for translations, and $R \times T$ a net for all rigid transformations.

- Factoid of the day: “Visualizing a function of six parameters is, of course, very hard.”

- Instead we define $ST^T(r_1, r_2, r_3) = \max_{(t_x, t_y, t_z) \in T}[\text{STRUCTAL}(r_1, r_2, r_3, t_x, t_y, t_z)]$

- Kolodny calculates the maximum over a heuristically-determined smaller set of translations, $T(ot)$, where $T(ot)$ is the set of translations that position an atom from protein $A$ exactly on top of an atom from protein $B$. 
Results
Results

• Using a quaternion representation, can represent a set of rotations as a sphere
• Can vary a parameter to produce spheres of different radii
• Then you can evaluate the alignment scoring function on each point of the surface of the sphere
Results

An “unrolled” sphere

Single alignment

Fig. 3. Example of a pair of structures with a single meaningful alignment. We plot the $ST_{10d}^2$ score for aligning 5rxn (54 residues) and 1brf (53 residues) over the space of rotations. These proteins have the same SCOP fold classification, rubredoxin-like, and each has three $\beta$-strands and three helices. The $ST_{10d}^2$ score function has a single maximum, implying one meaningful way of aligning the pair. The maximal score found is 993, aligning 53 residues to 0.797 Å crMS.

- Using a quaternion representation, can represent a set of rotations as a sphere
- Can vary a parameter to produce spheres of different radii
- Then you can evaluate the alignment scoring function on each point of the surface of the sphere
Results

An “unrolled” sphere

Single alignment

Fig. 3. Example of a pair of structures with a single meaningful alignment. We plot the ST^θφ score for aligning 5rxn (54 residues) and 1bfr (53 residues) over the space of rotations. These proteins have the same SCOP fold classification, rubredoxin-like, and each has three β-strands and three helices. The ST^θφ score function has a single maximum, implying one meaningful way of aligning the pair. The maximal score found is 993, aligning 53 residues to 0.797 Å cRMS.
Results

Fig. 3. Example of a pair of structures with a single meaningful alignment. We plot the $ST^{160}_2$ score for aligning 5r3n (54 residues) and 1bvf (53 residues) over the space of rotations. These proteins have the same SCOP fold classification, rubredoxin-like, and each has three $\beta$-strands and three helices. The $ST^{160}_2$ score function has a single maximum, implying one meaningful way of aligning the pair. The maximal score found is 993, aligning 53 residues to 0.797 Å cRMS.

Fig. 4. Example of a pair of structures with two meaningful alignments [this example was noted by Zu-Kang and Sippl (29)]. We plot the $ST^{160}_2$ score for aligning 1mjc (69 residues) and 1shf (59 residues). The two maxima can be seen clearly, as can additional, less significant maxima. One of the two best alignments scores 458, aligning 41 residues to 2.89 Å cRMS; the other scores 454, aligning 38 residues to 2.52 Å cRMS. These proteins are of the same SCOP class, all-$\beta$, and a different SCOP fold (OB and $\alpha$H3-like barrel, respectively).
Parameterized Algorithm for Protein Structure Alignment
Parameterized Algorithm for Protein Structure Alignment

- You give me two proteins
Parameterized Algorithm for Protein Structure Alignment

- You give me two proteins
- You tell me how densely packed the atoms of each protein are
Parameterized Algorithm for Protein Structure Alignment

• You give me two proteins
• You tell me how densely packed the atoms of each protein are
• You tell me how far away an atom from the first protein can be from the second protein in the final alignment
Parameterized Algorithm for Protein Structure Alignment

• You give me two proteins
• You tell me how densely packed the atoms of each protein are
• You tell me how far away an atom from the first protein can be from the second protein in the final alignment
• I’ll give you a solution within some bound of the optimal
Parameterized Algorithm for Protein Structure Alignment

- Models proteins using a contact map graph

- **Contact**: a pair of residues that are spatially close to each other

- **Contact Map Graph**: Consists of all residues (vertices) and contacts (edges), inferred from crystal structures

- Protein structural alignment becomes a maximum common subgraph problem
Challenges with Contact Map Modeling

• When two contact maps are aligned, doesn’t take into account geometric information in protein structure

• Xu et. al. created a parameterized algorithm for the problem when one protein structure is modeled as a contact map and the other by a contact map or distance matrix
What does the algorithm do?

• Generates a non-sequential alignment and corresponding rigid-body transformation such that:
What does the algorithm do?
What does the algorithm do?

- Alignment score $\geq (1 - 1/k) \cdot OPT(D_c)$
- Distance between two matched residues $\leq (1 + \epsilon) \cdot D_c$ after two proteins superimposed by rigid-body transformation
- Running time $= O(k^2 \cdot poly(n) \cdot 2^{twl\Delta} / (\epsilon \cdot D_c)^6)$

where

\[
D_c = \text{Max allowed distance between two matched residues after proteins are superimposed}
\]
\[
OPT(D_c) = \text{Optimal alignment score between two proteins}
\]
\[
poly(n) = \text{polynomial in size } n
\]
\[
tw = O(k^2 \cdot \max(2D_c, D_u)^3 / D_l^3)
\]
\[
\Delta = (1 + 2D_c/D_l)^3
\]
\[
D_u = \text{Distance threshold determining if two residues are in contact or not}
\]
\[
D_l = \text{Minimum inter-residue distance in a protein}
\]
\[
k = \text{positive integer}
\]
Preliminaries

• What’s a parameterized algorithm?
  • Approach to solving NP-hard problems
  • Time complexity is polynomial in problem size but exponential with respect to some user-defined parameters
  • If parameters are constant, can terminate within polynomial time
Contact Map Modeling

• Model a protein in 3d space as a graph \( G = (V,E) \)

• There is a contact edge \((i,j) \in E\) between residues \(i\) and \(j\) if and only if their spatial distance is less than \(D_u\),

• Let \(D_l\) be the minimum inter-residue distance in a protein

Then each residue is adjacent to at most \((1 + 2D_u/D_l)^3\) other residues
Contact Map Modeling

- Model a protein in 3d space as a graph $G = (V,E)$
- There is a contact edge $(i,j) \in E$ between residues $i$ and $j$ if and only if their spatial distance is less than $D_u$,
- Let $D_l$ be the minimum inter-residue distance in a protein

Then each residue is adjacent to at most $(1 + 2D_u/D_l)^3$ other residues

Take $D_u = 6, D_l = 2$
Contact Map Modeling

• Model a protein in 3d space as a graph $G = (V, E)$

• There is a contact edge $(i, j) \in E$ between residues $i$ and $j$ if and only if their spatial distance is less than $D_u$,

• Let $D_l$ be the minimum inter-residue distance in a protein

Then each residue is adjacent to at most $(1 + 2D_u/D_l)^3$ other residues

Take $D_u = 6$, $D_l = 2$
Contact Map Modeling

- Model a protein in 3d space as a graph $G = (V,E)$
- There is a contact edge $(i,j) \in E$ between residues $i$ and $j$ if and only if their spatial distance is less than $D_u$,
- Let $D_l$ be the minimum inter-residue distance in a protein

Then each residue is adjacent to at most $(1 + 2D_u/D_l)^3$ other residues

Take $D_u = 6$, $D_l = 2$

So a residue at $0$ is adjacent to $7$ residues (including itself)
What is an alignment?

- Given a protein chain $A$, let $G[A]$ denote its contact map graph.
- For a substructure $P$ of $A$, let $G[P]$ denote the contact map induced by $P$.
- Given $A$ and $B$, an alignment between $A$ and $B$ is a pair of substructures $P$ and $Q$ satisfying the following:
  - $P$ is a substructure of $A$ and $Q$ of $B$.
  - There is a one-to-one mapping between residues in $P$ and $Q$. Residue $p$ in $A$ is equivalent to residue $q$ in $B$ if and only if $p$ is mapped to $q$. One contact edge in $G[P]$ is equivalent to one in $G[Q]$ if and only if their two end points are equivalent.
- Optimal alignment is then just the alignment that maximizes the number of equivalent edges.
What’s the problem?
What’s the problem?

• Take two proteins, A and B, each represented by a contact graph
• Let E[A] and E[B] be the sets of contacts in proteins A and B
• For any residue u in A, let M(u) be the corresponding residue in B
• If there’s no corresponding residue in B for u, then M(u) = Φ
What’s the problem?
What’s the problem?

- Our goal, then is to maximize the objective function:
What’s the problem?

- Our goal, then is to maximize the objective function:

$$\sum_{u,v \in V[A], u < v} f(u, v, M(u), M(v)) \text{ where }$$

$$f(u, v, M(u), M(v)) = \begin{cases} -\infty & M(u) = M(v) \neq \phi \\ 1 & (u, v) \in E[A], (M(u), M(v)) \in E[B] \\ 0 & \text{otherwise} \end{cases}$$
Exact Protein Structure Alignment Algorithm
Exact Protein Structure Alignment Algorithm

• To figure out whether two residues in protein A align to the same residue in B or not extend the contact graph $G[A]$ to $G'[A] = (V[A], E'[A])$ by adding more edges to $G[A]$

• Add an extra edge $(u, v)$ to $G'[A]$ if the distance between $u$ and $v$ satisfies
Exact Protein Structure Alignment Algorithm

- To figure out whether two residues in protein A align to the same residue in B or not extend the contact graph $G[A]$ to $G'[A] = (V[A], E'[A])$ by adding more edges to $G[A]$
  - Add an extra edge $(u,v)$ to $G'[A]$ if the distance between $u$ and $v$ satisfies

$$D_u < ||u - v|| < 2 \cdot D_c$$
Exact Protein Structure Alignment Algorithm

- To figure out whether two residues in protein A align to the same residue in B or not extend the contact graph $G[A]$ to $G'[A] = (V[A], E'[A])$ by adding more edges to $G[A]$.

- Add an extra edge $(u, v)$ to $G'[A]$ if the distance between $u$ and $v$ satisfies

$$D_u < \|u - v\| < 2 \cdot D_c$$

- Then, for any two residues $u$ and $v$ in A, if there’s no edge between them in $G'[A]$, they can’t align to the same residue in B, because the maximum distance between two equivalent residues is no more than $D_c$. 

Exact Protein Structure Alignment Algorithm

\[ D_u < \|u - v\| < 2 \cdot D_c \]

- Then, for any two residues \( u \) and \( v \) in \( A \), if there’s no edge between them in in \( G'[A] \), they can’t align to the same residue in \( B \), because the maximum distance between two equivalent residues is no more than \( D_c \)
Then, for any two residues $u$ and $v$ in $A$, if there’s no edge between them in $G'[A]$, they can’t align to the same residue in $B$, because the maximum distance between two equivalent residues is no more than $D_c$. 

Exact Protein Structure Alignment Algorithm
Exact Protein Structure Alignment Algorithm
Exact Protein Structure Alignment Algorithm

Assume $||a - b|| = 7$, so greater than $2D_c = 6$  

Given:

Protein A $\quad a \quad\leftrightarrow\quad b$  

Protein B $\quad q$  

$D_c = 3$  

$D_u = 1$

Then both $a$ and $b$ can’t align to $q$, because $||a - b||$  

$\leq ||a - q|| + ||q - b||$,  

where $||a - b|| > 6$, but $||a - q|| < 3$ and $||q - b|| < 3$
Using this extended graph, we can revise the objective function as follows:

\[
\sum_{u,v \in E'[A], u<v} f(u, v, M(u), M(v)) \quad \text{where}
\]

\[
f(u, v, M(u), M(v)) = \begin{cases} 
-\infty & M(u) = M(v) \neq \phi \\
1 & (u, v) \in E[A], (M(u), M(v)) \in E[B] \\
0 & \text{otherwise}
\end{cases}
\]
Tree-Decomposition

Detour

• For computational purposes, we want to convert this graph, $G'[A]$, into a tree

• Once we have it in tree form, we can apply an existing tree-decomposition based algorithm to maximize the objective function
Let $G(V, E)$ be a graph. A tree decomposition of $G$ is a pair $(T, X)$ satisfying the following conditions:

1. $T = (I, F)$ is a tree with a node set $I$ and an edge set $F$
2. $X = X_i | i \in I, X_i \subseteq V$ and $\bigcup_{i \in I} X_i = V$ Each node of $T$ represents a subset of $V$, and the union of all subsets is just $V$
3. For every edge $e = v, w \in E$, there is at least one $i \in I$ such that both $v$ and $w$ are in $X_i$
4. For all $i, j, k \in I$, if $j$ is a node on the path from $i$ to $k$ in $T$, then $X_i \cap X_k \subseteq X_j$
5. Width of a tree decomposition given by $\max_{i \in I}(|X_i| - 1)$ and treewidth, $tw(G)$, of $G$ is the minimum width over all tree decompositions of $G$
Tree Decomposition

Example

Residue Interaction Graph

Tree Decomposition Graph

Treewidth = (4 - 1) = 3

This decomposition is not unique
Protein Side-Chain Packing

- Problem: Given the backbone coordinates of a protein, predict the coordinates of the side-chain atoms
Side-chain Packing

- Each residue has many possible side-chain positions.
- Each possible position is called a rotamer.
- Need to avoid atomic clashes.
Xu’s Prior Results

• We’ll take the following as known quantities

• Given a residue interaction graph, there is a low-degree polynomial time algorithm that can find a tree decomposition of treewidth:

\[ O(|V|^{(2/3) \lg |V|}) \]

• The tree decomposition based side-chain assignment algorithm has a computational complexity of

\[ O((|V| + |E|) n_{rot}^{O(|V|^{(2/3) \lg |V|})}) \]

• Instead of assigning rotamers to side chains, we’re going to be assigning residues
Tree Decomposition on $G'[A]$
Tree Decomposition on $G'[A]$

- Using Xu’s prior results, can prove that:

  \[
  \text{treewidth}(G'[A]) = O(\max(2D_c, D_u)/D_l \cdot n^{2/3} \cdot \lg n)
  \]

- Since the distance between two matched residues is no more than $D_c$, it follows that each residue in $A$ can be aligned to at most $O((1 + 2D_c/D_l)^3)$ residues in $B$
Tree Decomposition on $G'[A]$

- Using Xu’s prior results, can prove that:
  \[
  \text{treewidth}(G'[A]) = O\left( \max\left( 2D_c, D_u \right)/D_l \cdot n^{2/3} \cdot \lg n \right)
  \]
- Since the distance between two matched residues is no more than $D_c$, it follows that each residue in $A$ can be aligned to at most $O((1 + 2D_c/D_l)^3)$ residues in $B$
- This gives us the following theorem
  Let $A$ and $B$ be two protein structures in $\mathbb{R}^3$. Assume that the spatial positions of $A$ and $B$ are fixed and the distance between two equivalent residues is no more than $D_c$.
  Then there’s an algorithm with time complexity $O(n \cdot 2^{tw(\lg \Delta)})$ generating the optimal non-sequential alignment between $A$ and $B$

- where
  \[
  \begin{cases}
    n = \text{Protein size} \\
    \Delta = O((1 + 2D_c/D_l)^3) \\
    tw = O\left( \max\left( 2D_c, D_u \right)/D_l \cdot n^{2/3} \cdot \lg n \right)
  \end{cases}
  \]
Tree Decomposition on $G'[A]$

- Using Xu’s prior results, can prove that:

  \[
  \text{treewidth}(G'[A]) = O(\max (2D_c, D_u)/D_l \cdot n^{2/3} \cdot \lg n)
  \]

- Since the distance between two matched residues is no more than $D_c$, it follows that each residue in $A$ can be aligned to at most $O((1 + 2D_c/D_l)^3)$ residues in $B$

- This gives us the following theorem
  Let $A$ and $B$ be two protein structures in $\mathbb{R}^3$. Assume that the spatial positions of $A$ and $B$ are fixed and the distance between two equivalent residues is no more than $D_c$.
  Then there’s an algorithm with time complexity $O(n \cdot 2^{tw(\lg \Delta)})$ generating the optimal non-sequential alignment between $A$ and $B$

- where

  \[
  \begin{cases}
  n = \text{Protein size} \\
  \Delta = O((1 + 2D_c/D_l)^3) \\
  tw = O(\max (2D_c, D_u)/D_l \cdot n^{2/3} \cdot \lg n)
  \end{cases}
  \]

Note the sleight of hand here - this is just $O(n^*\Delta^{tw})$
Upper Bound on Tree Decomposition Algorithm

- Given $A$ and $B$, protein structures in $\mathbb{R}^3$. Assume that the spatial positions of $A$ and $B$ are fixed and the distance between two equivalent residues is no more than $D_c$.

Then there is an algorithm with time complexity $O(n \cdot 2^{tw(\lg \Delta)})$ generating the optimal non-sequential alignment between $A$ and $B$:

$$n = \text{Protein size}$$

$$\Delta = O((1 + 2 \cdot \frac{D_c}{D_l})^3)$$

$$tw = O(\max(2D_c, Du)/D_l \cdot \min(W_x W_y, W_x W_z, W_y W_z))$$
Upper Bound on Tree Decomposition Algorithm

- Given $A$ and $B$, protein structures in $\mathbb{R}^3$. Assume that the spatial positions of $A$ and $B$ are fixed and the distance between two equivalent residues is no more than $D_c$. Then there is an algorithm with time complexity $O(n \cdot 2^{tw(\lg \Delta)})$ generating the optimal non-sequential alignment between $A$ and $B$:

  \[
  \begin{align*}
  n &= \text{Protein size} \\
  \Delta &= O((1 + 2 \cdot D_c/D_l)^3) \\
  tw &= O(\max(2D_c, D_u)/D_l \cdot \min(W_xW_y, W_xW_z, W_yW_z))
  \end{align*}
  \]

We’re just refining our previous result

Instead of $n^{(2/3)} \cdot \lg n$
Upper Bound Proof

Sketch

• Assume left-bottom corner of rectangle is located at the origin and \( W_x W_y = \min(W_x W_y, W_x W_z, W_y W_z) \). Let \( D = \max(2D_c, D_u) \).

• Partition \( A \) into \( W_z / D \) blocks, using a set of hyperplanes \( z = j \cdot D, j = (1, 2, \ldots, W_z / D) \). Each block will have size \( W_x \cdot W_y \cdot D \).

• Because residues must be at least \( D_l \) apart, each block must have \( O(W_x W_y D / D_l^3) \) residues.

• Any two adjacent blocks form a tree-decomposition component, and the treewidth is no more than \( O(D / D_l^3 \cdot \min(W_x W_y, W_x W_z, W_y W_z)) \). Thus the time complexity of the tree-decomposition based algorithm is given by \( O(n \cdot 2^{tw(\log \Delta)}) \).
Upper Bound Proof

Sketch

- Assume left-bottom corner of rectangle is located at the origin and \( W_x W_y = \min(W_x W_y, W_x W_z, W_y W_z) \). Let \( D = \max(2D_c, D_u) \).

- Partition A into \( W_z / D \) blocks, using a set of hyperplanes \( z = j \cdot D, j = (1, 2, \ldots, W_z / D) \). Each block will have size \( W_x \cdot W_y \cdot D \).

- Because residues must be at least \( D_l \) apart, each block must have \( O(W_x W_y D / D_l^3) \) residues.

- Any two adjacent blocks form a tree-decomposition component, and the treewidth is no more than \( O(D / D_l^3 \cdot \min(W_x W_y, W_x W_z, W_y W_z)) \). Thus the time complexity of the tree-decomposition based algorithm is given by \( O(n \cdot 2^{tw(\lg \Delta)}) \).
Upper Bound Proof
Sketch

• Assume left-bottom corner of rectangle is located at the origin and $W_xW_y = \min(W_xW_y, W_xW_z, W_yW_z)$. Let $D = \max(2D_c, D_u)$.

• Partition $A$ into $W_z/D$ blocks, using a set of hyperplanes $z = j \cdot D, j = (1, 2, \ldots, W_z/D)$. Each block will have size $W_x \cdot W_y \cdot D$

• Because residues must be at least $D_l$ apart, each block must have $O(W_xW_yD/D_l^3)$ residues

• Any two adjacent blocks form a tree-decomposition component, and the treewidth is no more than $O(D/D_l^3 \cdot \min(W_xW_y, W_xW_z, W_yW_z))$. Thus the time complexity of the tree-decomposition based algorithm is given by $O(n \cdot 2^{tw(\lg \Delta)})$
Upper Bound Proof Sketch

- Assume left-bottom corner of rectangle is located at the origin and $W_x W_y = \min(W_x W_y, W_x W_z, W_y W_z)$. Let $D = \max(2D_c, D_u)$.

- Partition $A$ into $W_z/D$ blocks, using a set of hyperplanes $z = j \cdot D, j = (1, 2, \ldots, W_z/D)$. Each block will have size $W_x \cdot W_y \cdot D$.

- Because residues must be at least $D_l$ apart, each block must have $O(W_x W_y D/D_l^3)$ residues.

- Any two adjacent blocks form a tree-decomposition component, and the treewidth is no more than $O(D/D_l^3 \cdot \min(W_x W_y, W_x W_z, W_y W_z))$. Thus the time complexity of the tree-decomposition based algorithm is given by $O(n \cdot 2^{tw(lg \Delta)})$.

Assume $D_c = 3, D_u = 1$, then $D = 6$, so we have $W_z/6 = 3$ blocks.
Upper Bound Proof
Sketch

- Assume left-bottom corner of rectangle is located at the origin and $W_xW_y = \min(W_xW_y, W_xW_z, W_yW_z)$. Let $D = \max(2D_c, Du)$.

- Partition $A$ into $W_z/D$ blocks, using a set of hyperplanes $z = j \cdot D, j = (1, 2, \ldots, W_z/D)$. Each block will have size $W_x \cdot W_y \cdot D$.

- Because residues must be at least $D_l$ apart, each block must have $O(W_xW_yD/D_l^3)$ residues.

- Any two adjacent blocks form a tree-decomposition component, and the treewidth is no more than $O(D/D_l^3 \cdot \min(W_xW_y, W_xW_z, W_yW_z))$. Thus the time complexity of the tree-decomposition based algorithm is given by $O(n \cdot 2^{tw(\log \Delta)})$.

Assume $D_c = 3, Du = 1$, then $D = 6$, so we have $Wz/6 = 3$ blocks.
Upper Bound Proof Sketch

- Assume left-bottom corner of rectangle is located at the origin and \( W_x W_y = \min(W_x W_y, W_x W_z, W_y W_z) \). Let \( D = \max(2D_c, D_u) \).

- Partition A into \( W_z/D \) blocks, using a set of hyperplanes \( z = j \cdot D, j = (1, 2, \ldots, W_z/D) \). Each block will have size \( W_x \cdot W_y \cdot D \).

- Because residues must be at least \( D_l \) apart, each block must have \( O(W_x W_y D/D_l^3) \) residues.

- Any two adjacent blocks form a tree-decomposition component, and the treewidth is no more than \( O(D/D_l^3 \cdot \min(W_x W_y, W_x W_z, W_y W_z)) \). Thus the time complexity of the tree-decomposition based algorithm is given by \( O(n \cdot 2^{tw(\log \Delta)}) \).

Assume \( D_c = 3, D_u = 1 \), then \( D = 6 \), so we have \( W_z/6 = 3 \) blocks.
Upper Bound Proof Sketch

- Assume left-bottom corner of rectangle is located at the origin and $W_x W_y = \min(W_x W_y, W_x W_z, W_y W_z)$. Let $D = \max(2D_c, D_u)$.

- Partition $A$ into $W_z / D$ blocks, using a set of hyperplanes $z = j \cdot D, j = (1, 2, \ldots, W_z / D)$. Each block will have size $W_x \cdot W_y \cdot D$.

- Because residues must be at least $D_l$ apart, each block must have $O(W_x W_y D / D_l^3)$ residues.

- Any two adjacent blocks form a tree-decomposition component, and the treewidth is no more than $O(D / D_l^3 \cdot \min(W_x W_y, W_x W_z, W_y W_z))$. Thus the time complexity of the tree-decomposition based algorithm is given by $O(n \cdot 2^{tw(\log \Delta)})$.

Assume $D_c = 3, D_u = 1$, then $D = 6$, so we have $W_z / 6 = 3$ blocks.
Upper Bound Proof

Sketch

- Assume left-bottom corner of rectangle is located at the origin and \( W_x W_y = \min(W_x W_y, W_x W_z, W_y W_z) \). Let \( D = \max(2D_c, D_u) \).

- Partition \( A \) into \( W_z / D \) blocks, using a set of hyperplanes \( z = j \cdot D, j = (1, 2, \ldots, W_z / D) \). Each block will have size \( W_x \cdot W_y \cdot D \).

- Because residues must be at least \( D_l \) apart, each block must have \( O(W_x W_y D / D_l^3) \) residues.

- Any two adjacent blocks form a tree-decomposition component, and the treewidth is no more than \( O(D / D_l^3 \cdot \min(W_x W_y, W_x W_z, W_y W_z)) \). Thus the time complexity of the tree-decomposition based algorithm is given by \( O(n \cdot 2^{tw(\log \Delta)}) \).

Assume \( D_c = 3, D_u = 1 \), then \( D = 6 \), so we have \( W_z / 6 = 3 \) blocks.

- Each block between planes has a size of \( W_x \cdot W_y \cdot D \).
- Each block has on the order of \( O(W_x \cdot W_y \cdot D / D_l^3) \) residues, since each residue is at least \( D_l \) apart.
A PTAS for Protein Structure Alignment

- **PTAS**: Polynomial time approximate scheme--for a given $\varepsilon > 0$, a PTAS produces a solution of the optimization problem within $\varepsilon$ of the optimal

- Basic idea: chop up A into small blocks, align each block to B separately, and then combine the alignment results
A PTAS for Protein Structure Alignment
A PTAS for Protein Structure Alignment

- Given $A$ and $B$, protein structures in $\mathbb{R}^3$. Assume that the spatial positions of $A$ and $B$ are fixed and the distance between two equivalent residues is no more than $D_c$. Then there is an algorithm with time complexity $O(n \cdot k \cdot 2^{tw \log \Delta})$ generating a non-sequential alignment between $A$ and $B$ with an alignment score at least $(1 - 4/k)$ times the best possible.

- where
  \[
  \begin{aligned}
  n &= \text{Protein size} \\
  \Delta &= O((1 + 2 \cdot D_c/D_l)^3) \\
  tw &= O(k \cdot (\max(2D_c, Du))^2/D_l^3 \cdot \min(W_x, W_y, W_z))
  \end{aligned}
  \]
Intuition for protein partitioning

• Assume $W_x = \min(W_x, W_y, W_z)$

• Idea is to cut the protein structure $A$ into non-overlapping blocks using $k$ different partition schemes

• Then take each block, and tree-decompose it into components with no more than $O(k \cdot \max(2D_c, Du))^{2/D_l^3} W_x$ residues.

• Then the structural alignment between each block and $B$ can be done in time $O(\Delta^{O(k \cdot \max(2D_c, Du))^{2/D_l^3} W_x})$, where $\Delta$ is the max number of residues in $B$ that a residue in $A$ can align to.
1. Recall that $D = \max(2D_c, D_u)$. Take a group of hyperplanes, $y = y_j = j \cdot D, (j = 0, 1, \ldots, \frac{W_y}{D})$

2. Partition the protein $A$ into $\frac{W_y}{D}$ basic blocks along the $y$-axis, each with dimension $W_x \times D \times W_z$

3. Let $T_j = (j = 1, 2, \ldots, \frac{W_y}{D})$ be the residues between planes $y_{j-1}$ and $y_j$

4. Let $R_j = T_{j+1} \cup T_{j+2} \cup \cdots \cup T_{j+k-1}$

5. Let $G(R_j)$ be the subgraph induced by $R_j$ and $G(T_j)$ be the subgraph induced by $T_j$ plus the contact edges between $T_j$ and its two adjacent blocks

6. We’re going to optimize the structure alignment using $k$ different partition schemes and prove that at least one of them will give a good alignment
Cutting a Protein \( k \) ways

1. Recall that \( D = \max(2D_c, D_u) \).
   Take a group of hyperplanes, \( y = y_j = j \cdot D, (j = 0, 1, \ldots, W_y/D) \)

2. Partition the protein \( A \) into \( W_y/D \) basic blocks along the \( y \)-axis, each
   with dimension \( W_x \times D \times W_z \)

3. Let \( T_j = (j = 1, 2, \ldots, W_y/D) \) be the residues between planes \( y_{j-1} \) and \( y_j \)

4. Let \( R_j = T_{j+1} \cup T_{j+2} \cup \cdots \cup T_{j+k-1} \)

5. Let \( G(R_j) \) be the subgraph induced by \( R_j \) and \( G(T_j) \) be the subgraph
   induced by \( T_j \) plus the contact edges between \( T_j \) and its two adjacent blocks

6. We’re going to optimize the structure alignment using \( k \) different partition
   schemes and prove that at least one of them will give a good alignment

Assume \( D = 2 \)

in this case, and

\( k = 3 \)
Cutting a Protein k ways

1. Recall that $D = \max(2D_c, D_u)$.
   Take a group of hyperplanes, $y = y_j = j \cdot D, (j = 0, 1, \ldots, W_y/D)$

2. Partition the protein $A$ into $W_y/D$ basic blocks along the $y$-axis, each with dimension $W_x \times D \times W_z$

3. Let $T_j = (j = 1, 2, \ldots, W_y/D)$ be the residues between planes $y_{j-1}$ and $y_j$

4. Let $R_j = T_{j+1} \cup T_{j+2} \cup \cdots \cup T_{j+k-1}$

5. Let $G(R_j)$ be the subgraph induced by $R_j$ and $G(T_j)$ be the subgraph induced by $T_j$ plus the contact edges between $T_j$ and its two adjacent blocks

6. We’re going to optimize the structure alignment using $k$ different partition schemes and prove that at least one of them will give a good alignment

Assume $D = 2$ in this case, and $k = 3$
Cutting a Protein k ways

1. Recall that $D = \max(2D_c, D_u)$. Take a group of hyperplanes, $y = y_j = j \cdot D, (j = 0, 1, \ldots, W_y/D)$

2. Partition the protein $A$ into $W_y/D$ basic blocks along the $y$-axis, each with dimension $W_x \times D \times W_z$

3. Let $T_j = (j = 1, 2, \ldots, W_y/D)$ be the residues between planes $y_{j-1}$ and $y_j$

4. Let $R_j = T_{j+1} \cup T_{j+2} \cup \cdots \cup T_{j+k-1}$

5. Let $G(R_j)$ be the subgraph induced by $R_j$ and $G(T_j)$ be the subgraph induced by $T_j$ plus the contact edges between $T_j$ and its two adjacent blocks

6. We’re going to optimize the structure alignment using $k$ different partition schemes and prove that at least one of them will give a good alignment

Assume $D = 2$ in this case, and $k = 3$
Cutting a Protein $k$ ways

1. Recall that $D = \max(2D_c, D_u)$.
   Take a group of hyperplanes, $y = y_j = j \cdot D, (j = 0, 1, \ldots, W_y/D)$

2. Partition the protein $A$ into $W_y/D$ basic blocks along the $y$-axis, each with dimension $W_x \times D \times W_z$

3. Let $T_j = (j = 1, 2, \ldots, W_y/D)$ be the residues between planes $y_{j-1}$ and $y_j$

4. Let $R_j = T_{j+1} \cup T_{j+2} \cup \cdots \cup T_{j+k-1}$

5. Let $G(R_j)$ be the subgraph induced by $R_j$ and $G(T_j)$ be the subgraph induced by $T_j$ plus the contact edges between $T_j$ and its two adjacent blocks

6. We’re going to optimize the structure alignment using $k$ different partition schemes and prove that at least one of them will give a good alignment in this case, and $k = 3$
Cutting a Protein \( k \) ways

1. Recall that \( D = \max(2D_c, D_u) \).
   Take a group of hyperplanes, \( y = y_j = j \cdot D \), \((j = 0, 1, \ldots, W_y/D)\)

2. Partition the protein \( A \) into \( W_y/D \) basic blocks along the \( y \)-axis, each
   with dimension \( W_x \times D \times W_z \)

3. Let \( T_j = (j = 1, 2, \ldots, W_y/D) \) be the residues between planes \( y_{j-1} \) and \( y_j \)

4. Let \( R_j = T_{j+1} \cup T_{j+2} \cup \cdots \cup T_{j+k-1} \)

5. Let \( G(R_j) \) be the subgraph induced by \( R_j \) and \( G(T_j) \) be the subgraph
   induced by \( T_j \) plus the contact edges between \( T_j \) and its two adjacent
   blocks

6. We’re going to optimize the structure alignment using \( k \) different partition
   schemes and prove that at least one of them will give a good alignment
Cutting a Protein \( k \) ways

1. Recall that \( D = \max(2D_c, D_u) \).
   Take a group of hyperplanes, \( y = y_j = j \cdot D, (j = 0, 1, \ldots, W_y/D) \)

2. Partition the protein \( A \) into \( W_y/D \) basic blocks along the \( y \)-axis, each with dimension \( W_x \times D \times W_z \)

3. Let \( T_j = (j = 1, 2, \ldots, W_y/D) \) be the residues between planes \( y_{j-1} \) and \( y_j \)

4. Let \( R_j = T_{j+1} \cup T_{j+2} \cup \cdots \cup T_{j+k-1} \)

5. Let \( G(R_j) \) be the subgraph induced by \( R_j \) and \( G(T_j) \) be the subgraph induced by \( T_j \) plus the contact edges between \( T_j \) and its two adjacent blocks

6. We’re going to optimize the structure alignment using \( k \) different partition schemes and prove that at least one of them will give a good alignment

Assume \( D = 2 \) in this case, and \( k = 3 \)
Cutting a Protein $k$ ways

1. Recall that $D = \max(2D_c, D_u)$. Take a group of hyperplanes, $y = y_j = j \cdot D$, ($j = 0, 1, \ldots, W_y / D$)

2. Partition the protein $A$ into $W_y / D$ basic blocks along the $y$-axis, each with dimension $W_x \times D \times W_z$

3. Let $T_j = (j = 1, 2, \ldots, W_y / D)$ be the residues between planes $y_{j-1}$ and $y_j$

4. Let $R_j = T_{j+1} \cup T_{j+2} \cup \cdots \cup T_{j+k-1}$

5. Let $G(R_j)$ be the subgraph induced by $R_j$ and $G(T_j)$ be the subgraph induced by $T_j$ plus the contact edges between $T_j$ and its two adjacent blocks

6. We’re going to optimize the structure alignment using $k$ different partition schemes and prove that at least one of them will give a good alignment
1. Recall that $D = \max(2D_c, D_u)$. 
   Take a group of hyperplanes, $y = y_j = j \cdot D, (j = 0, 1, \ldots, W_y/D)$

2. Partition the protein $A$ into $W_y/D$ basic blocks along the $y$-axis, each with dimension $W_x \times D \times W_z$

3. Let $T_j = (j = 1, 2, \ldots, W_y/D)$ be the residues between planes $y_{j-1}$ and $y_j$

4. Let $R_j = T_{j+1} \cup T_{j+2} \cup \cdots \cup T_{j+k-1}$

5. Let $G(R_j)$ be the subgraph induced by $R_j$ and $G(T_j)$ be the subgraph induced by $T_j$ plus the contact edges between $T_j$ and its two adjacent blocks

6. We’re going to optimize the structure alignment using $k$ different partition schemes and prove that at least one of them will give a good alignment

Assume $D = 2$ in this case, and $k = 3$
Visualizing the Partitions, with $k = 3$
Visualizing the Partitions, with $k = 3$

1. For a given partition scheme $s, 0 \leq s < k$, let $RS_s = \bigcup_{j: j \% k = s} G(R_j)$ and $TS_s = \bigcup_{j: j \% k = s} G(T_j)$

2. Each residue in protein $A$ can only be aligned to residues in $B$ which are no more than $D_c$ away

3. It follows that any two residues in different $R_j$ will not be aligned to the same residue in $B$

4. Align the structure in $RS_s$ to protein $B$ first, and then align the remaining residues to $B$, using the tree-decomposition algorithm
Visualizing the Partitions, with $k = 3$
Visualizing the Partitions, with $k = 3$

Protein Structure cut into blocks
$T_1, ..., T_9$ along $y$-axis
Visualizing the Partitions, with $k = 3$

Protein Structure cut into blocks
T1, ..., T9 along y-axis

Partition Scheme 0: $R_0 = T_1 \cup T_2, R_3 = T_4 \cup T_5, R_6 = T_7 \cup T_8$
Visualizing the Partitions, with $k = 3$

Protein Structure cut into blocks
T1, ..., T9 along y-axis

Partition Scheme 0: $R_0 = T_1 \cup T_2$, $R_3 = T_4 \cup T_5$, $R_6 = T_7 \cup T_8$

Partition Scheme 1: $R_1 = T_2 \cup T_3$, $R_3 = T_5 \cup T_6$, $R_6 = T_8 \cup T_9$
Visualizing the Partitions, with \( k = 3 \)

Protein Structure cut into blocks
\( T_1, \ldots, T_9 \) along \( y \)-axis

Partition Scheme 0: \( R_0 = T_1 \cup T_2, R_3 = T_4 \cup T_5, R_6 = T_7 \cup T_8 \)

Partition Scheme 1: \( R_1 = T_2 \cup T_3, R_3 = T_5 \cup T_6, R_6 = T_8 \cup T_9 \)

Partition Scheme 2: \( R_2 = T_3 \cup T_4, R_3 = T_6 \cup T_7, R_6 = T_9 \cup T_1 \)
Alignment scores of partitions

1. Let $E(RS_s)$ and $E(TS_s)$ denote the optimal alignment scores of $RS_s$ and $TS_s$ respectively, and let $E_s = E(RS_s) + E(TS_s)$

2. Then $RS_s \cup TS_s$ contains all residues and inter-residue contact edges in the protein $A$

3. That means the alignment score $E_s$ must be at least as large as the globally optimized score $E_{opt}$
   
   $E_s = E(RS_s) + E(TS_s) \geq E_{opt}$

4. And summing across all values of $s$ we have:
   
   $\sum_{s=0}^{k-1} E_s \geq k \cdot E_{opt}$
Alignment scores of partitions

1. Let $E(RS_s)$ and $E(TS_s)$ denote the optimal alignment scores of $RS_s$ and $TS_s$ respectively, and let $E_s = E(RS_s) + E(TS_s)$

2. Then $RS_s \cup TS_s$ contains all residues and inter-residue contact edges in the protein $A$

3. That means the alignment score $E_s$ must be at least as large as the globally optimized score $E_{opt}$
   $$E_s = E(RS_s) + E(TS_s) \geq E_{opt}$$
   Because it’s easier to align smaller pieces of the protein

4. And summing across all values of $s$ we have:
   $$\sum_{s=0}^{k-1} E_s \geq k \cdot E_{opt}$$
1. Let \( E(RS_s) \) and \( E(TS_s) \) denote the optimal alignment scores of \( RS_s \) and \( TS_s \) respectively, and let \( E_s = E(RS_s) + E(TS_s) \)

2. Then \( RS_s \cup TS_s \) contains all residues and inter-residue contact edges in the protein \( A \)

3. That means the alignment score \( E_s \) must be at least as large as the globally optimized score \( E_{opt} \)
   \[
   E_s = E(RS_s) + E(TS_s) \geq E_{opt}
   \]
   Because it's easier to align smaller pieces of the protein

4. And summing across all values of \( s \) we have:
   \[
   \sum_{s=0}^{k-1} E_s \geq k \cdot E_{opt}
   \]
Alignment scores of partitions

1. Let $E(RS_s)$ and $E(TS_s)$ denote the optimal alignment scores of $RS_s$ and $TS_s$ respectively, and let $E_s = E(RS_s) + E(TS_s)$

2. Then $RS_s \cup TS_s$ contains all residues and inter-residue contact edges in the protein $A$

3. That means the alignment score $E_s$ must be at least as large as the globally optimized score $E_{opt}$
   
   $E_s = E(RS_s) + E(TS_s) \geq E_{opt}$
   
   Because it’s easier to align smaller pieces of the protein

4. And summing across all values of $s$ we have:
   
   $\sum_{s=0}^{k-1} E_s \geq k \cdot E_{opt}$
Alignment scores of partitions

1. Let $E(RS_s)$ and $E(TS_s)$ denote the optimal alignment scores of $RS_s$ and $TS_s$ respectively, and let $E_s = E(RS_s) + E(TS_s)$

2. Then $RS_s \cup TS_s$ contains all residues and inter-residue contact edges in the protein $A$

3. That means the alignment score $E_s$ must be at least as large as the globally optimized score $E_{opt}$

$$E_s = E(RS_s) + E(TS_s) \geq E_{opt}$$

Because it’s easier to align smaller pieces of the protein

4. And summing across all values of $s$ we have:

$$\sum_{s=0}^{k-1} E_s \geq k \cdot E_{opt}$$
Alignment scores of partitions

Because it's easier to align smaller pieces of the protein

Protein A
Alignment scores of partitions

RS0

T1 T2 T4 T5 T7 T8

+ TS0

T3 T6 T9

= Protein A

R0 R3 R6

T1 T2 T3 T4 T5 T6 T7 T8 T9
Alignment scores of partitions

1. If we can prove that $\sum_{s=0}^{k-1} E_s \leq 4 \cdot E_{opt}$ then we can show:
Alignment scores of partitions

1. If we can prove that $\sum_{s=0}^{k-1} E_s \leq 4 \cdot E_{opt}$ then we can show:

$$\sum_{s=0}^{k-1} E(RS_s) + E(TS_s) \geq k \cdot E_{opt},$$

so
Alignment scores of partitions

1. If we can prove that $\sum_{s=0}^{k-1} E_s \leq 4 \cdot E_{opt}$ then we can show:

$$\sum_{s=0}^{k-1} E(RS_s) + E(TS_s) \geq k \cdot E_{opt},$$

so

$$\sum_{s=0}^{k-1} E(TS_s) \geq k \cdot E_{opt} - \sum_{s=0}^{k-1} E(RS_s),$$

using the inequality:

\[
\sum_{s=0}^{k-1} E_s \leq 4 \cdot E_{opt}
\]
Alignment scores of partitions

1. If we can prove that $\sum_{s=0}^{k-1} E_s \leq 4 \cdot E_{opt}$ then we can show:

$$\sum_{s=0}^{k-1} E(RS_s) + E(TS_s) \geq k \cdot E_{opt},$$

so

$$\sum_{s=0}^{k-1} E(TS_s) \geq k \cdot E_{opt} - \sum_{s=0}^{k-1} E(RS_s),$$

using the inequality:

$$4 \cdot E_{opt} \geq \sum_{s=0}^{k-1} E(TS_s) \geq k \cdot E_{opt} - \sum_{s=0}^{k-1} E(RS_s),$$

yielding:
Alignment scores of partitions

1. If we can prove that $\sum_{s=0}^{k-1} E_s \leq 4 \cdot E_{opt}$ then we can show:

   \[ \sum_{s=0}^{k-1} E(RS_s) + E(TS_s) \geq k \cdot E_{opt}, \text{ so} \]
   \[ \sum_{s=0}^{k-1} E(TS_s) \geq k \cdot E_{opt} - \sum_{s=0}^{k-1} E(RS_s), \text{ using the inequality:} \]
   \[ 4 \cdot E_{opt} \geq \sum_{s=0}^{k-1} E(TS_s) \geq k \cdot E_{opt} - \sum_{s=0}^{k-1} E(RS_s), \text{ yielding:} \]
   \[ \sum_{s=0}^{k-1} E(RS_s) \geq (k - 4) \cdot E_{opt} \]
Alignment scores of partitions

1. If we can prove that $\sum_{s=0}^{k-1} E_s \leq 4 \cdot E_{opt}$ then we can show:

$$\sum_{s=0}^{k-1} E(RS_s) + E(TS_s) \geq k \cdot E_{opt},$$

so

$$\sum_{s=0}^{k-1} E(TS_s) \geq k \cdot E_{opt} - \sum_{s=0}^{k-1} E(RS_s),$$

using the inequality:

$$4 \cdot E_{opt} \geq \sum_{s=0}^{k-1} E(TS_s) \geq k \cdot E_{opt} - \sum_{s=0}^{k-1} E(RS_s),$$

yielding:

$$\sum_{s=0}^{k-1} E(RS_s) \geq (k - 4) \cdot E_{opt}.$$ 

2. If we can show this result, then it follows that there is at least one $s^*$ such that $E(RS_{s^*}) \geq (1 - 4/k) \cdot E_{opt}$—just divide the RHS of the last bullet by $k$.

3. Therefore, there’s a structural alignment with score at least $(1 - 4/k) \cdot E_{opt}$.

Protein A
Proving the bounds on structural alignment

- \( \bigcup_s T S_s = \bigcup_j G(T_j) \), which can in turn be divided into four disjoint subsets:
  - Each subset is \( \bigcup_j G(T_{l+4j}) \) with \( 0 \leq l < 4 \) such that for a given \( l \),
  \( G(T_{l+4j_1}) \) and \( G(T_{l+4j_2}) \) are disjoint if \( j_1 \neq j_2 \)

- Then the whole alignment score between \( \bigcup_j G(T_{l+4j}) \) and \( B \) is no more than \( E_{\text{opt}} \) no matter how we do the alignment

- It follows that \( \sum_{s=0}^{k-1} E(TS_s) \leq 4 \cdot E_{\text{opt}} \)
Proving the bounds on structural alignment

• $\bigcup_s T S_s = \bigcup_j G(T_j)$, which can in turn be divided into four disjoint subsets:
  
  – Each subset is $\bigcup_j G(T_{l+4j})$ with $(0 \leq l < 4)$ such that for a given $l$, $G(T_{l+4j_1})$ and $G(T_{l+4j_2})$ are disjoint if $j_1 \neq j_2$

• Then the whole alignment score between $\bigcup_j G(T_{l+4j})$ and $B$ is no more than $E_{opt}$ no matter how we do the alignment

• It follows that $\sum_{s=0}^{k-1} E(T S_s) \leq 4 \cdot E_{opt}$

\[ L = 0 \rightarrow G(T4) \cup G(T8) \]
\[ L = 1 \rightarrow G(T1) \cup G(T5) \cup G(T9) \]
\[ L = 2 \rightarrow G(T2) \cup G(T6) \]
\[ L = 3 \rightarrow G(T3) \cup G(T7) \]
Proving the bounds on structural alignment

• $\bigcup_s TS_s = \bigcup j G(T_j)$, which can in turn be divided into four disjoint subsets:
  - Each subset is $\bigcup j G(T_{l+4j})$ with $0 \leq l < 4$ such that for a given $l$, $G(T_{l+4j_1})$ and $G(T_{l+4j_2})$ are disjoint if $j_1 \neq j_2$

• Then the whole alignment score between $\bigcup j G(T_{l+4j})$ and $B$ is no more than $E_{opt}$ no matter how we do the alignment

• It follows that $\sum_{s=0}^{k-1} E(TS_s) \leq 4 \cdot E_{opt}$

\[\begin{align*}
L = 0 & \rightarrow G(T4) \cup G(T8) \\
L = 1 & \rightarrow G(T1) \cup G(T5) \cup G(T9) \\
L = 2 & \rightarrow G(T2) \cup G(T6) \quad \text{Alignment to B} \\
L = 3 & \rightarrow G(T3) \cup G(T7) \quad \text{no more than } E_{opt}
\end{align*}\]
Time complexity of the structural alignment

- For a particular partition scheme $s$, the algorithm aligns the partial structure in $RS_s$ to $B$
- From our previous result, the structural alignment between the partial structure in $R_j$ (a subset of $RS_s$) and $B$ can be optimized in $O(|R_j| \cdot 2^{tw \lg \Delta})$
- Once a structural alignment between $RS_s$ and $B$ is fixed, the algorithm aligns the remaining structure to $B$
- So the time complexity for structural alignment of each partition is $O(n \cdot 2^{tw \lg \Delta})$ and the time complexity of the algorithm is $O(k \cdot n \cdot 2^{tw \lg \Delta})$—since there are $k$ partitions
Cutting the protein in two dimensions

• If we cut the protein in two dimensions, using hyperplanes in $x$ and $y$, then we can arrive at the following theorem:

• Let $A$ and $B$ be two protein structures in $\mathbb{R}^3$. Assume that the spatial positions of $A$ and $B$ are fixed and the distance between the two equivalent residues is no more than $D_c$. Then there is an algorithm with time complexity $O(n \cdot k^2 \cdot 2^{tw \frac{1}{k} \Delta})$ generating a non-sequential alignment between $A$ and $B$ with an alignment score at least $(1 - 8/k)$ times the best possible

\[
\begin{align*}
\text{n} &= \text{Protein size} \\
\text{k} &= \text{a positive integer} \\
\Delta &= O((1 + 2 \cdot D_c/D_l)^3) \\
tw &= O(k^2 \cdot (\max(2D_c, Du))^2/D_l^3)
\end{align*}
\]
Cutting the protein in two dimensions

- If we cut the protein in two dimensions, using hyperplanes in $x$ and $y$, then we can arrive at the following theorem:

- Let $A$ and $B$ be two protein structures in $\mathbb{R}^3$. Assume that the spatial positions of $A$ and $B$ are fixed and the distance between the two equivalent residues is no more than $D_c$. Then there is an algorithm with time complexity $O(n \cdot k^2 \cdot 2^{tw \lg \Delta})$ generating a non-sequential alignment between $A$ and $B$ with an alignment score at least $(1 - 8/k)$ times the best possible

$\begin{align*}
\{ & n = \text{Protein size} \\
& k \text{ is a positive integer} \\
& \Delta = O((1 + 2 \cdot D_c / D_l)^3) \\
& tw = O(k^2 \cdot (\max(2D_c, D_u))^2 / D_l^3) \}
\end{align*}$

Proof similar to previous one, except this time we use two hyper planes--omitted for this talk
Structural Alignment with Transformation Preliminaries
Structural Alignment with Transformation Preliminaries

Before we had kept both proteins spatially fixed—now we’re going to move one
Structural Alignment with Transformation Preliminaries

Before we had kept both proteins spatially fixed—now we’re going to move one

- We take $W_x(A), W_y(A), W_z(B)$ to be the dimensions of the minimum-size rectangle that encloses $A$ and $R$ to be the radius of the protein structure.

- All the unit axes form the surface of a sphere with radius 1, the rotation angle ranges from 0 to $2\pi$, and all possible translations between $A$ and $B$ are in a rectangle with dimensions $W_x(A) + W_x(B), W_y(A) + W_y(B), W_z(A) + W_z(B)$.
Structural Alignment with Transformation Preliminaries

Before we had kept both proteins spatially fixed—now we’re going to move one
Structural Alignment with Transformation Preliminaries
Structural Alignment with Transformation Preliminaries

- We’re going to discretize the whole transformation space into a polynomial number of transformations, similar to Kolodny’s approach.
- By using only these discrete transformations, we’re going to find an alignment between two proteins with an alignment score very close to the optimal.
Optimal alignment with a discrete number of transformations

- Let $OPT(D_c)$ denote the optimal alignment score between two proteins $A$ and $B$ when the distance between two equivalent residues is no more than $D_c$ after two proteins are superimposed. There is an algorithm to generate a non-sequential alignment between two proteins such that

1. Time complexity is either $O(k^2 \cdot n^3 \cdot \Delta^{tw}/(\epsilon D_c)^6)$ or $O(k^2 \cdot n^5 \cdot \Delta^{tw}/(\epsilon D_c)^6)$

   - where $\Delta = O((D_c(1 + \epsilon)/D_l)^3)$ and $tw = O(k^2 \cdot ((\max(2D_c, Du))/D_l)^3)$

2. Alignment score is no less than $(1 - \Theta(1/k)) \cdot OPT(D_c)$, and

3. the distance between two equivalent residues is no more than $(1+\epsilon)D_c$
Optimal alignment with a discrete number of transformations

- Let $OPT(D_c)$ denote the optimal alignment score between two proteins $A$ and $B$ when the distance between two equivalent residues is no more than $D_c$ after two proteins are superimposed. There is an algorithm to generate a non-sequential alignment between two proteins such that

1. Time complexity is either $O(k^2 \cdot n^3 \cdot \Delta^{tw}/(\epsilon D_c)^6)$ or $O(k^2 \cdot n^5 \cdot \Delta^{tw}/(\epsilon D_c)^6)$

   - where $\Delta = O((D_c(1 + \epsilon)/D_l)^3)$ and $tw = O(k^2 \cdot ((\max(2D_c, Du))/D_l)^3)$

2. Alignment score is no less than $(1 - \Theta(1/k)) \cdot OPT(D_c)$, and

3. the distance between two equivalent residues is no more than $(1+\epsilon)D_c$

As the number of partitions, $k$ gets larger and $\epsilon$ gets smaller, your accuracy improves, but the algorithm takes longer
Construct two “close” transformations

- Given two possible rigid transformations \((w_1, \theta_1, t_1)\) and \((w_2, \theta_2, t_2)\), assume they satisfy:
  
  \[
  \begin{align*}
  &- |w_1 - w_2| \leq \epsilon D_c/3R \\
  &- |\theta_1 - \theta_2| \leq \epsilon D_c/3R \\
  &- |t_1 - t_2| \leq \epsilon D_c/3 \\
  \end{align*}
  \]

- Let \(\hat{A}_i\) be the transformation of \(A\) by \((w_i, \theta_i, t_i)\) for \(i = 1, 2\).

- For any residue \(r\) in \(A\), let \(\hat{r}_i\) denote the image of \(r\) in \(\hat{A}_i\)

- Then it follows that \(|\hat{r}_1 - \hat{r}_2| \leq \epsilon D_c\).
Construct two “close” transformations

- Given two possible rigid transformations \((w_1, \theta_1, t_1)\) and \((w_2, \theta_2, t_2)\), assume they satisfy:
  - \(|w_1 - w_2| \leq \epsilon D_c / 3R\)
  - \(|\theta_1 - \theta_2| \leq \epsilon D_c / 3R\)
  - \(|t_1 - t_2| \leq \epsilon D_c / 3\)

- Let \(\hat{A}_i\) be the transformation of \(A\) by \((w_i, \theta_i, t_i)\) for \(i = 1, 2\).

- For any residue \(r\) in \(A\), let \(\hat{r}_i\) denote the image of \(r\) in \(\hat{A}_i\)

- Then it follows that \(|\hat{r}_1 - \hat{r}_2| \leq \epsilon D_c\).
Construct two “close” transformations

- Given two possible rigid transformations \((w_1, \theta_1, t_1)\) and \((w_2, \theta_2, t_2)\), assume they satisfy:
  - \(|w_1 - w_2| \leq \epsilon D_c / 3R\)
  - \(|\theta_1 - \theta_2| \leq \epsilon D_c / 3R\)
  - \(|t_1 - t_2| \leq \epsilon D_c / 3\)

- Let \(\hat{A}_i\) be the transformation of \(A\) by \((w_i, \theta_i, t_i)\) for \(i = 1, 2\).

- For any residue \(r\) in \(A\), let \(\hat{r}_i\) denote the image of \(r\) in \(\hat{A}_i\)

- Then it follows that \(|\hat{r}_1 - \hat{r}_2| \leq \epsilon D_c\).
Construct two “close” transformations

- Given two possible rigid transformations \((w_1, \theta_1, t_1)\) and \((w_2, \theta_2, t_2)\), assume they satisfy:
  
  - \(|w_1 - w_2| \leq \epsilon D_c / 3R\)
  - \(|\theta_1 - \theta_2| \leq \epsilon D_c / 3R\)
  - \(|t_1 - t_2| \leq \epsilon D_c / 3\)

- Let \(\hat{A}_i\) be the transformation of \(A\) by \((w_i, \theta_i, t_i)\) for \(i = 1, 2\).

- For any residue \(r\) in \(A\), let \(\hat{r}_i\) denote the image of \(r\) in \(\hat{A}_i\).

- Then it follows that \(|\hat{r}_1 - \hat{r}_2| \leq \epsilon D_c\).
Construct two “close” transformations

- Given two possible rigid transformations \((w_1, \theta_1, t_1)\) and \((w_2, \theta_2, t_2)\), assume they satisfy:
  - \(|w_1 - w_2| \leq \epsilon D_c/3R\)
  - \(|\theta_1 - \theta_2| \leq \epsilon D_c/3R\)
  - \(|t_1 - t_2| \leq \epsilon D_c/3\)

- Let \(\hat{A}_i\) be the transformation of \(A\) by \((w_i, \theta_i, t_i)\) for \(i = 1, 2\).

- For any residue \(r\) in \(A\), let \(\hat{r}_i\) denote the image of \(r\) in \(\hat{A}_i\)

- Then it follows that \(|\hat{r}_1 - \hat{r}_2| \leq \epsilon D_c\).

At most \(\epsilon D_c\)
Construct two “close” transformations

• Now take $N_i(r, d)$ to be the set of residues in $B$ such that the distance between $\hat{r}_i$ and any residue in $N_i(r, d)$ is no more than $d$

• We’ll show that $N_1(r, D_c) \subseteq N_2(r, D_c(1+\epsilon))$ and $N_2(r, D_c) \subseteq N_1(r, D_c(1+\epsilon))$
Construct two “close” transformations
Construct two “close” transformations

\[ j \in N_1(r, D_c) \]

\[ k \in N_2(r, (1 + \epsilon)D_c) \]

\( j, k \) are both proteins in B
Construct two “close” transformations
The size of the discretized space

- Let $OPT(d,w,\theta,t)$ be the optimal alignment score between $A$ and $B$ after applying the transformation $(w,\theta,t)$ to $A$ and forcing the deviation between two equivalent residues to be no more than $d$

- Given an $\epsilon > 0$, let’s discretize as follows:
  - unit axis with step size $\epsilon D_c/(3R) \times \epsilon D_c/(3R)$
  - rotation angle with step size $\epsilon D_c/(3R)$
  - translation with step size $\epsilon D_c/3$

- Just as Kolodny showed, we have $O(R^3V/(\epsilon^6D_c^6))$ points
  - where $V = (W_x(A) + W_x(B))(W_y(A) + W_y(B))(W_z(A) + W_z(B))$
The size of the discretized space

- Let $OPT(d,w,\theta,t)$ be the optimal alignment score between $A$ and $B$ after applying the transformation $(w,\theta,t)$ to $A$ and forcing the deviation between two equivalent residues to be no more than $d$

- Given an $\epsilon > 0$, let’s discretize as follows:
  - unit axis with step size $\epsilon D_c/(3R) \times \epsilon D_c/(3R)$
  - rotation angle with step size $\epsilon D_c/(3R)$
  - translation with step size $\epsilon D_c/3$

- Just as Kolodny showed, we have $O(R^3V/(\epsilon^6D_c^6))$ points
  - where $V = (W_x(A) + W_x(B))(W_y(A) + W_y(B))(W_z(A) + W_z(B))$

\[
\frac{\epsilon D_c}{3} \quad \frac{2\epsilon D_c}{3} \quad \frac{3\epsilon D_c}{3}
\]

e tc...

\[
\begin{array}{c}
\text{X} \\
\text{etc...}
\end{array}
\]
The size of the discretized space

\[ \epsilon D_c / 3 \quad 2\epsilon D_c / 3 \quad 3\epsilon D_c / 3 \quad \text{etc...} \]
The size of the discretized space

- Take $\Sigma$ to be this whole set of discrete transformations
- For any possible transformation $(w_1, \theta_1, t_1)$ there is a 'nearby' discrete transformation $(w_2, \theta_2, t_2) \in \Sigma$
  - where 'nearby' means that $|w_1 - w_2| \leq \epsilon D_c/(3R)$, $|\theta_1 - \theta_2| \leq \epsilon D_c/(3R)$, and $|t_1 - t_2| \leq \epsilon D_c/3$
- From our prior result, we know that $OPT(D_c, w_1, \theta_1, t_1) \leq OPT(D_c(1 + \epsilon), w_2, \theta_2, t_2)$
- It follows that $OPT(D_c) \leq \max_{(w,\theta,t) \in \Sigma} OPT(D_c(1 + \epsilon), w, \theta, t)$

$$
\frac{\epsilon D_c}{3} \quad \frac{2\epsilon D_c}{3} \quad \frac{3\epsilon D_c}{3} \quad \text{etc...}
$$
Alignments in the discretized space

- From our previous result, for each discrete transformation, there’s an algorithm with time complexity $O(k^2n\Delta^{tw})$ to calculate $OPT(D_c(1 + \epsilon), w_2, \theta_2, t_2)$

- This algorithm generates a score of at least $(1 - \Theta(1/k))OPT(D_c(1 + \epsilon), w_2, \theta_2, t_2)$

- Going through all possible transformation in $\Sigma$, we can generate an alignment with score at least $(1 - \Theta(1/k))OPT(D_c)$, with the deviation between two residues at most $(1 + \epsilon)D_c$

- The running time of this procedure is then $O(k^2n\Delta^{tw}R^3V/(\epsilon D_c)^6)$ where $V$ is proportional to protein size, $n$, and

  where $R = \begin{cases} O(n^{1/3}) & \text{for a globular protein, so time complexity is } O(k^2 \cdot n^3 \cdot \Delta^{tw}/(\epsilon D_c)^6) \\ O(n) & \text{for a regular protein, so time complexity is } O(k^2 \cdot n^5 \cdot \Delta^{tw}/(\epsilon D_c)^6) \end{cases}$
Structural Alignment

Problem

• What does this mean?
  – As long as the ratio between $\max(2D_c, D_u)$ and $D_l$ is small relative to protein size, $n$, there is a polynomial-time approximation scheme for the non-sequential protein structure alignment problem.
  – If $\max(2D_c, D_u), l, k$ and $\epsilon$ are given constants, then the time complexity is polynomial.
Structural Alignment

Problem
Structural Alignment Problem

• Taking this discretization technique and the exact protein structure alignment algorithm outlined earlier we have the following result:

There is an algorithm to generate a non-sequential alignment with a score of at least $OPT(D_c)$ such that

1. Time complexity of the algorithm is $O(n^3 \Delta^{tw}/(\epsilon D_c)^6)$ for globular proteins, or $O(n^5 \Delta^{tw}/(\epsilon D_c)^6)$ for others, where:

   - $\Delta = O(((1 + \epsilon)D_c/D_l)^3)$
   - $tw = O((\max(2D_c, D_u)/D_l)n^{(2/3)} \log n)$

2. Distance between two equivalent residues is no more than $(1 + \epsilon)D_c$
Experimental Results
Experimental Results

- Xu set $D_u = 6.75\AA$ and $D_c = 3.0\AA$, and discretized as follows:
  - unit rotation axis into a $36 \times 18$ longitude-latitude grid
  - rotation angle evenly distributed into 36 possible angles
  - translation space into $35 \times 35 \times 35$ points
  - For a total of 1,000,188,000 discrete transformations
Experimental Results
Experimental Results

Alignments of Proteins with Flavodoxin-like folds

<table>
<thead>
<tr>
<th>protein (A)</th>
<th>protein (B)</th>
<th># residues (A)</th>
<th># residues (B)</th>
<th># contacts (A)</th>
<th># contacts (B)</th>
<th>time (s)</th>
<th># aligned contacts</th>
</tr>
</thead>
<tbody>
<tr>
<td>1b00a</td>
<td>1dbwa</td>
<td>122</td>
<td>123</td>
<td>441</td>
<td>457</td>
<td>2244</td>
<td>249</td>
</tr>
<tr>
<td>1b00a</td>
<td>1nat</td>
<td>122</td>
<td>119</td>
<td>441</td>
<td>435</td>
<td>2268</td>
<td>279</td>
</tr>
<tr>
<td>1b00a</td>
<td>1qmpc</td>
<td>122</td>
<td>125</td>
<td>441</td>
<td>452</td>
<td>1604</td>
<td>317</td>
</tr>
<tr>
<td>1nat</td>
<td>1b00b</td>
<td>119</td>
<td>122</td>
<td>435</td>
<td>444</td>
<td>1650</td>
<td>262</td>
</tr>
<tr>
<td>1nat</td>
<td>1dbwa</td>
<td>119</td>
<td>123</td>
<td>435</td>
<td>457</td>
<td>1315</td>
<td>285</td>
</tr>
<tr>
<td>1nat</td>
<td>4mya</td>
<td>119</td>
<td>118</td>
<td>435</td>
<td>446</td>
<td>1626</td>
<td>351</td>
</tr>
<tr>
<td>1qmpc</td>
<td>1b00b</td>
<td>125</td>
<td>122</td>
<td>461</td>
<td>444</td>
<td>2277</td>
<td>332</td>
</tr>
<tr>
<td>1qmpc</td>
<td>4mya</td>
<td>125</td>
<td>118</td>
<td>461</td>
<td>446</td>
<td>1044</td>
<td>373</td>
</tr>
<tr>
<td>4mya</td>
<td>1b00b</td>
<td>118</td>
<td>122</td>
<td>446</td>
<td>444</td>
<td>1501</td>
<td>289</td>
</tr>
</tbody>
</table>
Experimental Results

Alignments of Proteins with Flavodoxin-like folds

<table>
<thead>
<tr>
<th>protein (A)</th>
<th>protein (B)</th>
<th># residues (A)</th>
<th># residues (B)</th>
<th># contacts (A)</th>
<th># contacts (B)</th>
<th>time (s)</th>
<th># aligned contacts</th>
</tr>
</thead>
<tbody>
<tr>
<td>1b00a</td>
<td>1dbwa</td>
<td>122</td>
<td>123</td>
<td>441</td>
<td>457</td>
<td>2244</td>
<td>249</td>
</tr>
<tr>
<td>1b00a</td>
<td>1nat</td>
<td>122</td>
<td>119</td>
<td>441</td>
<td>435</td>
<td>2268</td>
<td>279</td>
</tr>
<tr>
<td>1b00a</td>
<td>1qmpc</td>
<td>122</td>
<td>125</td>
<td>441</td>
<td>452</td>
<td>1604</td>
<td>317</td>
</tr>
<tr>
<td>1nat</td>
<td>1b00b</td>
<td>119</td>
<td>122</td>
<td>435</td>
<td>444</td>
<td>1650</td>
<td>262</td>
</tr>
<tr>
<td>1nat</td>
<td>1dbwa</td>
<td>119</td>
<td>123</td>
<td>435</td>
<td>457</td>
<td>1315</td>
<td>285</td>
</tr>
<tr>
<td>1nat</td>
<td>4tmya</td>
<td>119</td>
<td>118</td>
<td>435</td>
<td>446</td>
<td>1626</td>
<td>351</td>
</tr>
<tr>
<td>1qmpc</td>
<td>1b00b</td>
<td>125</td>
<td>122</td>
<td>461</td>
<td>444</td>
<td>2277</td>
<td>332</td>
</tr>
<tr>
<td>1qmpc</td>
<td>4tmya</td>
<td>125</td>
<td>118</td>
<td>461</td>
<td>446</td>
<td>1044</td>
<td>373</td>
</tr>
<tr>
<td>4tmya</td>
<td>1b00b</td>
<td>118</td>
<td>122</td>
<td>446</td>
<td>444</td>
<td>1501</td>
<td>289</td>
</tr>
</tbody>
</table>

Alignments of Proteins with Cupredoxins folds

<table>
<thead>
<tr>
<th>protein (A)</th>
<th>protein (B)</th>
<th># residues (A)</th>
<th># residues (B)</th>
<th># contacts (A)</th>
<th># contacts (B)</th>
<th>time (s)</th>
<th># aligned contacts</th>
</tr>
</thead>
<tbody>
<tr>
<td>1bawa</td>
<td>1byoh</td>
<td>105</td>
<td>99</td>
<td>387</td>
<td>350</td>
<td>799</td>
<td>306</td>
</tr>
<tr>
<td>1bawn</td>
<td>1byoa</td>
<td>105</td>
<td>99</td>
<td>389</td>
<td>355</td>
<td>795</td>
<td>307</td>
</tr>
<tr>
<td>1bawa</td>
<td>1ptl</td>
<td>105</td>
<td>97</td>
<td>387</td>
<td>340</td>
<td>1309</td>
<td>298</td>
</tr>
<tr>
<td>1byoa</td>
<td>1kdi</td>
<td>99</td>
<td>102</td>
<td>355</td>
<td>361</td>
<td>1746</td>
<td>257</td>
</tr>
<tr>
<td>1byob</td>
<td>1nin</td>
<td>99</td>
<td>105</td>
<td>350</td>
<td>376</td>
<td>1679</td>
<td>231</td>
</tr>
<tr>
<td>1byob</td>
<td>1kdi</td>
<td>99</td>
<td>102</td>
<td>350</td>
<td>361</td>
<td>1478</td>
<td>264</td>
</tr>
<tr>
<td>1kpl</td>
<td>2b3ia</td>
<td>97</td>
<td>97</td>
<td>340</td>
<td>341</td>
<td>1188</td>
<td>226</td>
</tr>
<tr>
<td>2b3ian</td>
<td>2pcy</td>
<td>97</td>
<td>99</td>
<td>341</td>
<td>357</td>
<td>1053</td>
<td>265</td>
</tr>
<tr>
<td>2b3ian</td>
<td>2plt</td>
<td>97</td>
<td>98</td>
<td>341</td>
<td>367</td>
<td>1010</td>
<td>293</td>
</tr>
<tr>
<td>2pcy</td>
<td>2plt</td>
<td>99</td>
<td>98</td>
<td>357</td>
<td>367</td>
<td>527</td>
<td>343</td>
</tr>
</tbody>
</table>
Experimental Results

Alignments of Proteins with Flavodoxin-like folds

<table>
<thead>
<tr>
<th>protein (A)</th>
<th>protein (B)</th>
<th># residues (A)</th>
<th># residues (B)</th>
<th># contacts (A)</th>
<th># contacts (B)</th>
<th>time (s)</th>
<th># aligned contacts</th>
</tr>
</thead>
<tbody>
<tr>
<td>1b00a</td>
<td>1dwpa</td>
<td>122</td>
<td>123</td>
<td>441</td>
<td>457</td>
<td>2244</td>
<td>249</td>
</tr>
<tr>
<td>1b00a</td>
<td>1nat</td>
<td>122</td>
<td>119</td>
<td>441</td>
<td>435</td>
<td>2268</td>
<td>279</td>
</tr>
<tr>
<td>1b00a</td>
<td>1qmpc</td>
<td>122</td>
<td>125</td>
<td>441</td>
<td>452</td>
<td>1604</td>
<td>317</td>
</tr>
<tr>
<td>1nat</td>
<td>1b00b</td>
<td>119</td>
<td>122</td>
<td>435</td>
<td>444</td>
<td>1650</td>
<td>262</td>
</tr>
<tr>
<td>1nat</td>
<td>1dwpa</td>
<td>119</td>
<td>123</td>
<td>435</td>
<td>457</td>
<td>1315</td>
<td>285</td>
</tr>
<tr>
<td>1nat</td>
<td>4mya</td>
<td>119</td>
<td>118</td>
<td>435</td>
<td>446</td>
<td>1626</td>
<td>351</td>
</tr>
<tr>
<td>1qmpc</td>
<td>1b00b</td>
<td>125</td>
<td>122</td>
<td>461</td>
<td>444</td>
<td>2277</td>
<td>332</td>
</tr>
<tr>
<td>1qmpc</td>
<td>4mya</td>
<td>125</td>
<td>118</td>
<td>461</td>
<td>446</td>
<td>1044</td>
<td>373</td>
</tr>
<tr>
<td>4tmya</td>
<td>1b00b</td>
<td>118</td>
<td>122</td>
<td>446</td>
<td>444</td>
<td>1501</td>
<td>289</td>
</tr>
</tbody>
</table>

Alignments of Proteins with Cupredoxins folds

<table>
<thead>
<tr>
<th>protein (A)</th>
<th>protein (B)</th>
<th># residues (A)</th>
<th># residues (B)</th>
<th># contacts (A)</th>
<th># contacts (B)</th>
<th>time (s)</th>
<th># aligned contacts</th>
</tr>
</thead>
<tbody>
<tr>
<td>1bawa</td>
<td>1byoab</td>
<td>105</td>
<td>99</td>
<td>387</td>
<td>350</td>
<td>799</td>
<td>306</td>
</tr>
<tr>
<td>1bawa</td>
<td>1byoae</td>
<td>105</td>
<td>99</td>
<td>389</td>
<td>355</td>
<td>795</td>
<td>307</td>
</tr>
<tr>
<td>1bawa</td>
<td>1pfa</td>
<td>105</td>
<td>97</td>
<td>387</td>
<td>340</td>
<td>1309</td>
<td>298</td>
</tr>
<tr>
<td>1byoa</td>
<td>1kdi</td>
<td>99</td>
<td>102</td>
<td>355</td>
<td>361</td>
<td>1746</td>
<td>257</td>
</tr>
<tr>
<td>1byob</td>
<td>1nin</td>
<td>99</td>
<td>105</td>
<td>350</td>
<td>376</td>
<td>1679</td>
<td>231</td>
</tr>
<tr>
<td>1byob</td>
<td>1kdi</td>
<td>99</td>
<td>102</td>
<td>350</td>
<td>361</td>
<td>1478</td>
<td>264</td>
</tr>
<tr>
<td>2plia</td>
<td>2b3ia</td>
<td>97</td>
<td>97</td>
<td>340</td>
<td>341</td>
<td>1188</td>
<td>226</td>
</tr>
<tr>
<td>2b3ia</td>
<td>2pcy</td>
<td>97</td>
<td>99</td>
<td>341</td>
<td>357</td>
<td>1053</td>
<td>265</td>
</tr>
<tr>
<td>2plia</td>
<td>2pcy</td>
<td>97</td>
<td>98</td>
<td>341</td>
<td>367</td>
<td>1010</td>
<td>293</td>
</tr>
</tbody>
</table>

Alignments of Proteins from two different folds: Flavodoxin-like and Cupredoxins

<table>
<thead>
<tr>
<th>protein (A)</th>
<th>protein (B)</th>
<th># residues (A)</th>
<th># residues (B)</th>
<th># contacts (A)</th>
<th># contacts (B)</th>
<th>time (s)</th>
<th># aligned contacts</th>
</tr>
</thead>
<tbody>
<tr>
<td>1b00a</td>
<td>1bawa</td>
<td>122</td>
<td>105</td>
<td>441</td>
<td>387</td>
<td>3588</td>
<td>109</td>
</tr>
<tr>
<td>1b00a</td>
<td>1byoae</td>
<td>122</td>
<td>99</td>
<td>441</td>
<td>355</td>
<td>3609</td>
<td>92</td>
</tr>
<tr>
<td>1b00a</td>
<td>1dpbs</td>
<td>122</td>
<td>154</td>
<td>441</td>
<td>586</td>
<td>3384</td>
<td>125</td>
</tr>
<tr>
<td>1nat</td>
<td>1amk</td>
<td>119</td>
<td>250</td>
<td>435</td>
<td>933</td>
<td>2963</td>
<td>166</td>
</tr>
<tr>
<td>1nat</td>
<td>1dpbs</td>
<td>119</td>
<td>154</td>
<td>435</td>
<td>586</td>
<td>5163</td>
<td>136</td>
</tr>
<tr>
<td>1qmpc</td>
<td>2pcy</td>
<td>125</td>
<td>99</td>
<td>452</td>
<td>357</td>
<td>5778</td>
<td>107</td>
</tr>
<tr>
<td>1qmpc</td>
<td>8tmia</td>
<td>125</td>
<td>247</td>
<td>463</td>
<td>930</td>
<td>5516</td>
<td>169</td>
</tr>
<tr>
<td>4tmya</td>
<td>1bawa</td>
<td>118</td>
<td>105</td>
<td>446</td>
<td>387</td>
<td>3306</td>
<td>110</td>
</tr>
<tr>
<td>4tmya</td>
<td>1amk</td>
<td>118</td>
<td>250</td>
<td>446</td>
<td>933</td>
<td>2952</td>
<td>154</td>
</tr>
<tr>
<td>4tmya</td>
<td>1dpbs</td>
<td>118</td>
<td>154</td>
<td>446</td>
<td>587</td>
<td>3242</td>
<td>121</td>
</tr>
<tr>
<td>1bawa</td>
<td>1aw2b</td>
<td>105</td>
<td>254</td>
<td>387</td>
<td>966</td>
<td>2954</td>
<td>103</td>
</tr>
<tr>
<td>1bawa</td>
<td>1b9ba</td>
<td>105</td>
<td>252</td>
<td>387</td>
<td>953</td>
<td>2745</td>
<td>120</td>
</tr>
<tr>
<td>1bawa</td>
<td>1dpbs</td>
<td>105</td>
<td>154</td>
<td>387</td>
<td>586</td>
<td>2543</td>
<td>114</td>
</tr>
</tbody>
</table>
Questions?