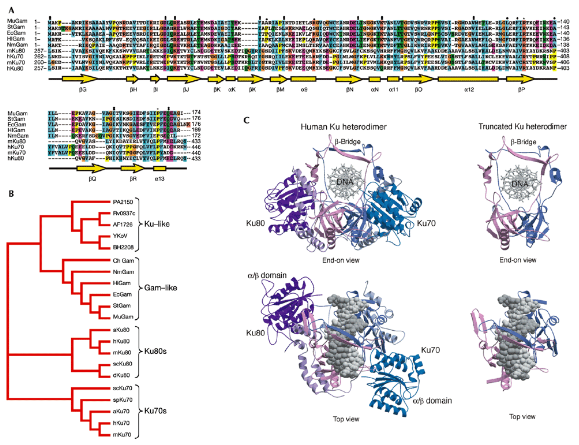
Kpax – Protein Structure Alignment

Dave Ritchie

Team Orpailleur
Inria Nancy – Grand Est

Outline

Protein Sequences and Structures



Source: "The Gam protein of bacteriophage Mu is an orthologue of eukaryotic Ku", F.A. di Fagagna *et al.*, EMBO Reports (2003), 4, 47–52

Comparing Two Strings

Q. Suppose we have two strings, e.g. `EXPONENTIAL` and `POLYNOMIAL`. How do we measure their similarity?

A1. In information theory, the *edit distance* measures the cost of transforming one string into another using one-character edits

A2. Match 3 letters

```
␣POLYNOMIAL
       |||
EXPONENTIAL
```

and then give a score for each pair...

Q. Suppose gaps are allowed. What is the best possible alignment?

A. How about

```
--POLYNOM-IAL
  ||  |   |||
EXPO--NENTIAL
```

or

```
--POLYNOMIAL
  ||  |  |||
EXPONEN-TIAL
```

?

Q. Which is better ?

A1. The second one? (6 matches + 3 gaps *v's* 6 matches + 5 gaps)
A2. ... It depends on the score for each pair and the penalty for a gap

Dynamic Programming

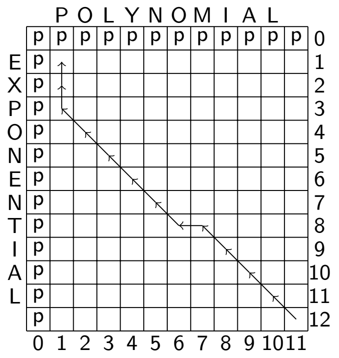
Dynamic programming (DP) is a method of dividing a problem into smaller sub-problems. It was first described by Richard Bellman in the 1940s. But instead of using recursion, it uses a table ("memoisation" in 1940s language).

- Goal: find similarity $E(n,m)$ between two strings: $x[1{:}n]$ and $y[1{:}m]$
- Sub-goal: find $E(i,j)$ between two <u>*prefixes:*</u> $x[1{:}i]$ and $y[1{:}j]$
- Observation: the best alignment must end on $\boxed{\begin{matrix}x[i]\\y[j]\end{matrix}}$ or $\boxed{\begin{matrix}x[i]\\-\end{matrix}}$ or $\boxed{\begin{matrix}-\\y[j]\end{matrix}}$
- Method: build similarity table with scores $S(i,j)$ and penalties $P(i)$:

$$E(i,j) = \max \begin{cases} E(i-1,j-1) + S(i,j) \\ E(i,j-1) - P(i) \\ E(i-1,j) - P(j) \end{cases}$$

- Then, "trace back" from $E(n,m)$ to $E(1,1)$ to extract the alignment

Back-Tracking Through The DP Scoring Table



- This gives the desired optimal alignment

```
--POLYNOMIAL
  ||  |  |||
EXPONEN-TIAL
```

3D Least-Squares Fitting

- Least-squares fitting finds the 3D rotation/translation matrix $\underline{M}$ that minimises the sum of squared distances:

$$F = \sum_{i=1}^{N} (\underline{x}_i^A - \underline{M}.\underline{x}_i^B)^2$$

- For proteins, the $\underline{x}_i$ are normally C_α atom coordinates
- The translational part is easy – shift centres of mass to the origin
- The rotation can be found using eigenvector or quaternion methods
- The residual error (RMSD) is then given by

$$RMSD = \sqrt{\frac{1}{N}\sum_{i=1}^{N} (\underline{x}_i^A - \underline{M}.\underline{x}_i^B)^2}$$

- So, given list of aligned C_α's, we can fit optimally to some RMSD

So, What's The Problem?

- DP is "perfect" for 1D string matching
- Least-squares fitting is "perfect" for 3D superposition

BUT

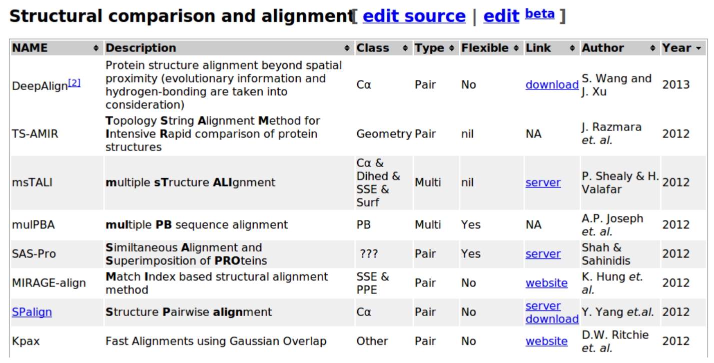
- Proteins are not made of 1D symbols or 3D points. They are made of complex 3D chemical components (amino acid residues). It is difficult to write a good scoring function to compare residues...
- Similar 1D protein sub-sequences can have different 3D shapes (α-helices, β-strands), i.e. global environment can affect local shape. We don't know *a priori* the right 1D pairings for 3D fitting...
- Proteins are globally flexible. Even if many local 1D regions "match", not all of them might simultaneously superpose well in 3D space...

ADDITIONALLY!

- Proteins can contain multiple repeats and/or transpositions...

Over 100 Structure Alignment Algorithms in 25 Years

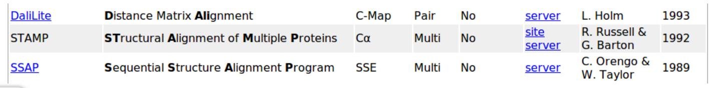
- http://en.wikipedia.org/wiki/Structural_alignment_software



Structural comparison and alignment[edit source | edit beta]

NAME	Description	Class	Type	Flexible	Link	Author	Year
DeepAlign[2]	Protein structure alignment beyond spatial proximity (evolutionary information and hydrogen-bonding are taken into consideration)	Cα	Pair	No	download	S. Wang and J. Xu	2013
TS-AMIR	**T**opology **S**tring **A**lignment **M**ethod for **I**ntensive **R**apid comparison of protein structures	Geometry	Pair	nil	NA	J. Razmara *et. al.*	2012
msTALI	**m**ultiple **sT**ructure **ALI**gnment	Cα & Dihed & SSE & Surf	Multi	nil	server	P. Shealy & H. Valafar	2012
mulPBA	**mul**tiple **PB** sequence alignment	PB	Multi	Yes	NA	A.P. Joseph *et. al.*	2012
SAS-Pro	**S**imiltaneous **A**lignment and **S**uperimposition of **PRO**teins	???	Pair	Yes	server	Shah & Sahinidis	2012
MIRAGE-align	**M**atch **I**ndex based structural alignment method	SSE & PPE	Pair	No	website	K. Hung *et. al.*	2012
SPalign	**S**tructure **P**airwise **align**ment	Cα	Pair	No	server download	Y. Yang *et.al.*	2012
Kpax	Fast Alignments using Gaussian Overlap	Other	Pair	No	website	D.W. Ritchie *et. al.*	2012

90 more...



DaliLite	**D**istance Matrix **Ali**gnment	C-Map	Pair	No	server	L. Holm	1993
STAMP	**ST**ructural **A**lignment of **M**ultiple **P**roteins	Cα	Multi	No	site server	R. Russell & G. Barton	1992
SSAP	**S**equential **S**tructure **A**lignment **P**rogram	SSE	Multi	No	server	C. Orengo & W. Taylor	1989

Quick List of Structural Alignment Approaches

- "elastic" Gaussian scoring
- "double dynamic programming" on C_α distance matrices
- triples or higher fragments (8-tuples) of C_α atoms
- backbone C_α vectors
- backbone torsion angles
- secondary structure elements
- geometric hashing
- Voronoi tessellations
- structural alphabets
- Lagrangian contact map optimisation
- eigenvector analysis of distance matrices
- Fourier correlations
- Gaussian fragments
- ...

Introducing Kpax

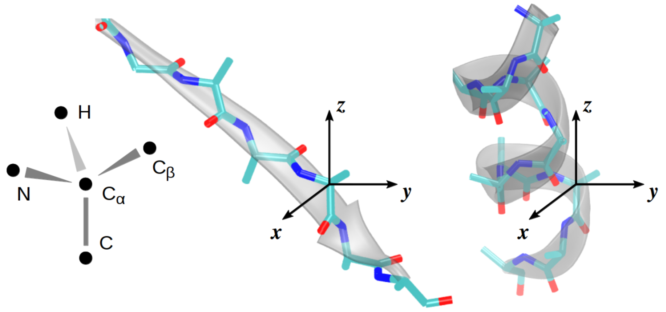
http://kpax.loria.fr/

- Dynamic programming with Gaussian scores
- Uses <u>NO</u> sequence similarity <u>OR</u> secondary structure information

- Very fast database search (CATH, SCOP, Pfam, ..., user-defined)
- Rigid *and* flexible structural alignments
- Multiple flexible alignments coming soon...

Defining Local Coordinate Frames

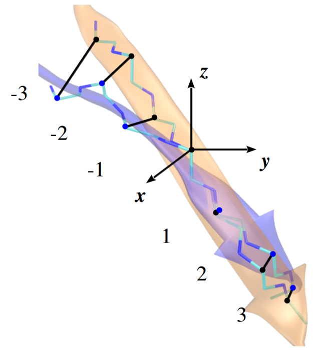
- All C_α atoms have highly conserved tetrahedral geometry
  - Exploit this to define a "canonical" C_α–C–N orientation
  - e.g. put C_α at origin; C on -ve z axis; N in +ve xz plane



- Now, <u>*ALL*</u> α-helices and β-strands look the same at the origin

Comparing Structural Fragments

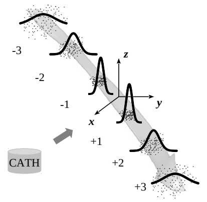
- In the canonical frame, similar structures have similar distances between their up-stream and down-stream $C_\alpha$atoms:



- But how to combine all the distances into a single score?

Representing Local Geometry as a Product of Gaussians

- Calculate Gaussian distribution of all C_α atoms in CATH



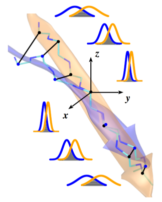
- Gives Gaussian width σ_k for each up-stream and down-stream C_α
- Then, represent residue i as a product of Gaussians:

$$\psi_i = \phi_i^{-1}(\underline{x}_{i-1})\phi_i^{+1}(\underline{x}_{i+1}) \dots \phi_i^{-n}(\underline{x}_{i-n})\phi_i^{+n}(\underline{x}_{i+n})$$

- each individual Gaussian function has the form:

$$\phi_i^k(\underline{x}_{i+k}) = N_k e^{-\beta_k r_k^2/2\sigma_k^2}$$

Calculating a Per-Residue Local Similarity Score



- Calculate the local-frame similarity, K_{ij}^{local}, as an overlap integral

$$K_{ij}^{\text{local}} = \int \psi_i \psi_j \, \mathrm{d}\underline{x}_{-n\dots+n}.$$

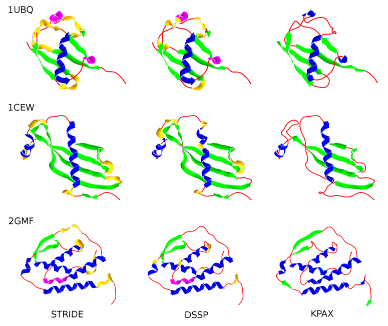
- With products of Gaussians, this reduces to a simple sum

$$K_{ij}^{\text{local}} = e^{-\sum_{k=-n}^{n} \beta_k R_{i+k,j+k}^2/4\sigma_k^2},$$

- In identical α-helices, β-strands, and even loops, $K_{ij}^{\text{local}} = 1$.

Detecting Secondary Structure Elements

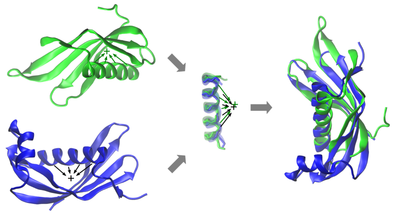
By sliding a model α-helix and β-strand along a structure, Kpax detects its secondary structure elements (SSEs) automatically (it does not distinguish π or 3_{10} helices or detect β-turns). Here are some examples:



- Nice, but how to match correctly a short α-helix with a longer one?

The Spatial Similarity Score

If two similar protein domains are superposed, their centres of mass (COM) will be close together. Therefore, *in the local coordinate frame*, well-aligned residues will "see" the COM in similar positions in space (but consecutive residues will see the COM in quite different positions).



From the COM direction vector, we get a *spatial similarity score*

$$K_{ij}^{\text{spatial}} = e^{-\sum_{k=-n}^{n} \beta_k R_{i+k,j+k}^2 / 4\tau_k^2}$$

The Kpax Structure Alignment Algorithm

- The "local" and "spatial" scores give a kind of "1D preview" of how two proteins might be aligned without actually moving them

$$K_{ij}^{1D} = (K_{ij}^{\text{local}} + K_{ij}^{\text{spatial}})/2$$

- Once the proteins are superposed, we can calculate real 3D Gaussian overlap scores for every pair of residues:

$$G_{ij}^{3D} = e^{-R_{ij}^2/4\tau^2}$$

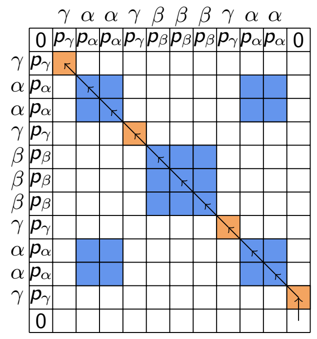
This leads to the following algorithm:

1. Set per-residue gap penalties according to SSE types
2. Apply DP to the K-scores to get the first correspondence
3. Fit some/all fragments by least-squares and superpose
4. Calculate 3D G-scores between close pairs of residues
5. Apply DP to the G-scores to get a new correspondence

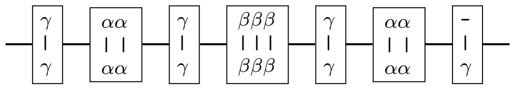
Generating Fitting Fragments From The K-Scores

Blocks of high K-scores arise when SSEs detect each other:

(γ means "loop")



The alignment:

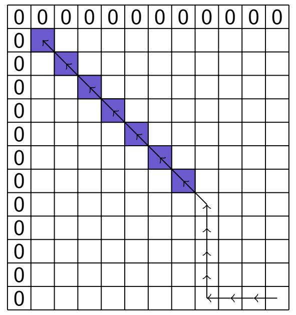


Next, use the pairs within each blue block as fitting candidates...

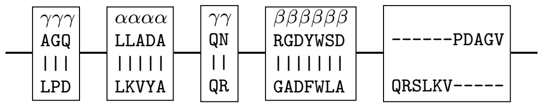
Scoring Trial Superpositions Using G-Scores

Evaluate each trial superposition using real 3D coordinates (G-scores):

(no gap penalties)

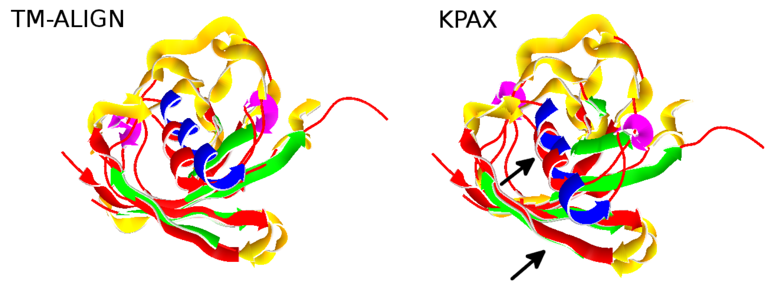


The aligned residues:


γγγ AGQ | | | LPD
αααα LLADA | | | | | LKVYA
γγ QN | | QR
ββββββ RGDYWSD | | | | | | | GADFWLA
-------PDAGV QRSLKV-----

Example 1: Aligning Ubiquitin and Ferrodoxin I

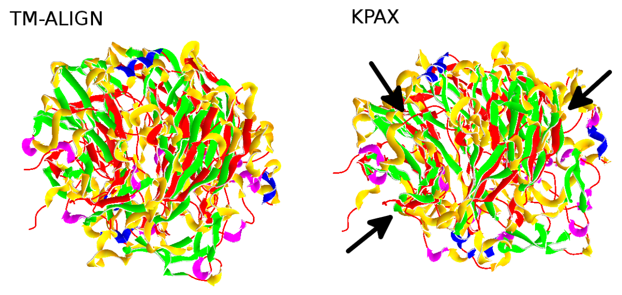
- TM-Align: 63 residues aligned, C_α RMSD = 2.6Å
- Kpax-1.0: 45 residues aligned, C_α RMSD = 2.0Å



- My claim: Kpax gives a tighter alignment than TM-Align

Example 2: methyl dehydroxygenase / galactose oxidase

- The SCOP domains d4aaha_ and d1gofa3
  - TM-Align: 336 residues aligned, C_αRMSD = 5.4Å
  - Kpax-1.0: 178 residues aligned, C_αRMSD = 3.9Å
  - Difference: 11.6 Å RMSD



- ... I believe the Kpax alignment is correct!

Building and Searching Structural Databases

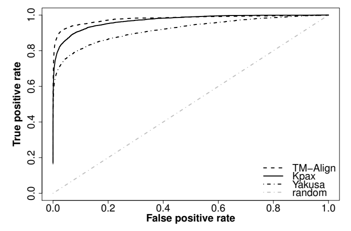
Before calculating an alignment, Kpax pre-processes each protein separately. The pre-processed data can be stored to make a structural database... NB. It takes more time to read a PDB file than to do an alignment !

- Shift the protein to put its COM at the world origin
- Calculate 6 local and 6 spatial "atom" coordinates per residue
- Determine the SSE type of each residue
- Save this data and original PDB coords as binary "blobs"
- Use Linux "memory mapping" to read a binary database
- Use Posix threads to do all calculations in parallel

Result: Searching 11,000 CATH structures takes about 4 seconds...

ROC-Plot Comparison with TM-Align and Yakusa

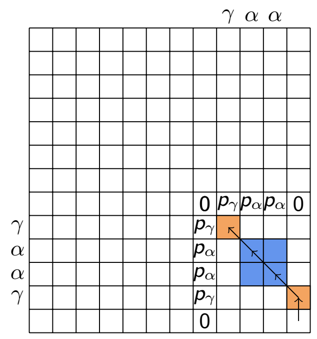
- We selected 213 CATH families, each having $\geq$ 10 members
- Searched CATH database with one structure from each family
  - TP (true pos) when [C.A.T.H] code of query matches database
  - FP (false pos) when query matches some other CATH code



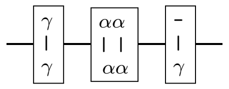
- TM-align AUC = 0.976, Kpax AUC = 0.966, Yakusa AUC = 0.915
- TM-align 46 h; Yakusa 2.2 h; Kpax 0.3h (i.e. Kpax is 150x / 6x faster)

Flexible Alignment – Finding More Fitting Fragments

For flexible alignment, just fill in the remaining boxes:

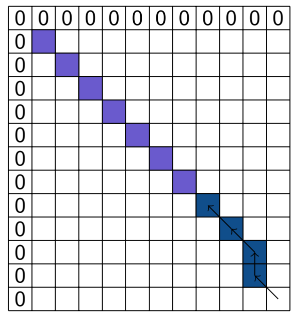


Candidate fitting fragments:

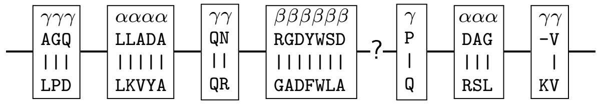


The Final Flexible Alignment

After a further round of DP, we get the final "flexible" alignment:



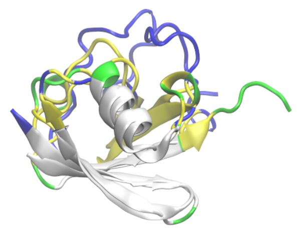
NB. the 3D structure may have discontinuities between the fitted segments:

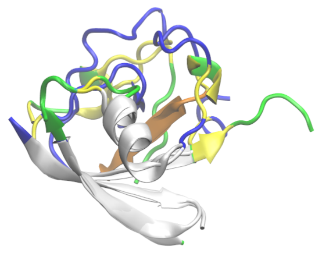


$\gamma\gamma\gamma$ AGQ ||| LPD
$\alpha\alpha\alpha\alpha$ LLADA ||||| LKVYA
$\gamma\gamma$ QN || QR
$\beta\beta\beta\beta\beta\beta$ RGDYWSD ||||||| GADFWLA
?
γ P | Q
$\alpha\alpha\alpha$ DAG ||| RSL
$\gamma\gamma$ -V | KV

Example 1 Revisited – Flexible Kpax

- ubiquitin (green) and ferrodoxin I (blue)
  - Kpax-1.0: 45 residues aligned, C_α RMSD = 2.0Å (rigid)
  - Kpax-2.2: 57 residues aligned, C_α RMSD = 2.8Å (rigid)
  - Kpax-2.2: 56 residues aligned, C_α RMSD = 2.2Å (flexible)

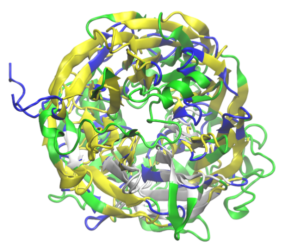


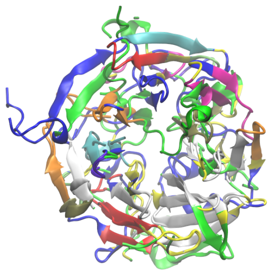


- Rigid superposition: white = anchor; yellow = aligned
- Flex superposition: (re-fitted anchors, new segment in orange)

Example 2 Revisited – Flexible Kpax

- d4aaha_ and d1gofa3 (looking directly at the β-propeller)
  - Kpax-1.0: 178 residues aligned, C_αRMSD = 3.9Å (rigid)
  - Kpax-2.2: 218 residues aligned, C_αRMSD = 3.2Å (rigid)
  - Kpax-2.2: 260 residues aligned, C_αRMSD = 1.7Å (flexible)

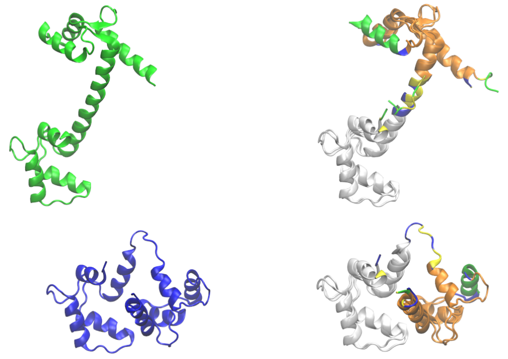




- Rigid superposition: white = anchor; yellow = aligned
- Flex superposition: 20 segments (anchor in white)

Aligning Human and Fly Calmodulin

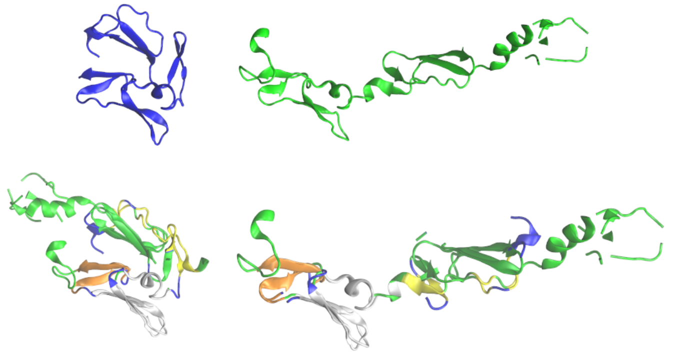
- Calmodulin (Ca-binding protein) is found in all eukaryotic cells
  - Green = 1CLL (human: *homo sapiens*)
  - Blue = 2BBM (fly: *dropsophila melanogaster*)



- Both superpositions have 4 segments (137 residues, 1.7 Å RMSD)

Human Blood Factor and a Parasite Surface Protein

- Green = 1DAN (human blood coagulation factor VIIA)
- Blue = 1B9W (*Plasmodium cynomolgi* merozoite surface protein)



- 4/4 segments, 70/77 residues, 18/19 identities, 1.8/1.9 Å RMSD

Multiple Flexible Alignments and Modeling

Multiple Structural Alignments (unpublished)

- Kpax 4.0 uses "pile-up" method for multiple alignments
  - Choose "centre" or "pivot" structure; fit the rest on to it
  - Also works with flexible alignments to the pivot

Automatic Homology Modeling Pipeline (unpublished)

- Use a protein sequence as the search query...
- ... find the structure with the closest sequence and use as "pivot"
- Perform a structural search using pivot as query...
- ... make multiple structural alignment of closest hits
- Generate Modeler command script automatically :-)

Worked Example – Cyp450

- kpax -db=cyp450 -model -show=20 my_secret_cyp.fasta

Conclusion and Future Prospects

Conclusions

- Tight high quality structural alignments
- Fast structural databases searches
- Flexible alignments now possible
- Multiple alignments now possible
- All-versus-all structural comparisons now possible

Future Prospects

- Better structural alignments → better holomology models...
- Should help study evolutionary relationships at structural level ...

Thank You!

Acknowledgments

Anisah Ghoorah
Lazaros Mavridis
Vishwesh Venkatraman
April Chung
Niruba Thiagarajan

Program and paper:

http://kpax.loria.fr/

Kpax Demo – Basic Operations

- Download Kpax from: http://kpax.loria.fr/download-3.2-beta/
- Example pairs of structures: http://hex.loria.fr/emmsb/kpax_examples/
- Aligning two structures
- Viewing the results in Hex and VMD
- Performing flexible structural alignments
- Building and searching a structural database
- Performing multiple structural alignments
- Viewing multiple alignments in Hex/VMD
- ...
- Ask me!
- Disclaimer: Kpax is not “commercial” software!

Practical Activities – 1

Downloading the data

- Download the API-A sequence from: http://hex.loria.fr/emmsb/t40.tgz
  - t40_c.fasta (API-A)
- Download the CATH database from: http://hex.loria.fr/emmsb/cath/
  - CathDomainPdb.S35.v3_4_0.tgz (260 Mb of compressed PDB files)
  - CathDomainList.gz (1.3 Mb file of CATH codes)
  - build_cath.sh (shell script to build a Kpax database)

Building a Kpax database

- Unzip the two zip files (use gunzip)
- Edit the script to have the correct path to the data (CATH_ROOT=???)
- Run the script to build that database (takes a few minutes)
- Run kpax with -help option to show all the parameters and options; try:
  - kpax -db=cath -list

Practical Activities – 2

Searching a database and visualising the results

- Try searching the database with t40_c.pdb as the query
- Go to the results folder (kpax_results/t40_c) and look at the results files
- Use Hex or VMD to visualise the results (.mac for Hex and .tcl for VMD)
- Do you agree with the superpositions?

Making a MSA for homology modeling

- Run kpax to make a multiple structure alignment from the seed sequence
- The command for this is of the form:
  - kpax -db=cath -model -top=24 t40_c.fasta
- cd to the results folder (kpax_results/2qn4A00) and examine the contents

Making a homology model (optional)

- Run Modeler (the actual command may differ on your machine):
  - /opt/modeller/bin/mod9.13 t40_c_modeller.py
- Use Kpax/Hex to compare the model and real structure (t40_c.pdb)