

Protein Structure Comparison and Structure Patterns - an Algorithmic Approach

Ingvar Eidhammer and Inge Jonassen
Dept. of Informatics,
Univ. of Bergen, Norway

[Ingvar.Eidhammer,Inge.Jonassen]@ii.uib.no

Outline

- Introduction and framework
- Description and representation
- Algorithms
 - Superposition
 - Alternating superposition and alignment
 - Double dynamic programming
 - Geometric hashing
 - Comparison by clustering
 - Multiple structure comparison
 - Local structure patterns
- Open problems and conclusions

Organisation of tutorial

- Part I Introduction Inge
- Part II Algorithms for pairwise structure comparison Ingvar
- Part III Comparison by clustering Ingvar
- Part IV Algorithms for multiple structure comparison and motif discovery, conclusions Inge

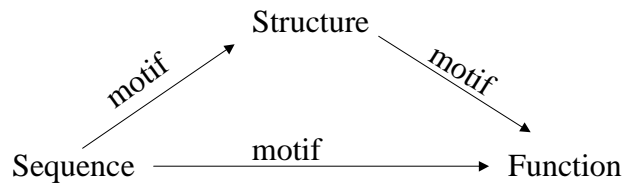
3

Part I - Introduction

- Motivation, motifs in protein analysis
- Protein comparison and classification
- Terminology
- Framework
 - structure description
 - alignment/equivalence
 - scoring
 - algorithm
- Example method
 - Alternating superposition and alignment

4

Motifs in Protein Analysis



Sequence → structure motifs } Sequence motifs
Sequence → function motifs }
Structure → function motifs Structure motifs

5

Sequence structure/function motifs

- Pattern shared by a the sequences of a set of proteins with similar functions and/or structures.
- Match to pattern suggests structure/function for protein.
- Databases:
 - PFAM, PRINTS, PROSITE, PRODOM, BLOCKS, SMART, ...
- Integrated database:
 - InterPro

6

Example sequence motif

```

          xxx
        V  x
       x   x
       x   x
       x   x
       C   H
       x \ / x
       x   Zn x
       x / \ x
       C   H
    xxxxx      xxxxxxx
    
```

C-x(2,4)-C-x(3)-[ILVMFYWC]-x(8)-H-x(3,5)-H

7

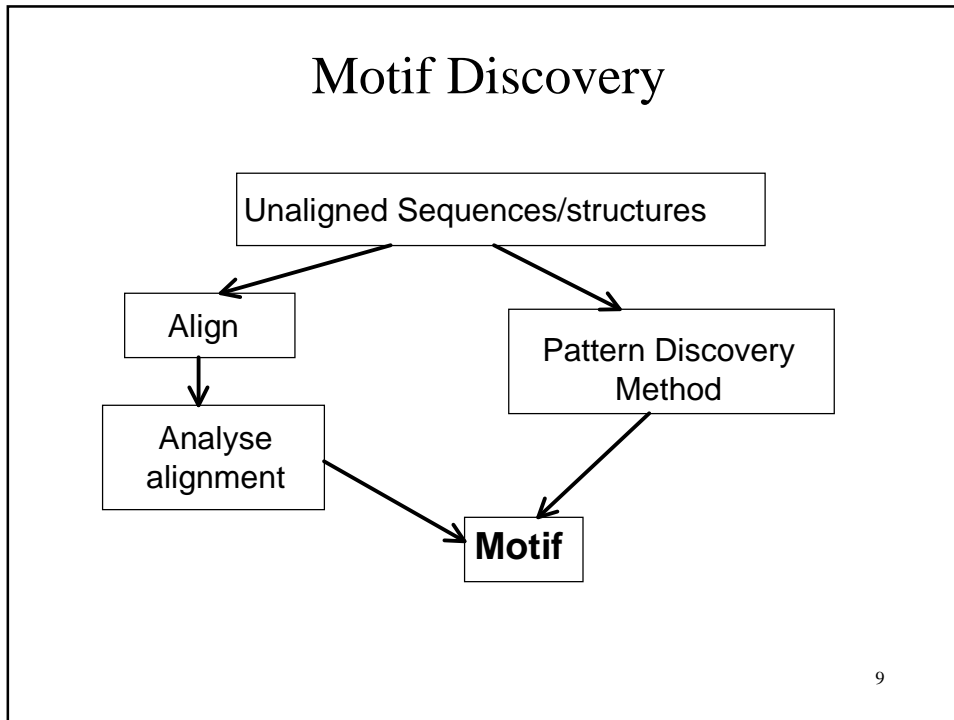
Sequence alignment

Alignment of chromo domains

Classical chromo domains		Chromo Shadow domains	
DmPC	19 84	dmprllylaa	140 205
Mom03	5 70	ssvqavvaac	147 212
CeY082	1 67	madsceltl	114 179
DmHP1_A	17 82	aeaeleeval	110 175
DmHP1_B	17 82	aeaeleeval	104 169
HuHP1_A	13 78	sseedleeval	104 170
Mom01_A	14 79	leeeleeval	129 193
Mom02_A	13 78	eeaeleeval	260 328
PCHET1_A	4 69	sgseleeval	
PCHET2_A	6 72	vpalee	
SNPA126	(49 219)	espgde	
SpSH16_A	74 143	eeeee	
PF0131C	(78 200)	eeeee	
CeT9A58	17 84	egksei	
DmSuv3-9	212 278	krppge	
HuMCH4	(250 448)	skrnly	
CFTRW	81 143	epel	
FoSKPY	1229 1236	eisg	
MoCHD1_A	263 362	qpde	
CeVK9A3	(2 133)		
ScVE24_A	188 257	kts	
MoCH1_B	380 450	ddl	
ScVE24_B	278 350	lde	
MgGRH	1265 1332	tge	
MgMAGGV	1130 1199	evog	
CeZRH12	39 136	tads	
DmHP1_B	140 205	stgr	
DmHP1_B	147 212	gtgr	
HuHP1_B	114 179	argr	
Mom01_B	110 175	prgr	
Mom02_B	104 169	prgr	
PCHET1_B	105 170	lmg	
PCHET2_B	129 193	vsd	
SpSW16_B	260 328	vkq	
consensus		%- % #E+#H	
predict		HHHHH E	
rule		1.....10.....20.....30.....40.....50.....60.....70.....	

8

Aasland and Stewart, (1995) Nucl. Acids. Res. 23:3168-3173



Protein structure-function motifs

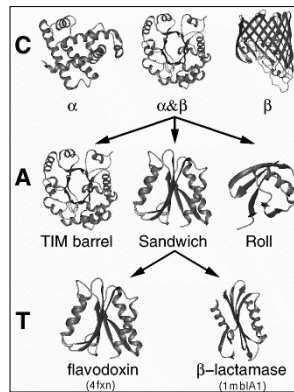
- PROCAT
 - A database of 3D enzyme active site templates
 - Thornton et al
- Fuzzy Functional Forms
 - structural descriptions of functional sites
 - www.geneformatics.com
 - Skolnick and Fetrow, 2000

The slide includes two 3D molecular models. The first model, associated with PROCAT, shows a complex arrangement of atoms in a protein active site. The second model, associated with Fuzzy Functional Forms, shows a simpler, more abstract representation of a functional site with three large spheres connected by lines.

10

Protein Classifications

- SCOP
 - Murzin et al
 - manually made
- CATH
 - Orengo et al
 - partly automated
- FSSP
 - Fold classification based on Structure-Structure alignment of Proteins
 - Fully automatic
 - Holm and Sander



Cartoon illustrating the three highest levels in CATH

11

Why compare structures?

- Learn about
 - structure-function relationships
 - evolution
 - common building blocks - motifs
- Make order out of the universe of protein structures
- Help structure prediction

12

Why is there more than one way?

- Proteins are complex three-dimensional structures.
- They can be described and compared in different ways
 - at different levels with
 - different constraints
- Choice depends on purpose of comparison
 - similarity of secondary structure packing
 - similar topology
- Example structure: 2nad

13

Some terminology

- A protein structure consists of one or more *polypeptide chains*.
- The order of amino acids along one chain is its *primary structure* (and its amino acid sequence).
- Some regions form regular local structure
 - *alpha helices*
 - *beta strands*
 - collectively called *secondary structure elements (SSEs)*
- Regions connecting SSEs are *loops*.
- Description of the type and location of SSEs is a chain's *secondary structure*.

14

Terminology (cont'd)

- Three-dimensional coordinates of the atoms of a chain is its *tertiary structure*.
- *Quarternary structure*: describes the spatial packing of several folded polypeptides
- From CATH:
 - *Class* - composition of SSEs
 - *Architecture* - spatial packing of SSEs
 - *Topology* - includes arrangement of SSEs along the chain
 - (Homology - proteins with common ancestry apparent from sequence)

15

How to compare structures?

- Align
 - require co-linearity of paired-up elements
 - analogous to sequence alignment, but more complex
- Find similar substructures
 - ignore sequence order/topology
- Compare composition in terms of SSEs



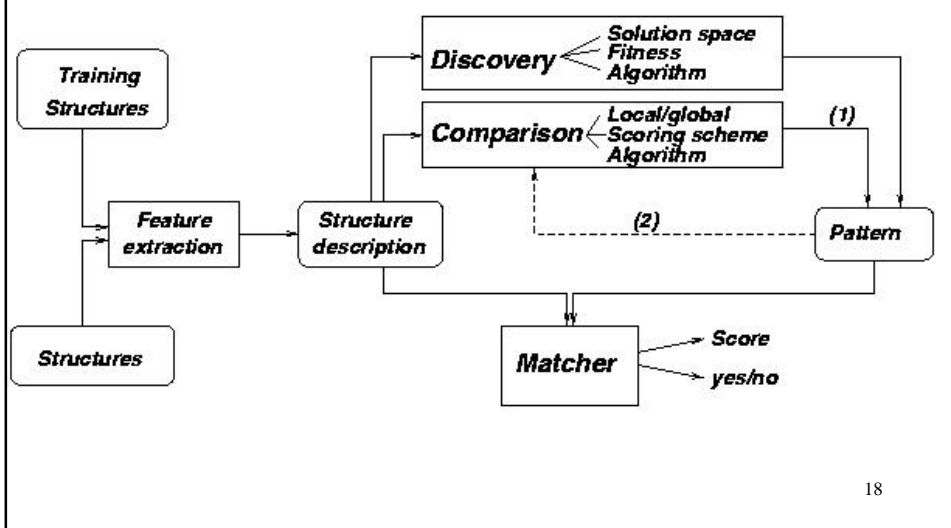
16

Structure Motifs

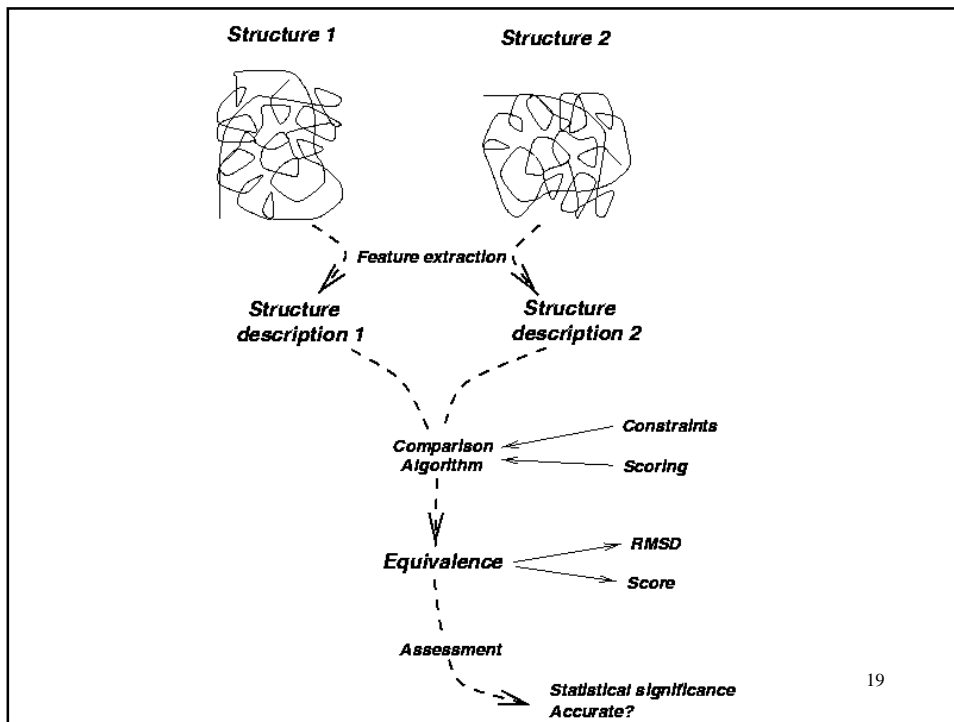
- Common arrangements of SSEs (core motifs)
 - beta barrel
- Binding/active site (residues)
 - catalytic triad,
 - zinc binding residues

17

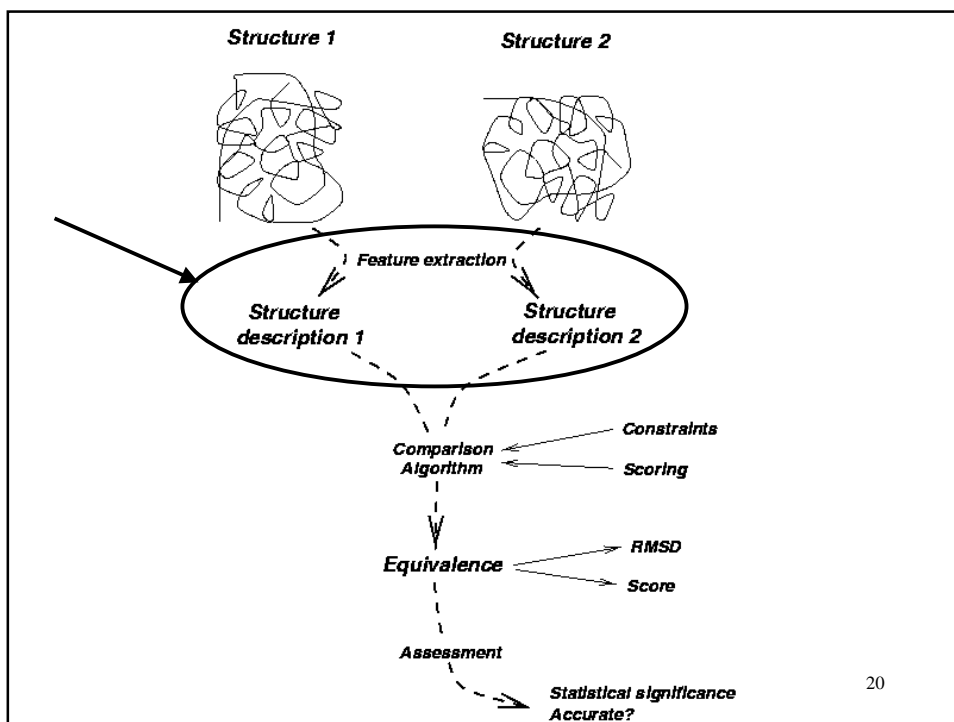
Framework



18



19



20

Structure Description - Levels

- Atom/atom group
- Residue
- Fragment
- Secondary structure element (SSE)

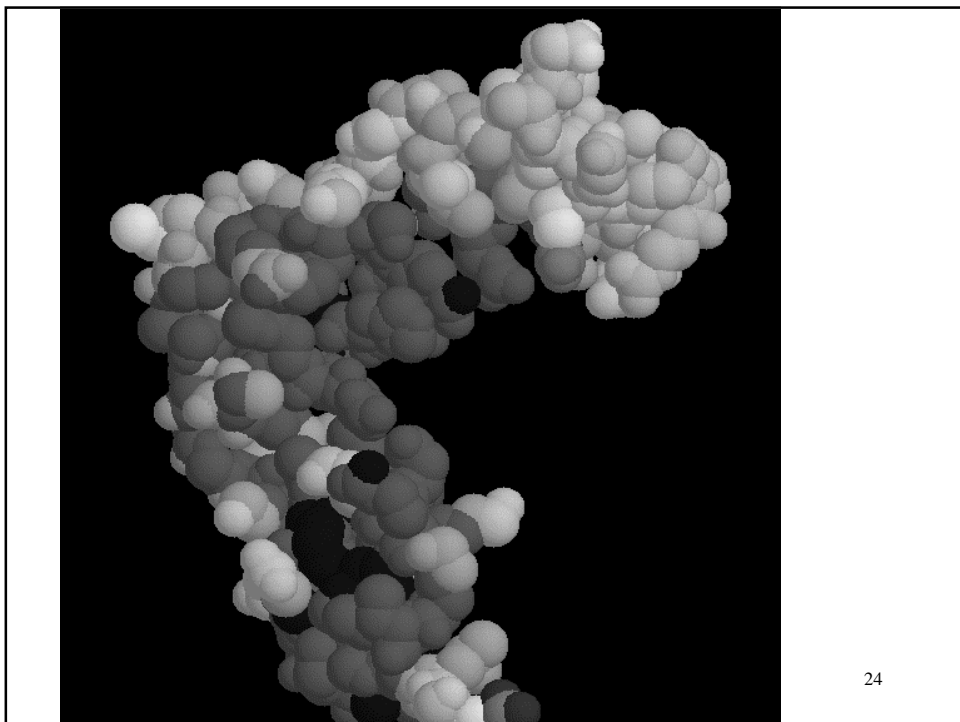
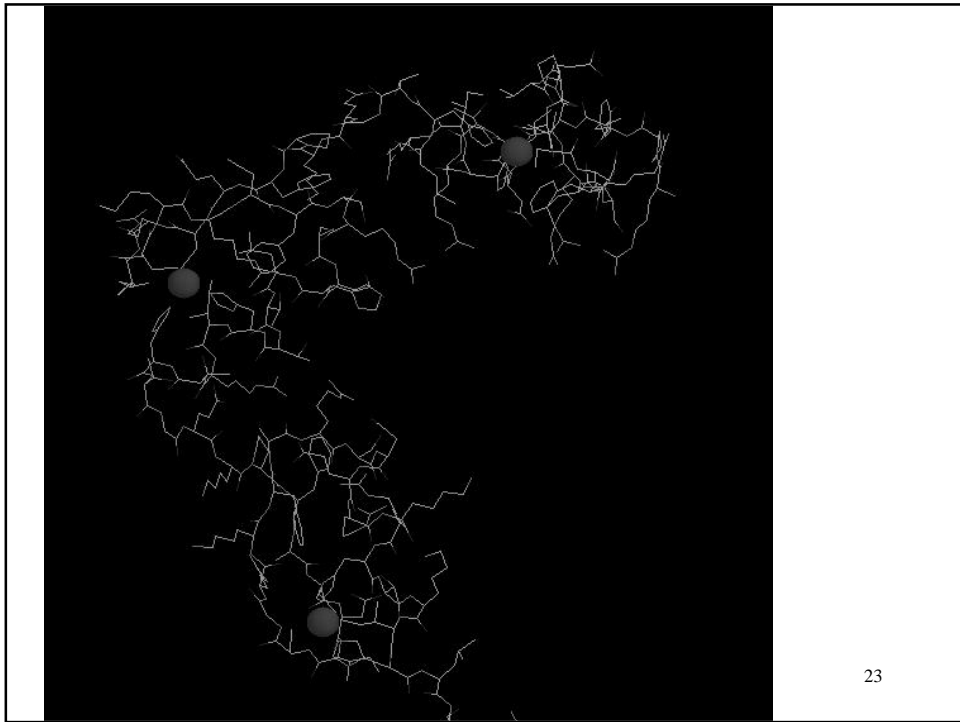
- Structure described by *elements* of the chosen type

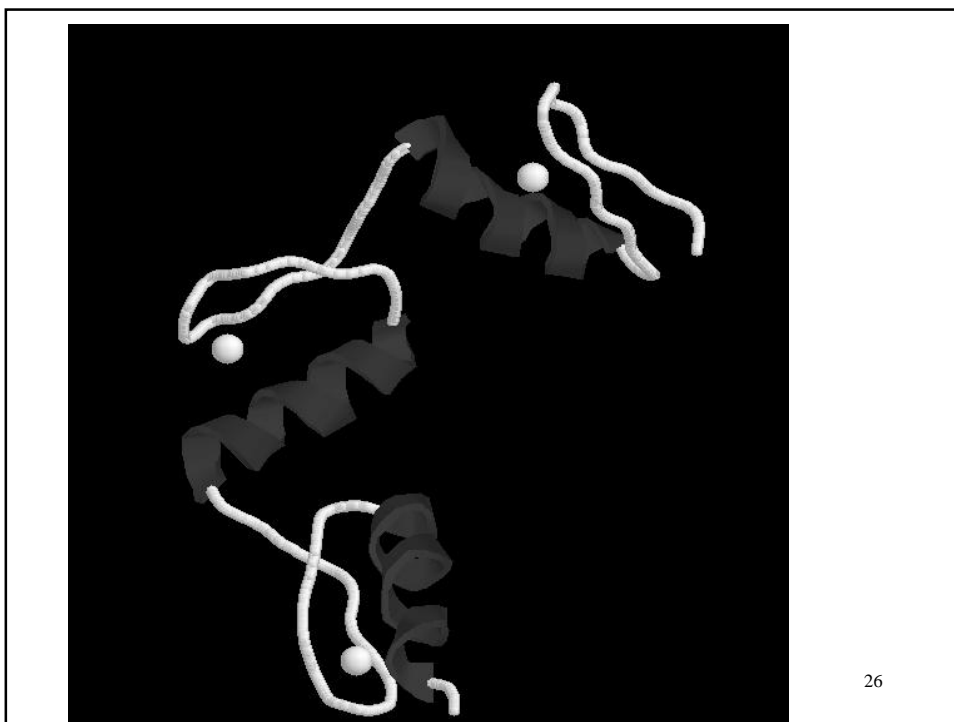
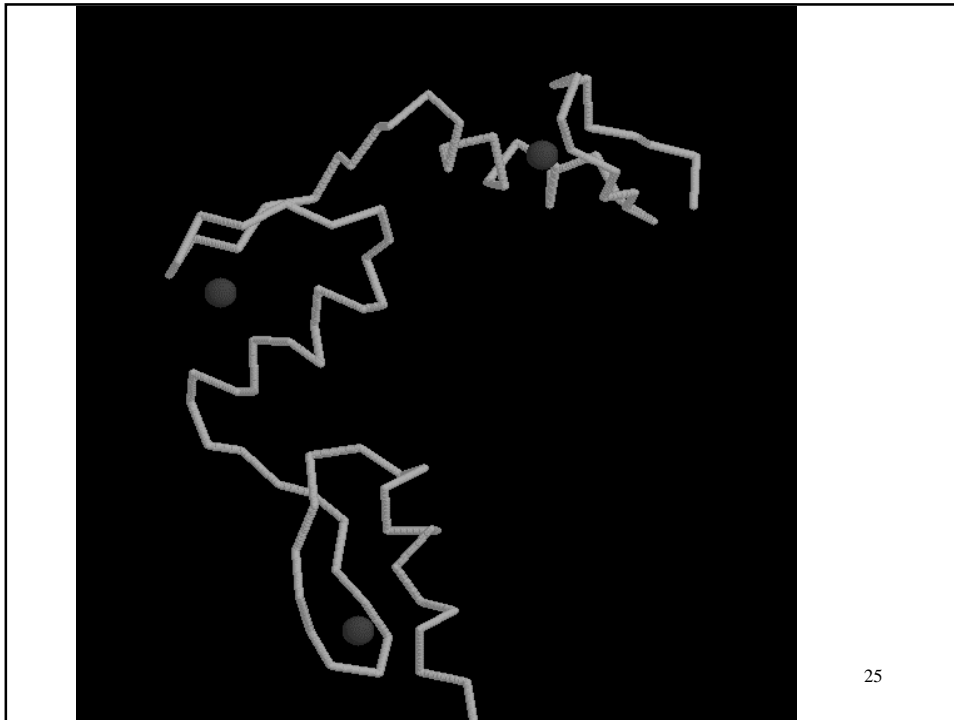
21

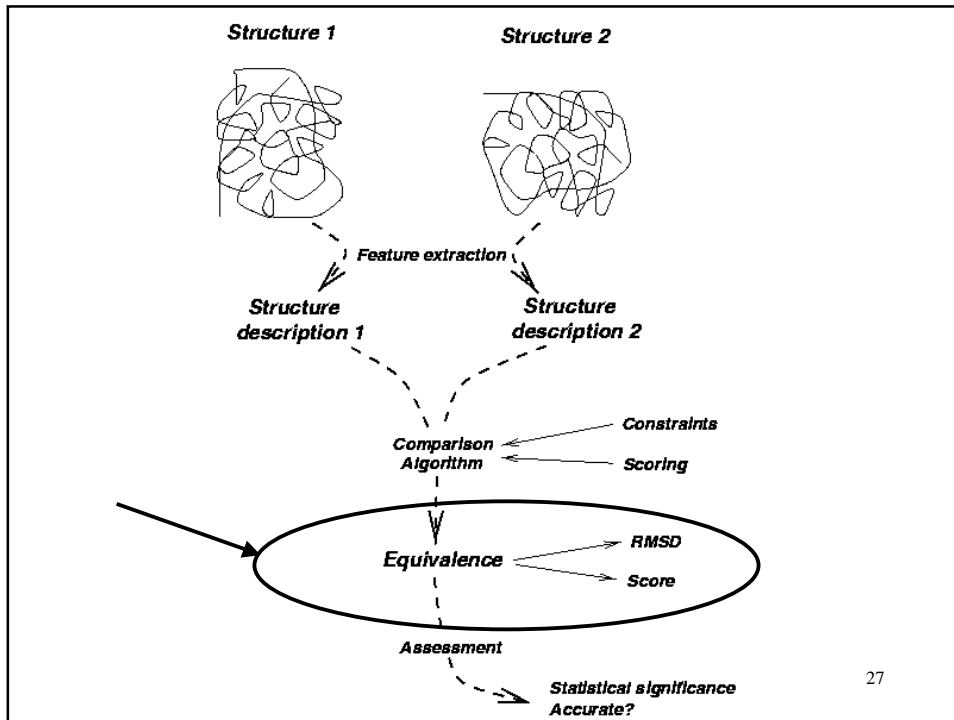
Structure Description - Features

- Geometry/architecture
 - coordinates/relative positions, ...
- Topology
 - sequential order of residues along backbone, ...
- Properties
 - physio-chemical properties of residues, ...

22







Equivalences and Alignments

- Object A with elements a_1, a_2, \dots, a_m
- Object B with elements b_1, b_2, \dots, b_n
- **Equivalence:** $L(A,B) = (a_{i_1}, b_{j_1}), (a_{i_2}, b_{j_2}), \dots, (a_{i_l}, b_{j_l})$
- **Alignment:** co-linear equivalence
 - $i_1 < i_2 < \dots < i_l$ and $j_1 < j_2 < \dots < j_l$
- Can be extended to multiple structures

Scoring Equivalences

- *Coordinate based*
 - defined using a transformation of one structure onto the other:
root mean square deviation - RMSd

29

Coordinate based RMSd

- Define transforms T that consists of a
 - translation
 - rotation
- Structures A and B consist of coordinate sets $A_1 \dots A_m$,
 $B_1 \dots B_m$ - equivalence $(A_1, B_1), \dots, (A_m, B_m)$

$$RMSd(A, B) = \min_T \sqrt{\sum_{i=1}^m (A_i - TB_i)^2}$$

30

Calculating coordinate based RMSd

- Need to find the transform that minimises the equation.
- Least square minimisation to find best combined transform (Muirhead et al 1967; Rao & Rossman 1973)
- Two step method:
 - Relocate origin to center of mass for *A* and *B*
 - Find best rotation:
 - Stepwise search of rotational space (Remington and Matthews, 1978)
 - Least Squares minimization (McLachlan, 1972)
 - Matrix methods
 - singular value decomposition (McLachlan, 1979)
 - Langrangian multipliers (Kabsch 1976, 1978)

31

Scoring Equivalences

- *Coordinate* based
 - defined using a transformation of one structure onto the other:
root mean square deviation - RMSd
- Similarity of *properties* between equivalenced elements
 - conserved/similar amino acid
- Similarity of *relations* between pairs of equivalenced elements
 - similar distances, internal RMSd

32

Distance based RMSD

- No need to transform.
- Definition:

$$RMSd_D(A, B) = \frac{1}{m} \sqrt{\sum_{i=1}^m \sum_{j=1}^m (d_{ij}^A - d_{ij}^B)^2}$$

where

– d_{ij}^A is the distance between atoms i and j in structure A

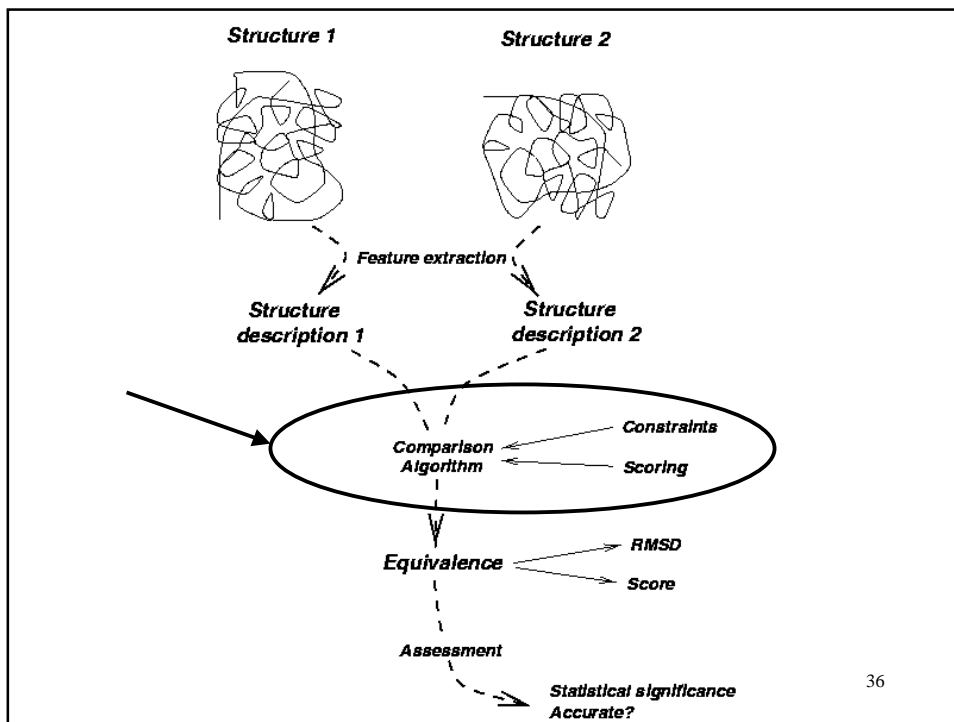
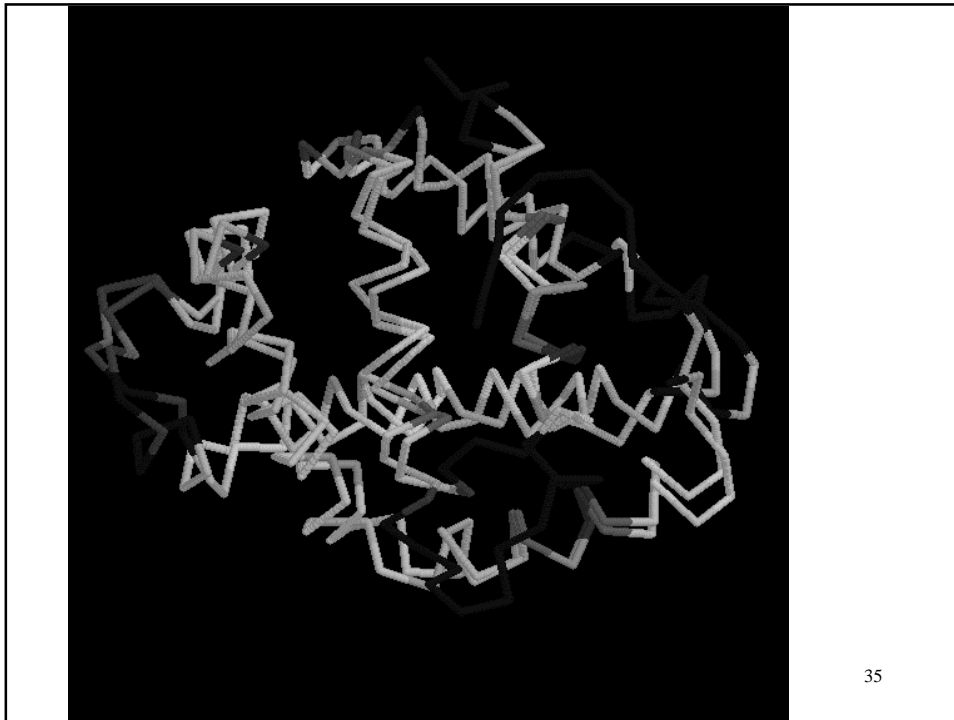
- Linear relationship between RMSD and $RMSD_D$
- $RMSD_D$ cannot distinguish between mirror images.

33



by SAP

entity 34



Comparison Algorithms

- Compare two structure descriptions - find equivalence (alignment) having high score.
- Computationally expensive

37

Dynamic Programming - DP

- **Sequences**
 S, T can be aligned optimally in time $O(|S||T|)$
- Next slides gives a summary of sequence alignment by DP.

38

Edit distance

- The edit distance between $S1$ and $S2$ is the *minimum* number of deletions, insertions and substitutions needed to transform $S1$ to $S2$.
- $|S1|=m, |S2|=n$
- $D(i,j)$ is the edit distance between $S1[1..i]$ and $S2[1..j]$.
- Recurrence relation:
 - basis:

$$D(0, j) = j \text{ for all } 0 \leq j \leq m \quad \textit{Insert } j \textit{ characters in } S1$$

$$D(i, 0) = i \text{ for all } 0 \leq i \leq n \quad \textit{Delete } i \textit{ characters from } S1$$

39

Edit distance (cont'd)

$$D(i, j) = \min \left\{ \begin{array}{ll} D(i-1, j) + 1 & \textit{Delete character } i \textit{ from } S1 \\ D(i, j-1) + 1 & \textit{Insert character in } S1 \\ D(i-1, j-1) + t(i, j) & \textit{if } t(i, j) \neq 0, \textit{ substitute character } \\ & \textit{ } S1(i) \textit{ with } S2(j) \end{array} \right.$$

where $t(i, j) = 0$ if $S1(i) = S2(j)$ and $t(i, j) = 1$ otherwise.

$D(m, n)$ is the edit distance between $S1$ and $S2$.

40

Dynamic programming - tabular computation

- Allocate table $D[0..m,0..n]$
- Fill in row 0 and column 0
 $D(0, j) = j$ for all $0 \leq j \leq m$
 $D(i, 0) = i$ for all $0 \leq i \leq n$
- Fill in remaining elements, for example row-wise or column-wise
$$D(i, j) = \min \begin{cases} D(i-1, j) + 1 \\ D(i, j-1) + 1 \\ D(i-1, j-1) + t(i, j) \end{cases}$$
- $D(m, n)$ gives edit distance $S1, S2$.
- Find alignment/edit transcript by tracing back from m, n to $0, 0$

41

Alignment of biological sequences

- Use *scoring matrix* that rewards alignment of similar amino acids.
- Find alignment that *maximises* similarity of aligned amino acids.
- Gaps penalised typically using *affine* gap penalty
 - gap of length k penalised by $ak+b$
- Simple modification of edit distance algorithm

42

Global alignment of two sequences

- Assuming linear gap penalty ($b=0$):

$$S(0, j) = -ja \text{ for all } 0 \leq j \leq m$$

$$S(i, 0) = -ia \text{ for all } 0 \leq i \leq n$$

$$S(i, j) = \max \begin{cases} S(i-1, j) - a \\ S(i, j-1) - a \\ S(i-1, j-1) + \text{score}(i, j) \end{cases}$$

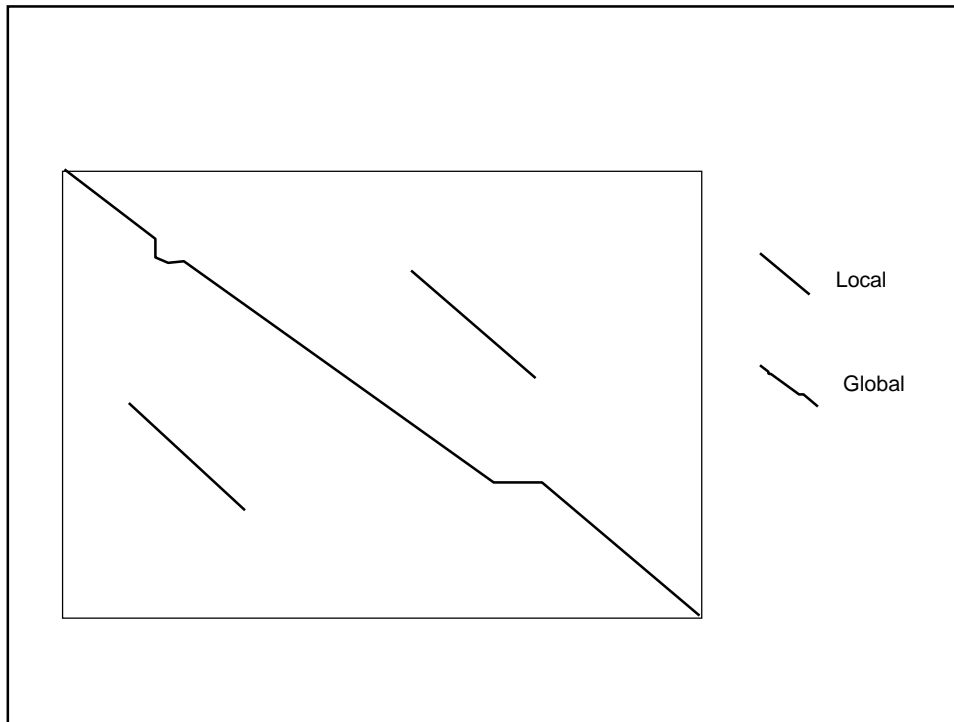
43

Example of pairwise global alignment

	I N D U S T R Y									
	0.0	-0.5	-1.0	-1.5	-2.0	-2.5	-3.0	-3.5	-4.0	
I	-0.5	1.0	0.5	0.0	-0.5	-1.0	-1.5	-2.0	-2.5	
N	-1.0	0.5	2.0	1.5	1.0	0.5	0.0	-0.5	-1.0	
T	-1.5	0.0	1.5	1.0	1.0	0.5	1.5	1.0	0.5	
E	-2.0	-0.5	1.0	1.0	0.5	0.5	1.0	1.0	0.5	
R	-2.5	-1.0	0.5	0.5	0.5	0.0	0.5	2.0	1.5	
E	-3.0	-1.5	0.0	0.0	0.0	0.0	0.0	1.5	1.5	
S	-3.5	-2.0	-0.5	-0.5	-0.5	1.0	0.5	1.0	1.0	
T	-4.0	-2.5	-1.0	-1.0	-1.0	-1.0	2.0	-1.5	1.0	

Score:
 Identical letters: 1.0
 Different letters: -0.5
 Gap: -0.5

44



Local Alignment

- Sequences: A and B
- Pick
 - substring a from A
 - substring b from B
- so that a and b can be aligned giving score $Sc(a,b)$
- and $Sc(a,b)$ is maximum over all substrings a, b from A, B .

Local alignment - recurrence relation

$$S(0, j) = 0 \text{ for all } 0 \leq j \leq m$$

$$S(i, 0) = 0 \text{ for all } 0 \leq i \leq n$$

$$S(i, j) = \max \begin{cases} S(i-1, j) - a \\ S(i, j-1) - a \\ S(i-1, j-1) + \text{score}(i, j) \\ 0 \end{cases}$$

47

Dynamic Programming for alignment of structures

- **Structures:**
Scoring of equivalenced elements are not independent.
 - Coordinates transformed - rigid body
 - Relations between elements (e.g., distances)
- For DP to work, the optimality of earlier made choices cannot be affected by new choices.
 - Does not hold for structure alignment

48

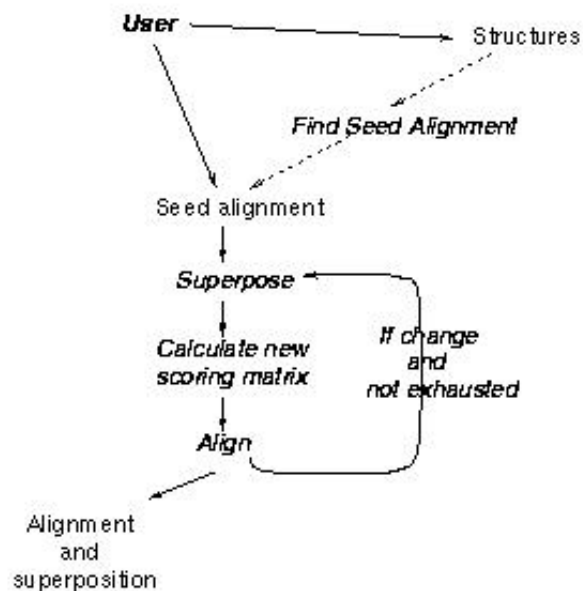
Alternating superposition and alignment

Align A and B.

1. P = Initial alignment/equivalence
2. Superpose A onto B based on P
→ make scoring matrix R
3. Align A and B using DP and R
→ new P'
4. If ($P' \neq P$), then
 $P = P'$, go to 2

Rao & Rossmann (1973), Rossmann & Argos (1975, 1976)

49



A simple alternating method - initial alignment

- Multiple ways
 - user specifies list of residue pairs
 - motif matches
- Given alignment (list of residue pairs), superpose structures A and B to minimize RMSd

51

A simple alternating method - from superposition to alignment

- From Superposition to scoring matrix and alignment
 - let $d(A_i, B_j)$ be the Euclidean distance between A_i and B_j after superposition
 - let τ be the upper limit on distance between residues to be rewarded
 - R is a scoring matrix that can be used for DP alignment of the superposed structures

$$R(A_i, B_j) = \frac{1}{d(A_i, B_j)} - \frac{1}{\tau}$$

- zero gap penalty, and allow zero diagonal step

52

A simple alternating method - dynamic programming

$$D(0, j) = 0 \text{ for all } 0 \leq j \leq m$$

$$D(i, 0) = 0 \text{ for all } 0 \leq i \leq n$$

$$D(i, j) = \max \begin{cases} D(i-1, j) \\ D(i, j-1) \\ D(i-1, j-1) + R(i, j) \\ D(i-1, j-1) \end{cases}$$

53

A simple alternating method - dynamic programming (cont'd)

- **Forward:**
 - fill in matrix row-wise/column wise
 - $D(m, n)$ gives total score of best possible alignment (set of co-linear residue pairs)
- **Traceback:**
 - find all contributing (non-zero) diagonal moves
 - this is the new alignment to be used for the next superposition

54

Part II - Algorithms for pairwise structure comparison

- Double dynamic programming

- Geometric hashing

55

Double Dynamic Programming

- W.R. Taylor and C.O. Orengo. Protein structure alignment. *J. Mol. Biol.*, 208:1-22, 1989.
- C.O. Orengo and W.R. Taylor. A rapid method for protein structure alignment. *J. Theor. Biol.* 147:517-551, 1990
- W.R. Taylor. Random structural models for double dynamic program score evaluation. *J. Mol. Evol.*, 44:174-180, 1997.
- W.R. Taylor. Protein structure comparison using iterated double dynamic programming. *Prot. Sci.*, 8:654-665, 1999

56

Double dynamic programming

- Ideally, one wishes to simultaneously align and superpose the structures
- Thus the independency requirement is violated
- However, several heuristic algorithms trying to extend DP to solve this problem have been developed
- One such method is SSAP (Structure Sequence Alignment Program), based on Double Dynamic Programming (DDP)

57

DDP - outline

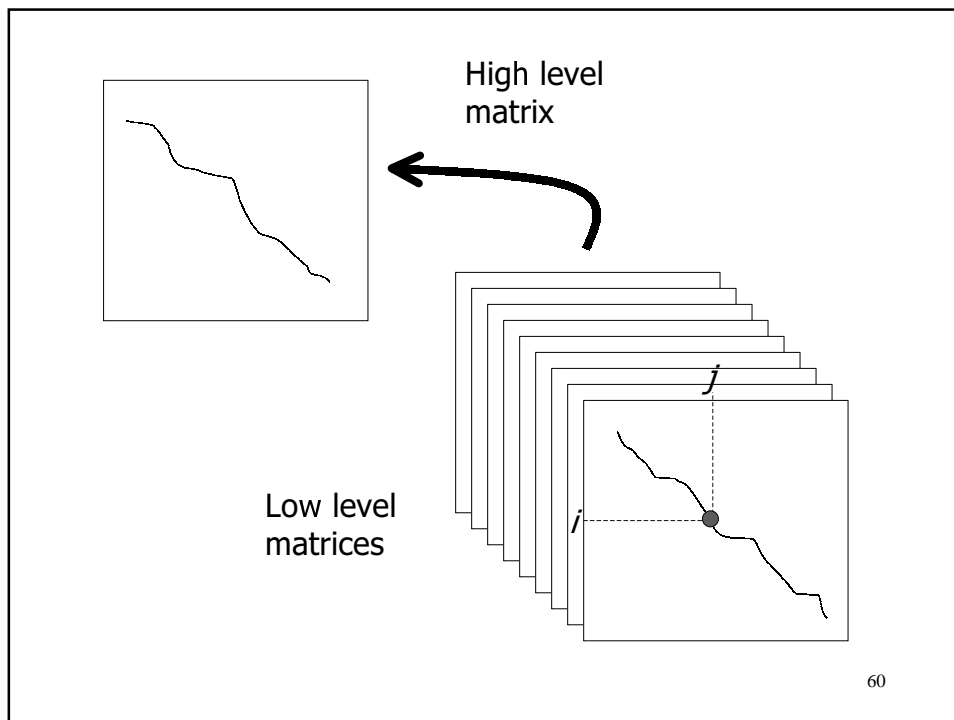
- DDP ideas
- Algorithms
- Iterated DDP

58

Ideas for DDP

- Use two levels of dynamic programming, to construct a *high level* scoring matrix that can be used for ordinary DP
- For each residue pair (a_i, b_j) this matrix should show how likely it is that the pair is on an optimal alignment
- For each (a_i, b_j) , this likelihood is found by a (*low level*) optimal alignment with the constraint that (a_i, b_j) is part of the alignment
- The scores along the low level alignments are accumulated in the high level scoring matrix

59



60

Ideas for DDP (cont'd)

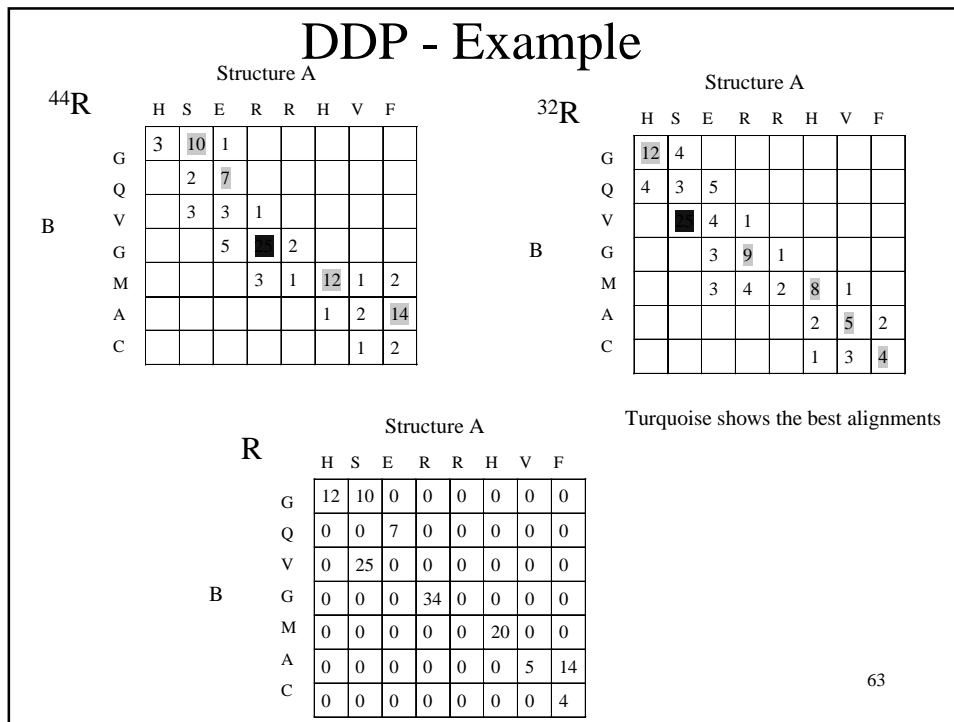
- A low level scoring matrix ${}^{ij}R$ is defined for each pair (a_i, b_j)
- ${}^{ij}R_{kl}$ is a score showing how well the residue a_k fits to b_l under the constraint that a_i is aligned to b_j .
- Low level DP is performed using ${}^{ij}R$ (in principle for each pair (i,j))
- The contribution from ${}^{ij}R$ is all ${}^{ij}R_{kl}$ such that (a_k, b_l) lies on the optimal (low level) path

61

Algorithm for DDP

```
 $R := \{0\}$  High level scoring matrix 0  
for each pair  $(a_i, b_j)$  do  
  compute the low level scoring matrix  ${}^{ij}R$   
   $(s, P) := \mathbf{DP}_{{}^{ij}R}^*(A, B)$  Low level DP forced through  $(a_i, b_j)$   
  P is the optimal path, s the score  
  forall  $(a_p, b_q)$  in  $P$  do  $R_{pq} := R_{pq} + {}^{ij}R_{pq}$  Accumulate into R  
end  
 $(s, P) := \mathbf{DP}_R(A, B)$  High level DP using R
```

62



Noise

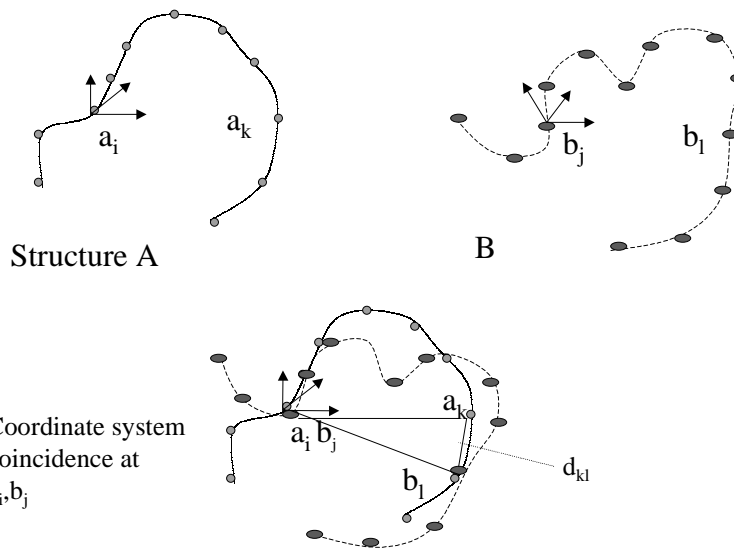
- As R contains a sum of values from the low level matrices, it is normalised before high level DP
- R is the sum of nm low level matrices, most of them representing pairs which are not in the final alignment, thus introducing noise
- Therefore, only low level alignments scoring above a cutoff is accumulated

The low level scoring matrices

- ijR_{kl} should be based on superposition bond on a_i and b_j
- Define local reference systems around a_i and b_j e.g. by using (the C_α s of) a_{i-1} , a_i , a_{i+1} and b_{j-1} , b_j , b_{j+1}
- The coordinates of A's and B's residues are transformed into the respective coordinate systems
- One simple scoring scheme for aligning a_k and b_l depends only on the distance between a_k and b_l in their respective coordinate systems

65

Example



66

More advanced scoring scheme

- A *direction* component, realised as a function of the vector d_{kl}
- An *orientation* component, reflecting the difference in reference frames around a_k and b_l (after superposition)
- A *sequence* component g_{kl} , calculated as an increasing function of $|k-i| + |l-j|$. This should damp the contribution from near neighbours in the sequence
- A *spatial* component h_{kl} , calculated as a decreasing function of $d_{ik} + d_{jl}$, where d_{ik} is the distance between a_i and a_k . This will damp the contribution from large distances in space (want high contribution from pairs near in space and far in sequence)

67

Iterated DDP

- As DDP is heuristic, one might achieve better results by iterating, doing a DDP in each cycle
- This, together with limiting the work for each DDP is developed into a program called SAP (Taylor 1999)

68

Iterated DDP (cont'd)

- In each cycle, only some of the pairs (a_i, b_j) are used for low level DP
- Some pairs, called **seeds**, are chosen for the first cycle either chosen randomly, or found by a (motif discovery) program (e.g. SPratt)
- In each cycle the high level scoring matrix R is updated, and also the selected pairs (a_i, b_j) for the next cycle
- A bias matrix (Q) is used in updating R

69

Algorithm for iterated DDP

```
initialise the bias matrix  $Q$ 
select the seed pairs using  $Q$ 
iter
  place the selected pairs in  $I$ 
   $R := \{0\}$  Set high level matrix to zero
  for each pair  $(a_i, b_j)$  in  $I$  do
     $(s, P) := DP_{iR}^*(A, B)$  Low level DP
    accumulate the low level results to  $R$ 
  end
  update  $Q$  using old  $Q$  and  $R$ 
   $(s, P) := DP_Q(A, B)$  High level DP
  select new pairs  $(a_i, b_j)$  based on  $Q$ 
until termination criterion is satisfied
```

70

Similarity Between Alternating Approach and Iterated DDP

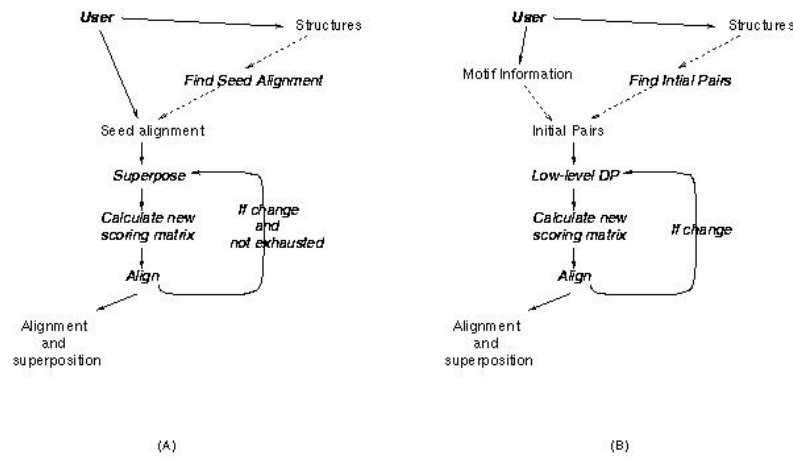


Figure 6: (A): Outline of algorithm alternating between alignment and superpositioning. (B): For comparison, outline of the SAP method.

Algorithm statements to explain

- How is Q initialised?
- How many residue pairs should be chosen in each cycle?
- How should Q be updated?
- Should the high level scoring matrix contribute to the lower ones?
- What is the termination criterion?

Initialising the bias matrix Q

Q is initialised by using the three components:

- Secondary structure
- Burial
- Amino acid similarity

73

Selecting pairs and updating Q

- The highest values in Q are used for selecting the pairs for low level DP
- Continuity through the cycles (p) is maintained by using Q as base for incremental revision (weakening contribution from initial Q):
$${}^{p+1}Q := {}^pQ / 2 + \log(1 + {}^{p+1}R / 20)$$
- 10-20 pairs are initially selected, but the number is gradually increased, (and more "true pairs" are found)
- The number of selected pairs is $K = 10 + \frac{p+1}{20} \sqrt{mn}$

74

Coherence problem

- The initial selection of residues pairs might be quite random with respect to the final "true" equivalences
- As a consequence, the comparison of their environments might provide little *coherent* direction towards the final solution
- Although the bias matrix provides a platform from which the selection of pairs can be refined, it has no effect on the scores derived from the low-level matrices

75

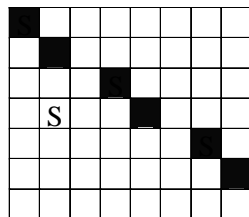
High level contribution to low level

- A contribution from the bias matrix can therefore be introduced to provide stability into early cycles
- This is done by using a revised matrix
$${}_{ij}R^* = {}_{ij}R + pQG(p, I)$$
where p is the cycle number and G is the Gaussian transform function
- This provides a smooth transition from the, ideally, local information in the bias-matrix into the full global view provided by the comparison of the residue environments

76

Termination criterion

- The iteration should stop when the selected pairs coincide with the best (high level) path
- Let U be the number of selected pairs on the best path using Q , and V the number of selected pairs outside. Then stop if $\frac{U}{U+V}$ becomes 1, or stops to increase



4 pairs selected
 $U = 3, V = 1$

- An upper limit for the number of cycles must be defined

77

Geometric Hashing

- R. Nussinov and H.J. Wolfson. Efficient detection of three-dimensionale structural motifs in biological macromolecules by computer vision techniques. Proc. Natl. Acad. Sci. USA., 88:10495-10499, 1991.
- L. Holm and C. Sander. 3-d lookup: Fast protein structure database searches at 90 % reliability. Proceedings of the Third International Conference on Intelligent Systems for Molecular Biology 179-187.
- V. Alesker, R. Nussinov and H.J. Wolfson. Detection of non-topological motifs in protein structures. Protein Eng., 9:1103-1119, 1996.
- H.J. Wolfson. Geometric hashing: An overview. IEEE comp. Science and Eng., 1997.

78

Geometric Hashing - Outline

- Ideas
- Geometric hashing for 2 D
- Geometric hashing for structure comparison
- Geometric hashing for SSE-representation

79

Geometric hashing

- Geometric hashing was originally developed as a computer vision technique for matching geometric features
- It can be used to find common subfigures, invariant under rotation, translation and scale
- In structure comparison it can be used
 - as a full comparison method
 - for finding seeds for other methods (methods that use iteration or clustering)
- We will first describe Geometric Hashing for 2D figures

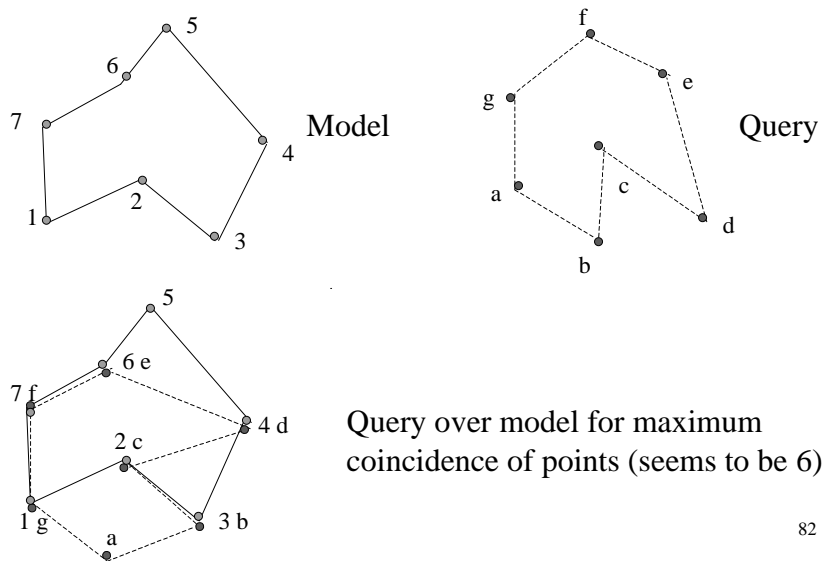
80

2D geometric hashing

- Two figures are given, a **model** A , and a **query** B , described by m and n points, respectively
- Find common subfigures, invariant under rotation and translation (scale is not used)
- One approach is to "place the query on top of the model", and consider how many points coincide (*here ignoring the edges*)

81

Example



82

Coincidence sets

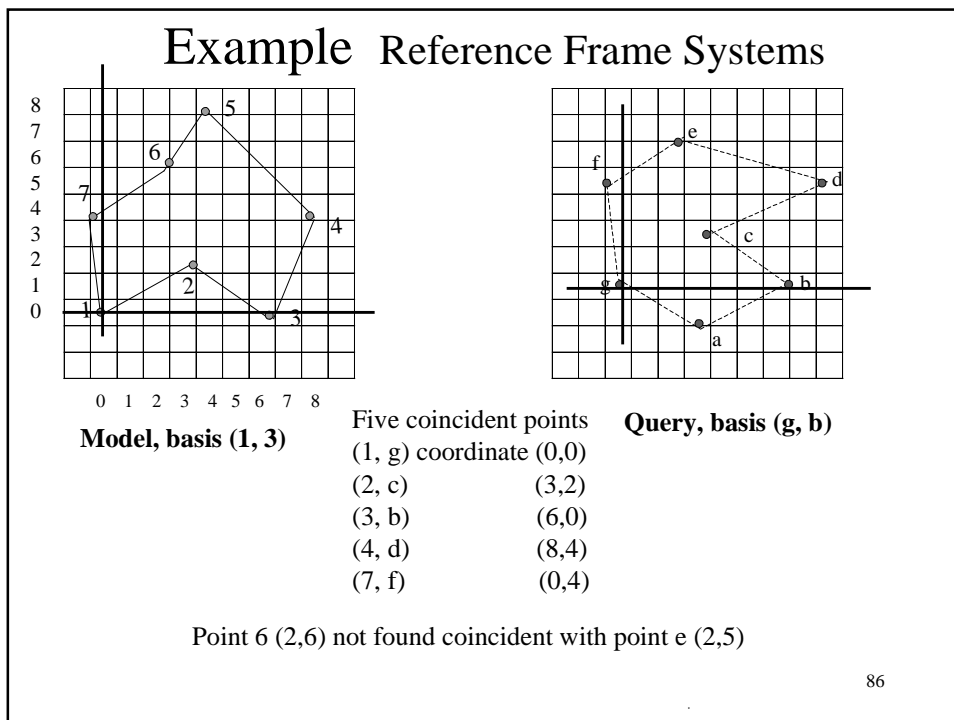
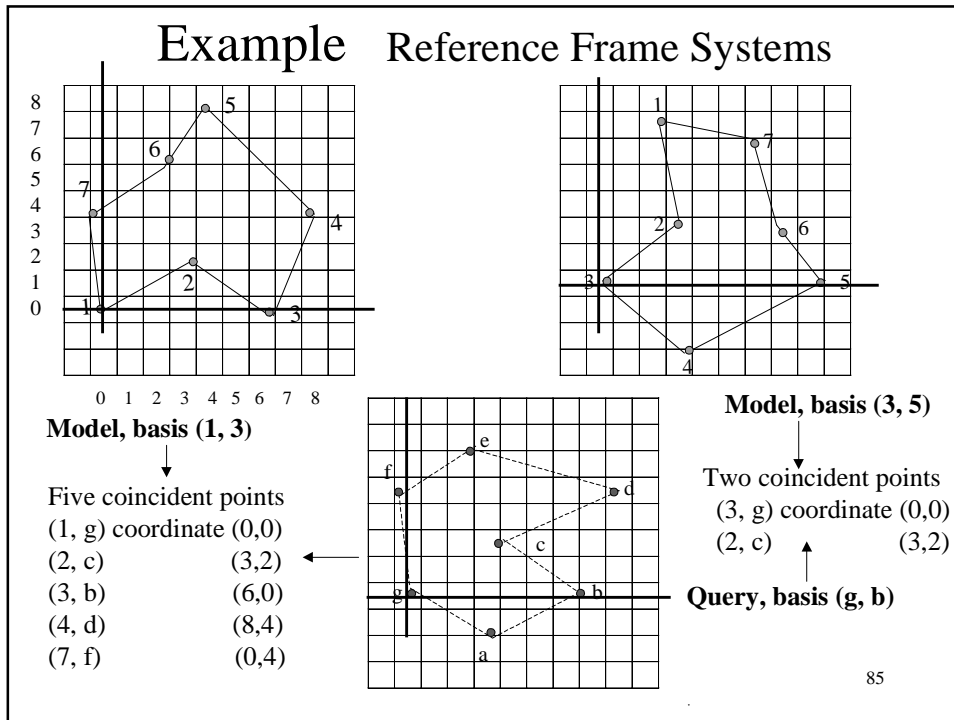
- Finding the maximal coincidence set is NP-hard
- In this context, we want to find *all* coincidence sets with the number of pairs over a given threshold

83

Reference frames

- Define coordinate systems for both figures (A , B), called **reference frames**
- Two points (**basis pair**) can define a reference frame, e.g. origo at one of them, and one of the axis through both points
- The coordinates of the points are computed in the reference frame, constituting a **reference frame system**
- Count how many pair of points (one from each figure) have the same coordinates

84



Remarks

- The number of coincident points depends on the *resolution* of the coordinate system, on the *basis pairs* used
- Generally, all combination of points should be used as basis pairs, resulting in comparing
reference frame systems $\frac{m(m-1)}{2} \cdot \frac{n(n-1)}{2}$
- Using all combinations might introduce redundancy. Let (a_i, a_k) and (b_j, b_l) be the basis pairs, and (a_r, b_u) and (a_s, b_v) both coincide. Then it is likely that the same coincidence set is found if (a_r, a_s) and (b_u, b_v) are used as basis. *Note however that similarity and not exact equality is used*

87

Hashing

- For dealing efficiently with all combinations, hashing is used. It is especially efficient when several queries are to be compared to one model, or to several models
- The comparison problem can be formulated as: *given a query reference frame system, for each model reference frame system, in how many cells are there points from both the query and model frame system*
- The hashing technique makes it possible to simultaneously compare a query frame system to all model frame systems

88

Preprocessing

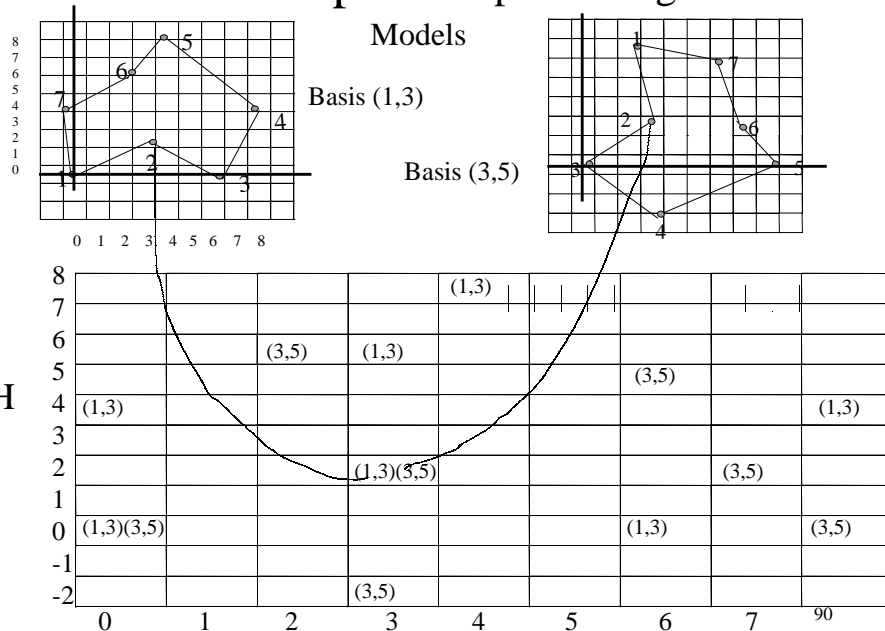
- In this example, a 2D hash table H is used. It has a bin for each cell in the frame systems. In a **preprocessing** phase, the coordinates of all points in each **model** frame system are found. If there is a point in the cell (p,q) in the frame system with basis (a_p, a_k) , then (a_p, a_k) is placed in the bin $H(p,q)$
- Since all pairs of points from the model will (generally) act as basis pairs, totally

$$m \frac{m(m-1)}{2}$$

pairs will be in H

89

Example Preprocessing

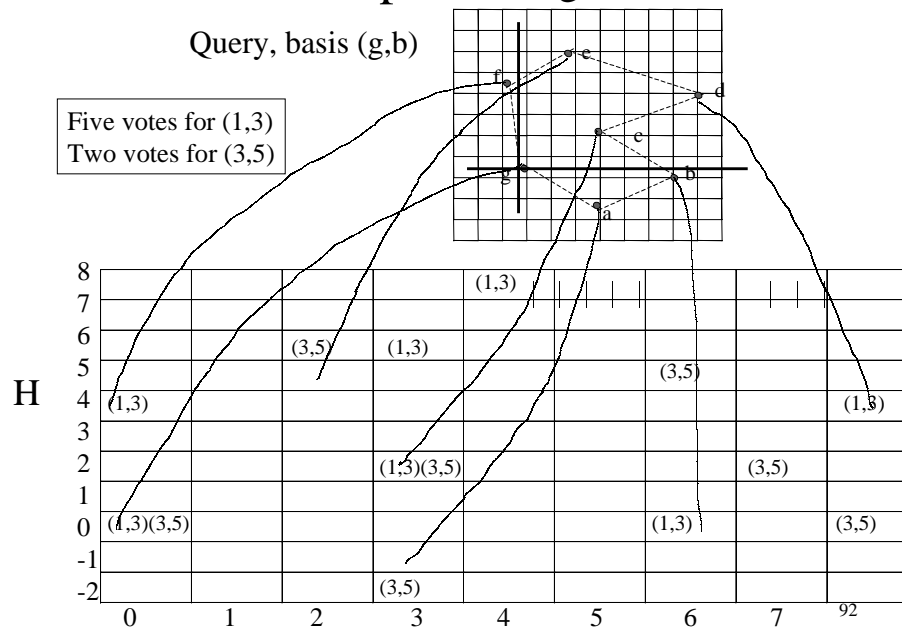


Recognition

- The **query** is compared to the model in the **recognition** phase
- A pair is chosen as basis, and the coordinates of the other points are calculated
- These coordinates are used as indices into H, and for each cell being indexed, a vote is given for the (model) basis pairs in the cell. The number of votes for a model basis pair is the number of coinciding points to the query (using the specified query basis pair)

91

Example Recognition



Extensions

- Labels (e.g. colours and/or forms) assigned to the points might also be included, such that coinciding points also must have equal labels. This can be implemented in two ways:
 - Storing the labels in the table
 - The table can be hashed by using the labels in addition to the coordinates
- It is straightforward to extend the hashing technique to include several models, such that a query is simultaneously compared to several models. The only extension in the hash table is that a model identification must be stored along with the basis pairs

93

GH for structure comparison

- Since comparing structures should be invariant to translation and rotation, GH is a good candidate method, when the order of the residues (elements) along the chain is ignored
- Finding a coinciding set between two structures then means finding an equivalence (not necessarily an alignment)
- A 3D reference (frame) system must be defined, and often the C_{α} -coordinates of three (non-colinear) residues are used

94

Algorithm for preprocessing

```
for each model  $M_u$  do  
  for each (unordered, noncollinear) triples  $(a_i, a_k, a_p) \in M_u$  do  
    calculate the reference frame  $R_{ikp}$   
    for each residue  $r$  do  
      calculate  $F = F(a_i, a_k, a_p, a_r, p_L)$  ;index in  $H$   
       $H(F) := H(F) \cup (M_u, R_{ikp})$   
    end  
  end  
end
```

The label L is included in the hashing

95

Algorithm for recognition

```
repeat  
  initialise the vote table  $V$  to 0  
  choose three atoms  $(a_j, a_b, a_s)$  from the query as basis  
  calculate the reference frame system  $R_{jls}$   
  for each residue  $q$  do  
    calculate  $F := F(a_j, a_b, a_s, a_q, q_L)$   
    for each pair  $(M, R) \in H(F)$  do  
       $V(M, R) := V(M, R) + 1$   
    end  
  end  
until satisfactory coincidence sets are found  
  or all query reference frames are used
```

96

Remarks

- The result of the recognition is a list of (M, R_M, R_B) , showing that there is a coincidence set between the model M and the query B , becoming evident when the reference frames R_M and R_B are used
- The residues of the coincidence sets are known (or can be found), and a superposition can be done for checking the substructure similarities
- *Techniques for reducing the run time for Geometric Hashing exist*
- *Techniques also exist for "practical adaption" (e.g. checking neighbouring cells)*

97

Geometric hashing for SSE- representation

- Typically the SSE's are represented as *sticks*
- By use of two sticks, a coordinate reference frame can be defined (usually by placing one axis along one of the sticks)
- An entry in the hash table might be (Holm and Sander) a list where each list-element contains:
 - identification of the SSE
 - identification of the basis
 - the type of SSE (alpha, beta)
 - the midpoint of the SSE (in the reference frame)
 - the direction of the SSE
 - possible other information

98

Part III - Comparison by Clustering

- Introduction
- Compability and consistency
- Clustering methods
- Clustering by use of transformation
- Clustering by use of relation
- Clustering as graph problem

99

Comparison by Clustering

- The clustering methods for structure comparison works by first finding small similar substructures in the two structures, and then constructing larger similar substructures by joining smaller ones
- Generally they allow for insertions, deletions and topological permutations
- The word *clustering* is a bit misleading, not all "objects" have to be clustered. It is used here, since it is widely used in papers describing the relevant methods. (Perhaps *grouping* would be better)

100

Compatibility

- The basic objects are **compatible elements**, one from each structure
- Two elements are compatible if they share some similarities, what kind of similarities vary with the method
 - Residues may be compatible if they are the same amino acid, or in the same amino acid group
 - Secondary structures may be compatible if they are of the same type
 - For fragments may equal intern distances be required

101

Seed matches

- The methods first find **seed matches** between the structures, and then join these into larger clusters, a cluster representing one substructure from each structure

102

Consistency

- A seed match (being a basic cluster) consists of one or several pairs of compatible elements
- The pairs must be **consistent**, meaning that the two substructures consisting of the elements from the pairs together must be compatible, i.e. similar to a certain degree
- Seed matches are joined into consistent clusters, and scored, measuring how similar the two substructures are

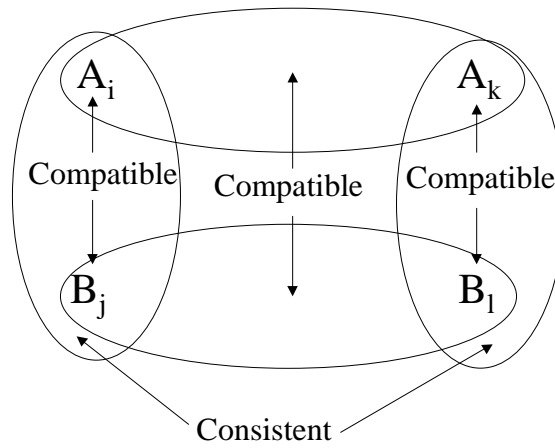
103

Compatibility and consistency

- *Compatibility* is a binary relation between elements from different structures
- *Consistency* is a binary relation between clusters (of compatible elements)

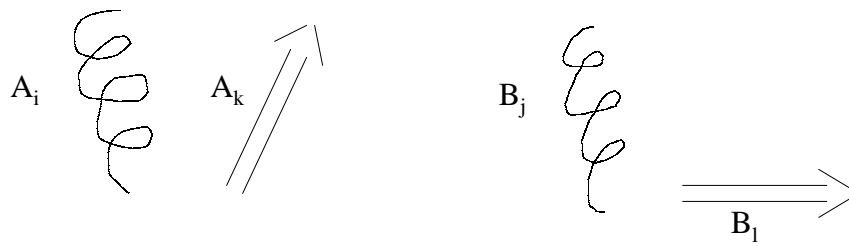
104

Compatibility and consistency



105

Example



A_i compatible with B_j
 A_k compatible with B_l
 (A_i, B_j) not consistent with (A_k, B_l)

[(A_i, A_k) not "similar to" (B_j, B_l)]

106

The philosophy

- Find seed matches, i.e. sets of consistent pairs of compatible elements $SM = \{P_s\} = \{(A_{s_i}, B_{s_j})\}$
 - All elements in a seed match must be different
 - An element may be in several seed matches
- Group consistent seed matches iteratively into clusters, representing the (k) substructures with highest score
- Optional refinement (often using the iteratively alternating superposition/alignment)

107

The methods

The methods vary in how they solve the following:

- How is compatibility defined
- How to find the compatible elements, and the seed matches
- How is consistency defined
- How is clustering performed – clustering algorithms
- How are the clusters scored
- How is the refinement done (if at all)

108

Searching for seed matches

Different searching methods are used:

- Straightforward search, when each seed match consists of only one pair
- Geometric hashing, elements in seed matches constraint to lie within (small) spheres
- ...

109

Consistency

- Two clusters $C_1 = (C_1^A, C_1^B)$, $C_2 = (C_2^A, C_2^B)$, (a cluster consisting of one or more seed matches) can be joined if they are consistent
- This means that the substructure of A consisting of $(C_1^A \cup C_2^A)$ must be "similar" to the substructure of B consisting of $(C_1^B \cup C_2^B)$

110

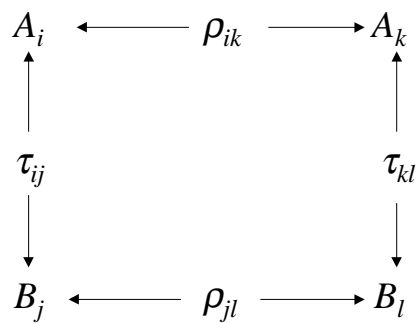
Decide consistency

To decide consistency, one can use:

- **transformation** (τ) between compatible elements from *different* structures
- **relation** (ρ) between elements of the *same* structure

111

Consistency



A_i compatible with B_j

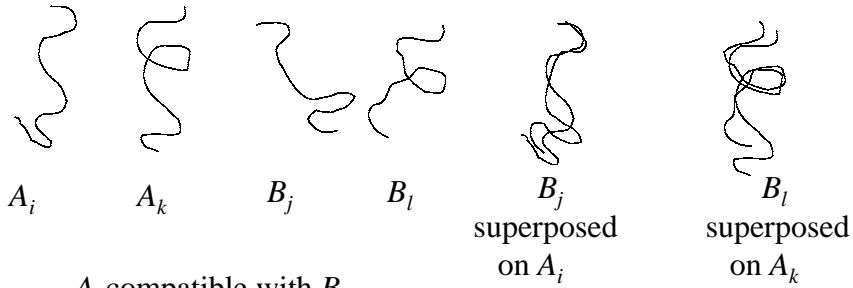
A_k compatible with B_l

Is (A_i, B_j) consistent with (A_k, B_l) ?

Yes, if $(\tau_{ij} = \tau_{kl})$ or $(\rho_{ik} = \rho_{jl})$

112

Example transformation

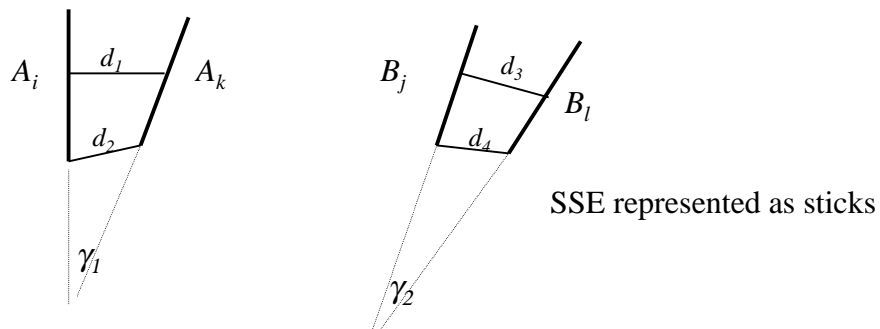


A_i compatible with B_j
 A_k compatible with B_l

(A_i, B_j) is consistent with (A_k, B_l) if
 the transformation for best superposition of (A_i, B_j)
 is similar to the transformation for the best
 superposition of (A_k, B_l)

113

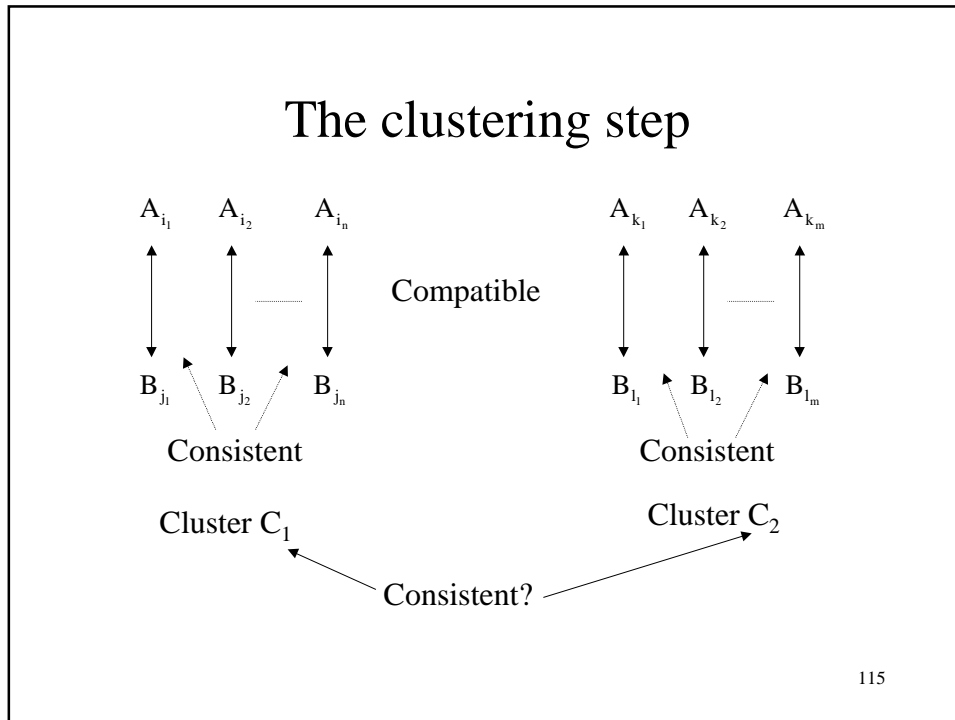
Example relation



A_i compatible with B_j
 A_k compatible with B_l

(A_i, B_j) is consistent with (A_k, B_l) if $d_1 \approx d_3 \wedge d_2 \approx d_4 \wedge \gamma_1 \approx \gamma_2$

114



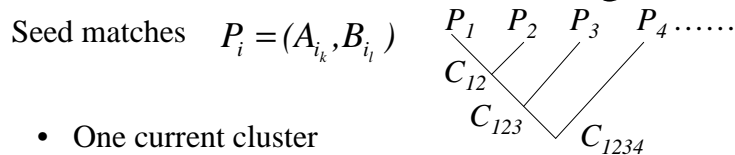
- ### Consistency test
- $C_P = \{P_1, P_2, P_3, \dots\}$ $P_i = (A_{i_k}, B_{i_l})$
 - $C_Q = \{Q_1, Q_2, Q_3, \dots\}$
 - Can C_P be joined with C_Q ? (C_P consistent with C_Q ?)
 - Use of *local* criteria: Use the pairs (P_i, Q_j) for test
 - Transitivity: $\text{cons}(P_i, Q_j) \Rightarrow \forall kl : \text{cons}(P_k, Q_l)$
 - Enough to test one pair (Example =)
 - Transitivity not satisfied
 - Must test each pair (in principle) (Example distance)
 - Use *global* criteria
 - Test the clusters C_P, C_Q "as units"
- 116

Clustering methods

- Linear clustering
- Parallel linear clustering
- Hierarchical clustering

117

Linear clustering



- One current cluster
- Add one consistent seed match to current cluster
- The result may depend on the order of the seed matches (due to inaccuracy)

Algorithm

select one seed match as current cluster C

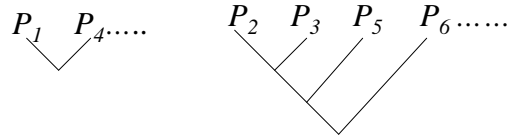
for each other seed match sm **do**

if sm is consistent with C **then** let C be sm joined to C

end

118

Parallel linear clustering



- Several current clusters

Algorithm

make a current cluster of each seed match (retaining them)

repeat

find the seed match *sm* "most consistent" to a cluster *C*

if the consistency is strong enough **then**

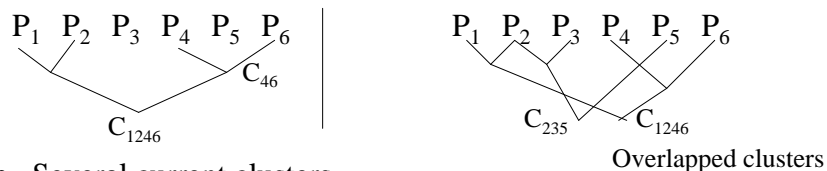
join *sm* to *C*; remove *sm* from the set of seed matches

end

while a seed match is joined to a cluster

119

Hierarchical clustering



- Several current clusters
- Two clusters are joined in each cycle

- *Algorithm* (one example)

make a set of clusters, each seed match being a cluster

loop

compare all pairs of clusters, and find the highest scoring pair

if the found score is higher than the highest scoring current cluster **then**

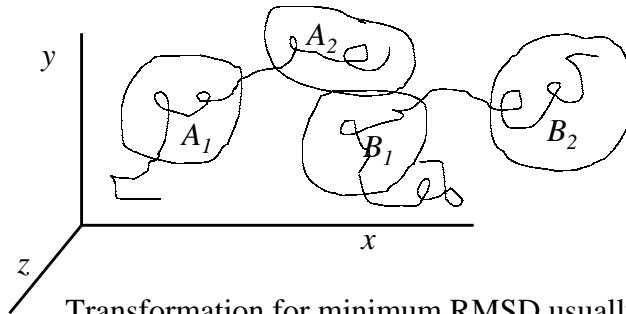
remove the two clusters from the cluster set; join the two clusters

end

until no more clusters are joined

120

Clustering by use of transformation



Transformation for minimum RMSD usually used

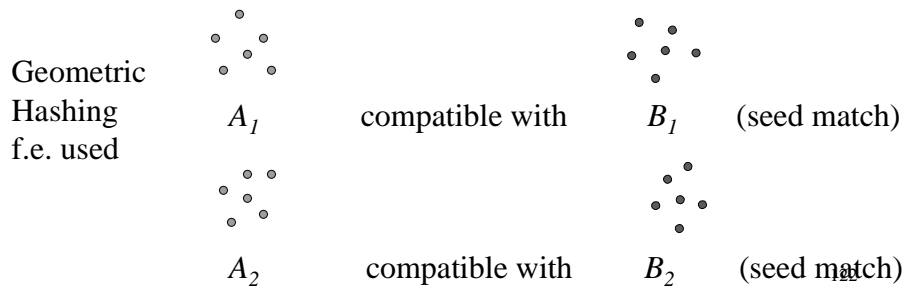
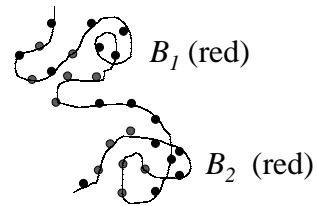
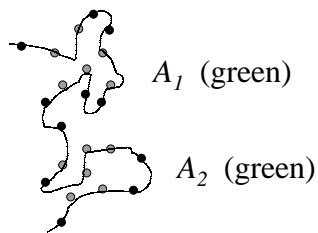
T_1 define best superposition for (A_1, B_1)

T_2 define best superposition for (A_2, B_2)

$T_1 = T_2 \Rightarrow T_1$ is best transformation for $(A_1 \cup A_2, B_1 \cup B_2)$

121

Seed matches



The transformation

- The transformation (for best superposition) is defined by 6 parameters: 3 for translation and 3 for rotation
- These are represented as a matrix
- Ideally *local* clustering criteria can be used, but due to inaccuracy, global is *used*
- This means that the transformation for the best superposition of $(A_1 \cup A_2, B_1 \cup B_2)$ is calculated for later cycles

123

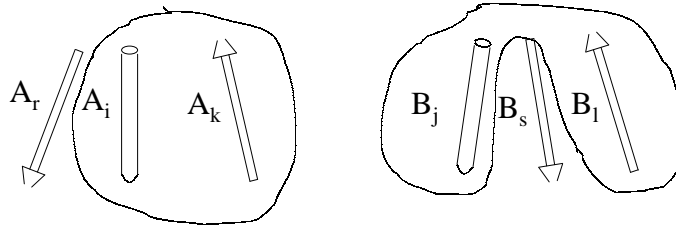
Clustering by use of relation

- The elements are mainly SSEs, represented as sticks
- A seed match is usually a pair of sticks
- To decide if two compatible pairs (A_i, B_j) and (A_k, B_l) are consistent, an intern relation ρ is used
- The pairs are consistent if $\rho(A_i, A_k)$ is approximately equal to $\rho(B_j, B_l)$
 ρ should satisfy the following: $\rho(A_i, A_k)$ and $\rho(B_j, B_l)$ should be approximately equal if and only if the substructures (A_i, A_k) and (B_j, B_l) are considered as similar
 ρ should be invariant with respect to rotation and translation, and in most cases to sequential order

124

Clustering by use of relation

Local clustering criteria is used
Transitivity is generally not fulfilled



Example

Cluster $C = \{(A_i, B_j), (A_k, B_l)\}$

(A_r, B_s) consistent with (A_i, B_j)

(A_r, B_s) not consistent with (A_k, B_l)

This is the same as $\tilde{n}(A_i, A_r) \approx \tilde{n}(B_j, B_s)$ but not $\tilde{n}(A_k, A_r) \approx \tilde{n}(B_l, B_s)$

125

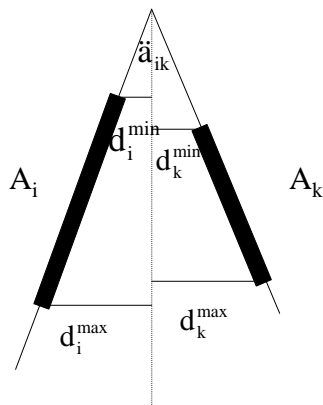
The relation

- The different methods use a different number of parameters for defining ρ
- The number depends on how stringent the constraints for being consistent is defined, i.e. given two sticks, how many parameters are needed for describing the "structural relation" between them? (means changing the value of one of the parameters might change the (sub)structure)
- Checking for consistency using transformation use 6 parameters, hence 6 should be enough. Usually one angle and 2-4 distances are used

126

Alexandrov and Fisher

Four distances and one angle



\ddot{a}_{ik} is found by projection of A_i and A_k into a plane parallel to both A_i and A_k

127

Alexandrov and Fischer

Requirement for consistency (A_i, B_j) (A_k, B_l)

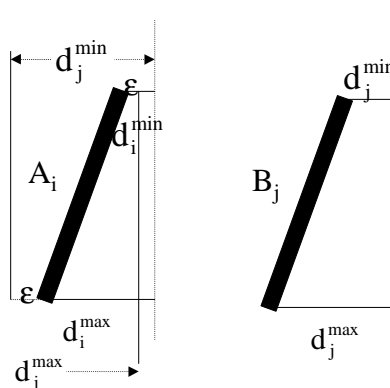
$$d_i^{\min} - d_j^{\max} < \ddot{a} \quad (\text{f.e. } 1.5\text{\AA})$$

$$d_j^{\min} - d_i^{\max} < \ddot{a}$$

$$d_k^{\min} - d_l^{\max} < \ddot{a}$$

$$d_l^{\min} - d_k^{\max} < \ddot{a}$$

$$|\ddot{a}_{ik} - \ddot{a}_{jl}| < \ddot{a}_{\text{lim}}$$



128

Clustering as graph problem

- Make a graph of a structure:
 - The nodes being the elements
 - Node labels being the properties
 - The edges are labeled by the relations
 - Edges between nodes where the relation satisfies some constraints
- Find subgraphs with maximum "similarity" (isomorphy)
 - Corresponding nodes must be compatible
 - Corresponding edges must be sufficient similar

129

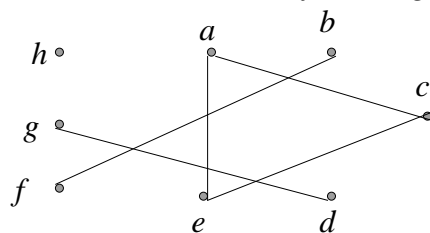
Node product graph

- If the scoring of the similarity of two substructures is number of compatible pairs, finding the maximum similar substructures is the same as finding a maximal clique in a *product graph*
- To construct a node product graph
 - Make a node for each compatible pair
 - Make edges between consistent nodes (compatible pairs)

130

Node product graph - Example

- Assume we have the following compatible elements:
 $a:(A_1, B_1)$ $b:(A_1, B_3)$ $c:(A_2, B_2)$ $d:(A_2, B_4)$
 $e:(A_3, B_3)$ $f:(A_3, B_1)$ $g:(A_4, B_2)$ $h:(A_4, B_4)$
- And the following relations are similar:
 $\rho(A_1, A_2) \approx \tilde{n}(B_1, B_2) \Rightarrow (A_1, B_1) \text{ cons } (A_2, B_2) : a - c$
- The same for $a-e$, $b-f$, $c-e$, $d-g$



Maximal clique $\{a, c, e\}$

$\{A_1, A_2, A_3\}$ similar to
 $\{B_1, B_2, B_3\}$

131

Refinement

- A final adjustment or refinement may be worth while
- Mainly done on residue level
- An iteration of alternating superposition/alignment is often done

132

Part IV

Multiple structure alignment and Motif discovery

- Extending pairwise approaches
 - Progressive alignment
 - Alternative methods
- Motif discovery
 - framework
 - the SP Pratt algorithm

133

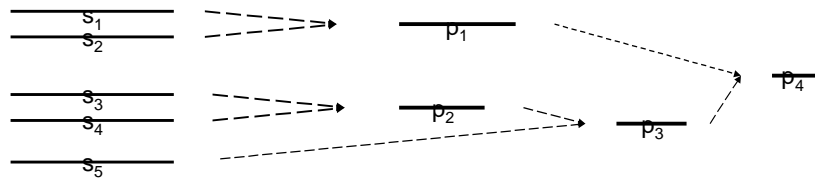
Multiple Alignment

- N sequences of length m
- Dynamic programming (N -dim matrix):
 $O(m^N)$
- Becomes unfeasible for, say, $N > 5$

134

Multiple Structure Comparison

- Computational complexity of most pairwise methods does not allow extension to multiple structures.
 - For example DP based methods.
- Progressive alignment



135

Example: Extending DDP to multiple alignment

- Align every structure to every other structure - get a *similarity matrix* - measure of similarity for every pair
 - Align the most similar pair - produce *consensus structure*
 - Align the new consensus structure to all others - update similarity matrix
 - Iterate until only one consensus structure remains
-
- Taylor et al 1994

136

MSSAP: Representing the consensus structures

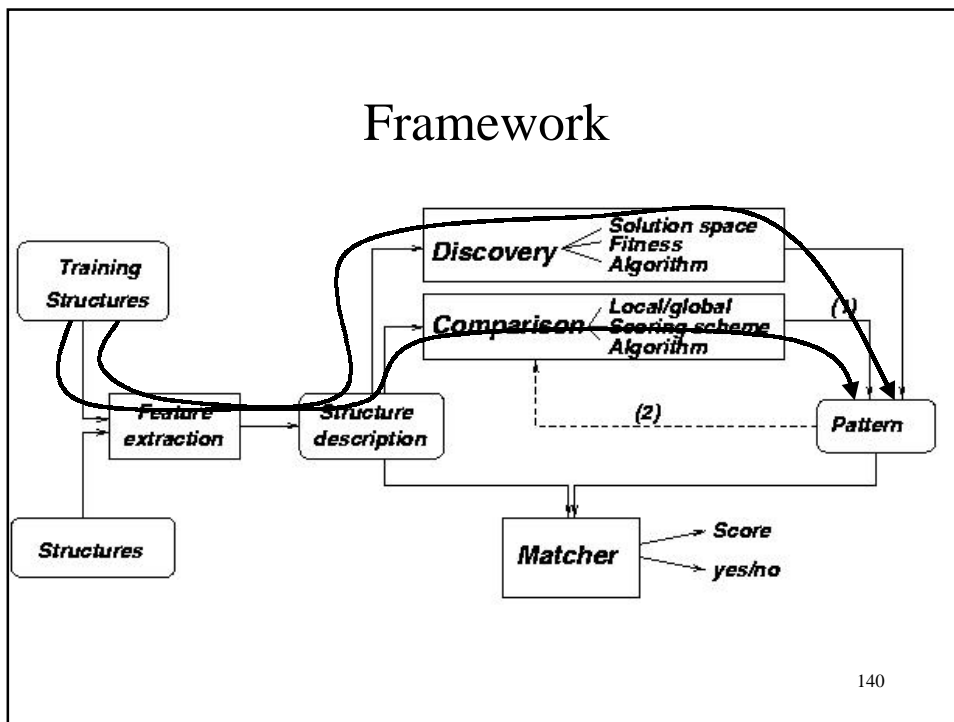
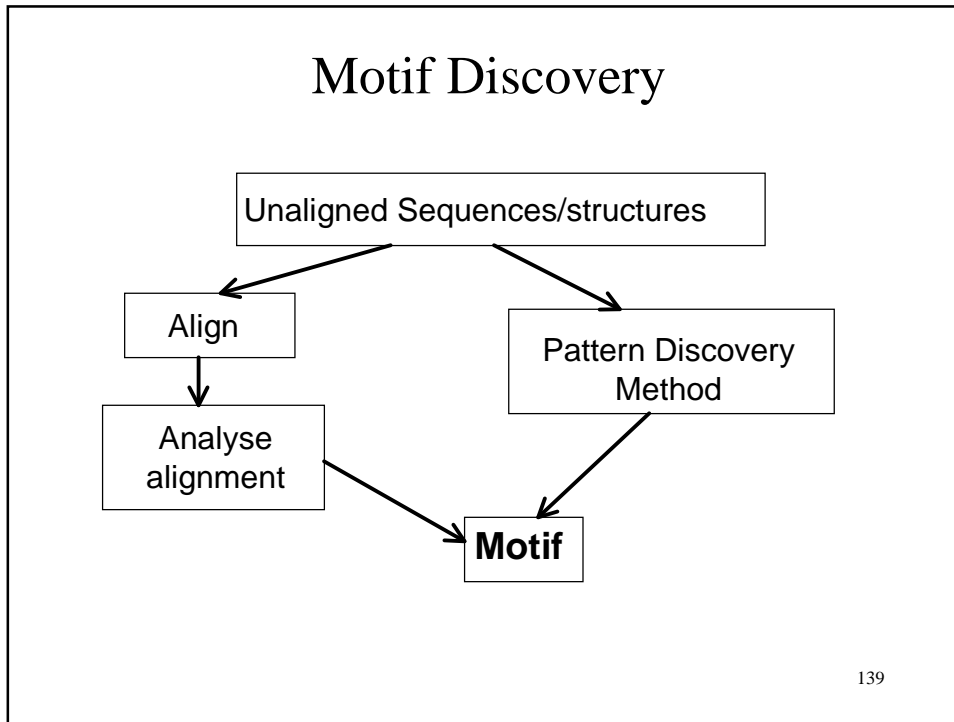
- Equivalent to a profile in multiple sequence alignment
- In SSAP interatomic vectors are used
- Here: bundles of interatomic vectors - they are represented by average vector and its variance (error measure)

137

Alternative approach - using a median structure

- Find median structure (the one closest to all the others)
 - Align each of the other structures onto the median by consistently combining the alignments
-
- Gerstein and Levitt (1998)

138



SPratt - Pattern Driven Algorithm for the Discovery of Structure Motifs

- Discover motifs consisting of a small number of *individual residues*
 - *close in space*
 - *preserve sequence order.*
- Do not use pairwise comparisons - use simultaneously information from all structures.
- Can find motifs recurring in one structure or in a set of structures.

Jonassen, Eidhammer, Taylor, *Proteins* 1999.

141

SPratt - Idea

- Describe (spatial) local neighbourhoods as strings (Karlin & Zhu, 1997)
- Use the sequence pattern discovery method Pratt to find pattern common to the strings
- Check if string similarity reflects structural similarity

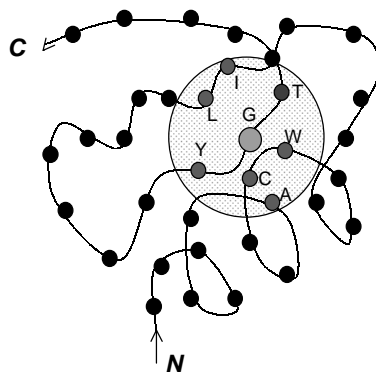
142

SPratt - Neighbour Strings

- For each residue a in each structure:
 - N_a : sequence of residues with spatial distance less than D from a , in order from N-to-C terminal, starting with a .
 - C_a : sequence of residues with spatial distance less than D from a , in order from C-to-N terminal, starting with a .

143

Neighbour string for each residue



Neighbour strings
for G:

C-string:
GT

N-string:
GYLIWCA

*Remember the
index of each residue*

144

SPratt - Discovery Algorithm

- Encode the neighbourhood of each residue in each structure as 2 strings (N-string and C-string)
- Use Pratt to find patterns in the N strings and the C strings *separately*
- For each pattern, check if the neighbour string matches reflect structural similarity

145

SPratt - results

- Can be used to analyse up to around 50 structures to find patterns matching (almost) all.
- Time consuming step: Pratt

146

Example output: Cystein proteases

Protein	1hucAB	1gcb	1fieA	1aim
length	252	452	705	215
Pattern				
S	S220	S393	S397	S176
[DN]	N219	N392	D396	N175
H	H199	H369	H373	H159
F	F32	F76	F317	F28
C	C29	C73	C314	C25

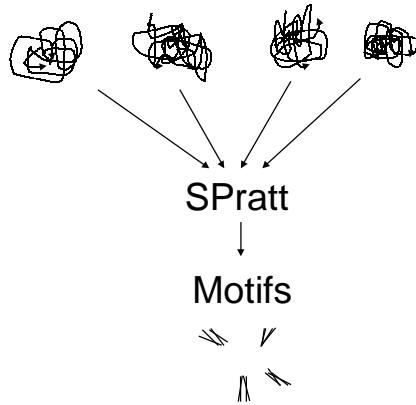
147

RMSd matrix

	1hucAB	1gcb	1fieA	1aim
1hucAB		0.2	1.6	0.6
1gcb			1.5	0.7
1fieA				1.8
1aim				

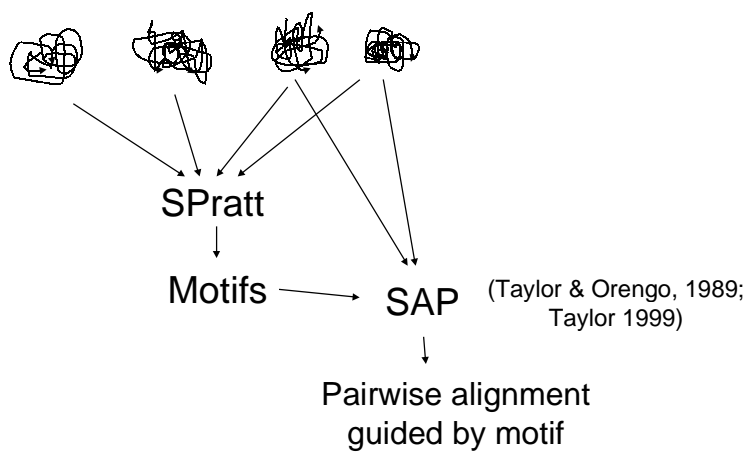
148

SPratt: Structures \rightarrow Motif

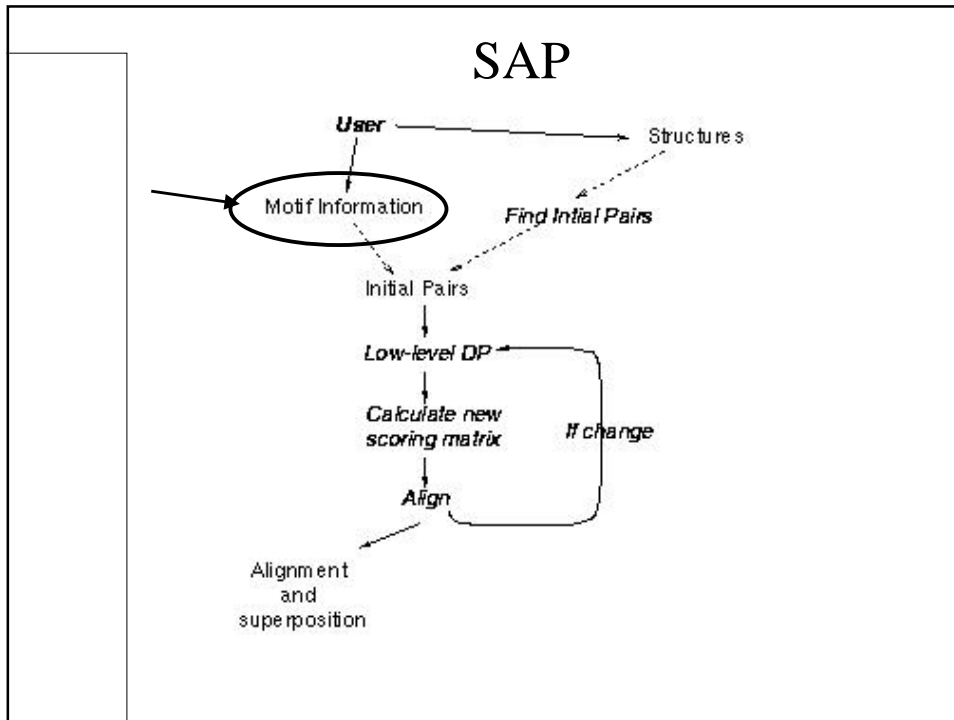


149

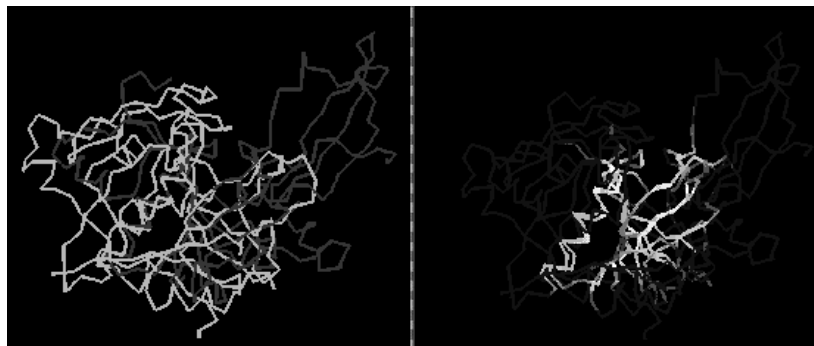
Combining SPratt with SAP



150



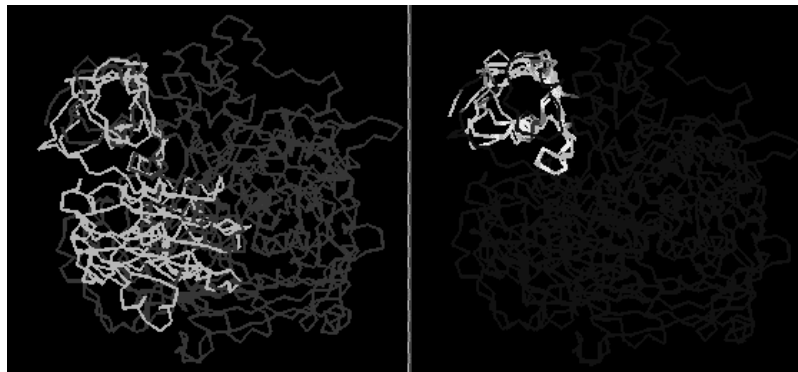
SAP output - cystein proteases



Red: cathepsin-B (1huc),
Green: Human coagulation factor XIII
(1fieA)

Cold (blue): bad superposition
Hot (red, yellow): good fit

SAP output - 2Fe2S Ferredoxins



Red: (1alo),
Green: (2pia)

Cold (blue): bad superposition
Hot (red, yellow): good fit

153

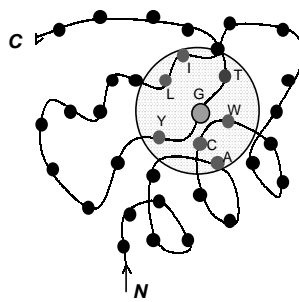
SPratt2

- Same ideas as SPratt in that spatial neighbourhoods are encoded as neighbour strings.
- Instead of using Pratt, a new built-in search procedure is used.
 - Use structural information in search
 - Different (string) pattern solution space
- More efficient
 - can take more structures as input
 - can do more general pattern classes
- Jonassen et al, German Conference on Bioinformatics 2000.

154

Neighbour strings in SPratt2

- One neighbour string per residue including all residues within D Ångström in N-to-C direction.



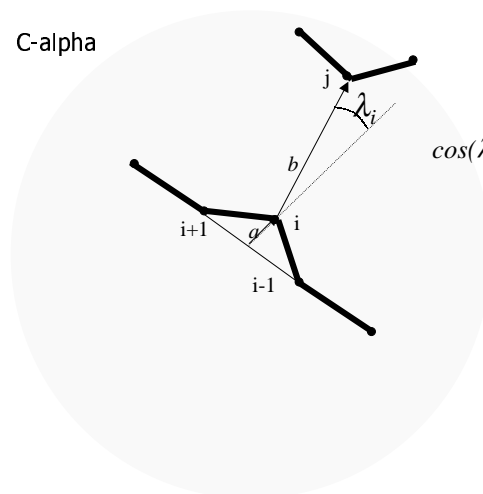
Neighbour string
for G:

ACWILYGT

155

Neighbour String Angle Constraint

- C-alpha



$$\cos(\lambda_i) = a \cdot b / (|a| |b|)$$

*Include residue j in
 i 's neighbour string
only if $\lambda_i > 0$ and $\lambda_j > 0$
(or another threshold)*

156

Neighbour String *-Patterns

- String patterns consist of
 - single amino acids
 - match sets
 - * which matches an arbitrary number of consecutive amino acids
- Example:
 - S*[DN]*H*F*C
- Different from SPratt that uses Prosite style patterns, e.g.
 - S-x(2,3)-[DN]-x(3,5)-H-x-F-x(8,10)-C

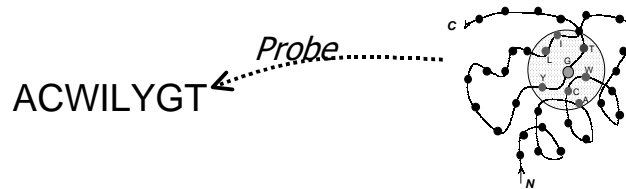
157

Search Problem in SPratt2

- Find *-patterns common to neighbour strings of at least k structures
- Each patterns contains one “center residue” - matching neighbour strings have their center residue aligned with that of the pattern.
- The structural conformation of the residues in the occurrences of a pattern should be similar.

158

Probe \rightarrow *-Pattern



Search problem:

- Find *-pattern generalisation of the probe so that
- it matches neighbour strings from at least k structures
 - the structures of the matches can be superposed onto the probe structure within d angstrom

159

*-Pattern Exploration Seeded by Probe



160

Use of Probe Structure in Search

- Matches that cannot be superposed onto the probe within d angstrom, are discarded - not counted.
- Internal distance RMSd is used since this is easy to compute.
- Used only when *-patterns have 4 or more components (limit to be explored with respect to speed and sensitivity).

161

Which Probes?

- If N structures are analysed and patterns matching at least k structures are sought, use as probes all neighbour strings from the smallest $n-k+1$ structures.
- If a query structure is given, generate probes only from this.

162

Ranking of patterns

- The neighbour string pattern has an information content I .
- The matches can be superposed onto each other giving a matrix of RMSd values.

$$Score = \frac{I}{\max RMSD}$$

- Increases with increasing number of residues in pattern
- Decreases with softening amino acid constraints
- Decreases with poorer structural fit

163

Implementation

- *One* program - string pattern discovery has access to structure information
- Minimize memory usage
 - output patterns as they are found - do post-processing externally
- Allows mining of non-redundant subset of PDB
 - e.g., more than 2000 chains simultaneously

164

Performance of SPratt2 compared to that of SPratt

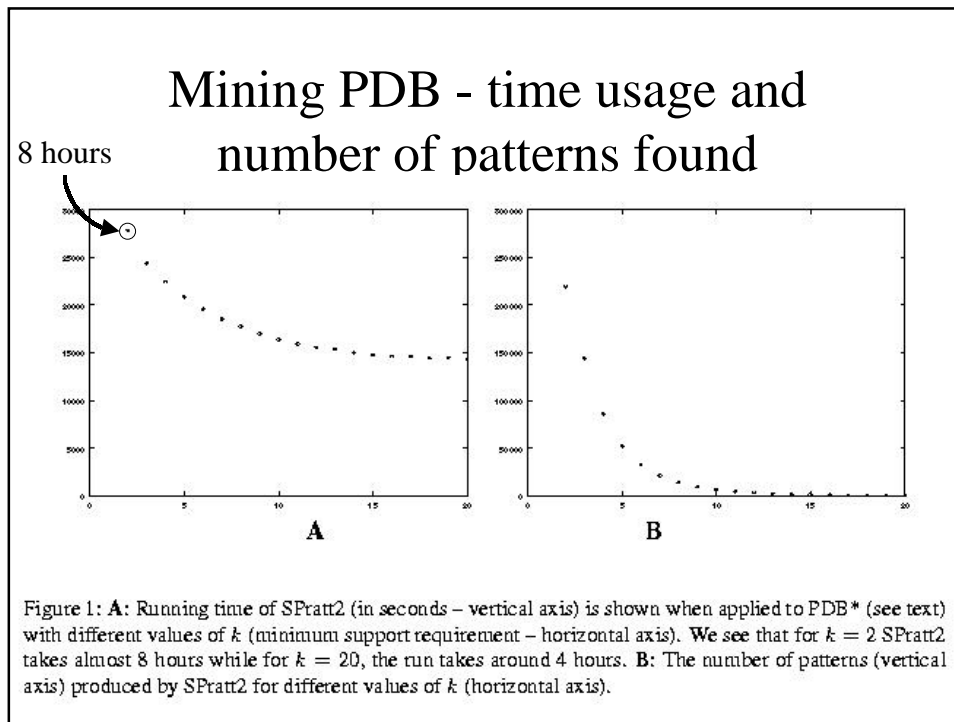
Family	<i>k/N</i>	av. len	Time
Cupredoxins	10/10	263	2 min → 6 sec
Cys. prot.	5/5	560	58 min → 16 sec
4Fe-4S	5/5	86	11 sec → 1 sec
Ser. Prot	10/12	231	12 min → 57 sec

165

Mining PDB

- Maximum 30% identity non-redundant subset of PDB, resolution 2.0 or better: 779 chains (culledPDB, 18/5/00)
- Radius 10 Ångstrøm
- Min-angle: $\lambda > 0$
- Max. RMSd: 1 Å
- Memory usage: < 200 Mbyte
- *k*: minimum nr matching structures

166



Need to Reduce Pattern Set: Select a “Covering” Subset of Patterns

- Discard patterns with score < 20
- Pick pattern with maximum
 - Score
 - Support (number of matching structures)
- Remove patterns matching the same structures.
- Pick maximum pattern in remaining pattern set.
- Keep going until no more patterns to be picked.
- “Representative subset” of the patterns - remove patterns matching the same structure set

Conclusions

- Protein 3D structures are more complex than sequences
- Protein structure comparison and alignment is complex
- We have
 - presented a framework
 - reviewed *some* methods
- A lot of methods out there

- Have not touched too much on
 - scoring, assessment

169

Conclusions (cont'd)

- Motif discovery useful alternative approach to alignment
 - can identify patterns not significant when found between two structures
 - avoids laborious and in-direct pairwise comparisons

- Motif discovery by (unsupervised) data mining feasible

- Benchmarks could be useful, but difficult to find set of common interest
 - Protein family databases (esp. SCOP) can be used for assessment

170

Open Problems and Future Directions

- Will soon have all (most) protein structures (experimentally or by homology modelling).
- Gives increased importance to structure comparison, classification, and motif discovery.
- Need more powerful methods for
 - analysing large data sets
 - finding subtle (remote) similarities

171

Acknowledgments

- Willie Taylor (NIMR London and UoB)
- Darrell Conklin (Zymogenetics, Seattle, USA)
- David Gilbert (City University, London, UK)
- Alvis Brazma (EMBL-EBI, Cambridge, UK)

<http://www.ii.uib.no/forskningsgrupper/bio>

172