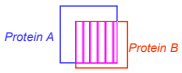
## DALI Method

- Distance mAtrix aLIgnment
- Liisa Holm and Chris Sander, “Protein structure comparison by alignment of distance matrices”, *Journal of Molecular Biology Vol. 233*, 1993.
- Liisa Holm and Chris Sander, “Mapping the protein universe”, *Science Vol. 273*, 1996.
- Liisa Holm and Chris Sander, “Alignment of three-dimensional protein structures: network server for database searching”, *Methods in Enzymology Vol. 266*, 1996.

## How DALI Works?

- Based on fact: similar 3D structures have similar intra-molecular distances.
- Background idea
  - Represent each protein as a 2D matrix storing intra-molecular distance.
  - Place one matrix on top of another and slide vertically and horizontally – until a common the sub-matrix with the best match is found.

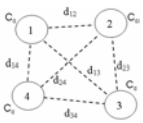



- Actual implementation
  - Break each matrix into small sub-matrices of fixed size.
  - Pair-up similar sub-matrices (one from each protein).
  - Assemble the sub-matrix pairs to get the overall alignment.

## Structure Representation of DALI

- 3D shape is described with a *distance matrix* which stores all *intra-molecular distances* between the $C_\alpha$ atoms.
- Distance matrix is independent of coordinate frame.
- Contains enough information to re-construct the 3D coordinates.

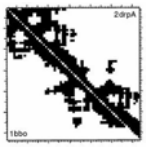
*Protein A*



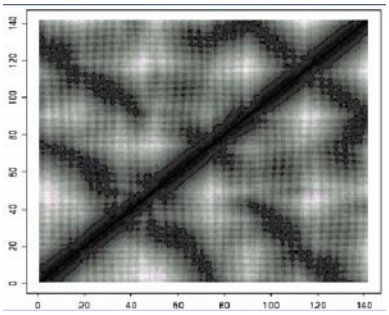

*Distance matrix for Protein A*

| | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | 0 | $d_{12}$ | $d_{13}$ | $d_{14}$ |
| 2 | $d_{12}$ | 0 | $d_{23}$ | $d_{24}$ |
| 3 | $d_{13}$ | $d_{23}$ | 0 | $d_{34}$ |
| 4 | $d_{14}$ | $d_{24}$ | $d_{34}$ | 0 |

*Distance matrix for 2drpA and 1bbo*




## Intra-molecular distance for myoglobin



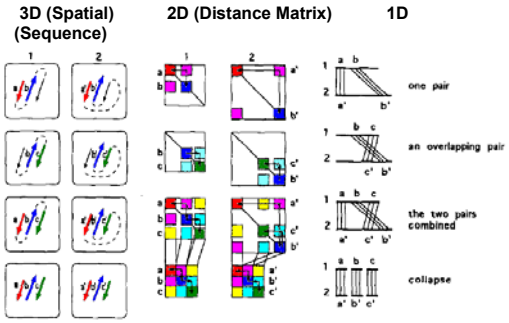

## DALI Algorithm

1. Decompose distance matrix into elementary *contact patterns* (sub-matrices of fixed size)
   - Use hexapeptide-hexapeptide contact patterns.
2. Compare contact patterns (pair-wise), and store the matching pairs in *pair list.*
3. Assemble pairs in the correct order to yield the overall alignment.

## Assembly of Alignments

- Non-trivial combinatory problem.
- Assembled in the manner (AB) – (A’B’), (BC) – (B’C’), . . . (i.e., having one overlapping segment with the previous alignment)
- Available Alignment Methods:
  - *Monte Carlo* optimization
  - Brach-and-bound
  - Neighbor walk

## Schematic View of DALI Algorithm

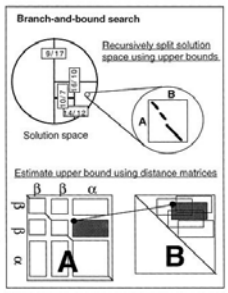



## *Monte Carlo* Optimization

- Used in the earlier versions of DALI.
- Algorithm
  - Compute a similarity score for the current alignment.
  - Make a random trial change to the current alignment (adding a new pair or deleting an existing pair).
  - Compute the change in the score ($\Delta S$).
  - If $\Delta S > 0$, the move is always accepted.
  - If $\Delta S <= 0$, the move may be accepted by the probability $\exp(\beta * \Delta S)$, where $\beta$ is a parameter.
  - Once a move is accepted, the change in the alignment becomes permanent.
  - This procedure is iterated until there is no further change in the score, i.e., the system is converged.

## Branch-and-bound method

- Used in the later versions of DALI.
- Based on *Lathrop and Smith's* (1996) *threading* (sequence-structure alignment) algorithm.
- *Solution space* consists of all possible placements of residues in protein A relative to the segment of residues of protein B.
- The algorithm recursively split the solution space that yields the highest upper bound of the similarity score until there is a single alignment trace left.




## LOCK

- Uses a hierarchical approach
- Larger secondary structures such as helixes and strands are represented using vectors and dealt with first
- Atoms are dealt with afterwards
- Assumes large secondary structures provide most stability and function to a protein, and are most likely to be preserved during evolution

## LOCK (Contd.)

- Key algorithm steps:
  1. Represent secondary structures as vectors
  2. Obtain initial superposition by computing local alignment of the secondary structure vectors (using dynamic programming)
  3. Compute atomic superposition by performing a greedy search to try to minimize *root mean square deviation* (a RMS distance measure) between pairs of nearest atoms from the two proteins
  4. Identify "core" (well aligned) atoms and try to improve their superposition (possibly at the cost of degrading superposition of non-core atoms)
- Steps 2, 3, and 4 require iteration at each step

## Alignment of SSEs

- Define an orientation-dependent score and an orientation-independent score between SSE vectors.
- For every pair of query vectors, find all pairs of vectors in database protein that align with a score above a threshold. Two of these vectors must be adjacent. Use orientation independent scores.
- For each set of four vectors from previous step, find the transformation minimizing rmsd. Apply this transformation to the query.
- Run dynamic programming using both orientation-dependent and orientation-independent scores to find the best local alignment.
- Compute and apply the transformation from the best local alignment.
- Superpose in order to minimize rmsd.

## Atomic superposition

- Loop
  - find matching pairs of $C_\alpha$ atoms
  - use only those within 3 A
  - find best alignment
- until rmsd does not change

## Core identification

- Loop
  - find the best core (symmetric nns) and align; remove the rest
- until rmsd does not change

## VAST

- Begin with a set of nodes (a,x) where SSEs a and x are of the same type
- Add an edge between (a,x) and (b,y) if angle and distance between (a,b) is same as between (x,y)
- Find the maximal clique in this graph; this forms the initial SSE alignment
- Extend the initial alignment to $C_\alpha$ atoms using Gibbs sampling
- Report statistics on this match

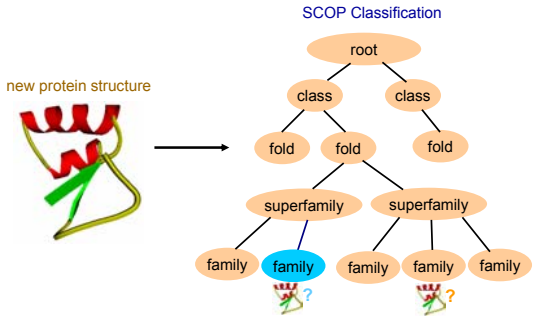
## Quality of a structure match

- Statistical theory similar to BLAST
- Compare the likelihood of a match as compared to a random match
- Less agreement regarding score matrix
  - z-scores of CE, DALI, and VAST may not be compatible

## Protein Structure Classification

- Protein structure classification
  - CATH
  - SCOP
  - FSSP
- Up-to-date view of the protein structure universe
  - SCOP is updated every six months.
  - Determining SCOP classifications of protein structures automatically as they are published in Protein Data Bank (PDB).

## Problem definition

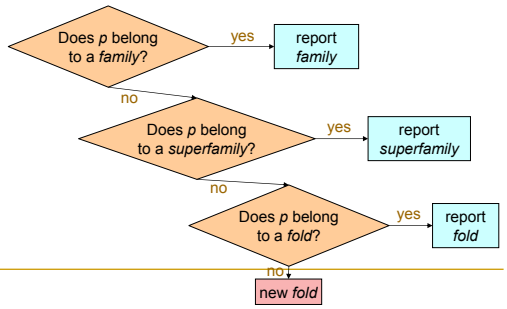


## Two problems

- Class membership?
  - Does the query protein belong to a SCOP category? Or does it need a new category to be defined?
  - Binary classification problem:
    - *member*, *non-member*
- Class label assignment?
  - What SCOP category is the query protein assigned to?
  - Multi-class classification problem

## Hierarchical classification

- Let *p* be a protein structure, proceed bottom-up from *family* level to *fold* level:




## Component classifiers

- Using a sequence/structure comparison tool as a classifier
  - Perform a nearest neighbor query:
    if *similarityScore*(query, NN) < *trained cutoff*
    then not a member of any category
    else member of *class*(NN)
- Comparison tools we have used:

  Sequence: PSI-Blast, HMMER+SUPERFAMILY database

  Structure: CE, Dali, Vast

## Performance of component classifiers

- Database: SCOP 1.59
- Query: SCOP 1.61 – SCOP 1.59

Class membership

| | HMM | BLAST | CE | Dali | Vast | At least one |
|---|---|---|---|---|---|---|
| ***family*** | **94.5%** | **92.6%** | **89%** | **89%** | **89%** | **98.2%** |
| ***superfamily*** | **78.6%** | **66.1%** | **72.2%** | **77.6%** | **78.4%** | **96%** |
| ***fold*** | **73%** | **60.7%** | **78.5%** | **82%** | **85%** | **100%** |

## Performance of component classifiers

- Database: SCOP 1.59
- Query: SCOP 1.61 – SCOP 1.59

Class label assignment

| | HMM | BLAST | CE | Dali | Vast | At least one |
|---|---|---|---|---|---|---|
| ***family*** | **94.8%** | **92.3%** | **91%** | **88%** | **92%** | **97.9%** |
| ***superfamily*** | **69%** | **12%** | **81%** | **80.4%** | **81.7%** | **93.9%** |
| ***fold*** | **40.5%** | **0%** | **40.5%** | **46%** | **54%** | **64.9%** |

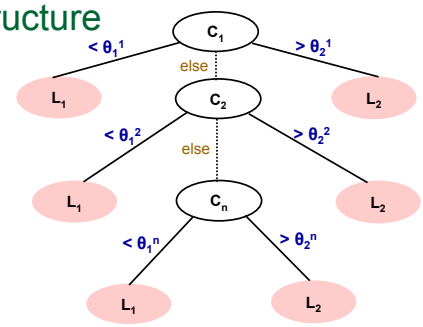
## Normalization of similarity scores

- Universal confidence levels instead of tool-specific scores
- Perform nearest neighbor queries
  - Database: SCOP 1.59
  - Query: SCOP 1.61 – SCOP 1.59
- Partition score space of tools into confidence levels
  - e.g. CE *z-score* of 5.4 → we are 80% confident that the query protein is a member of an existing fold.

## Consensus Decision

- Each component classifier reports a confidence level for the query protein:
  - $c = [C_1, C_2, C_3, C_4, C_5]$
- What is the best way to combine these probabilistic decisions?
  - A solution: decision trees.
  - Decision trees:
    - Attribute order?
    - Branching factor?

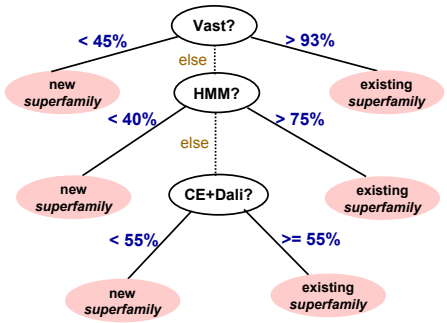
## Proposed decision tree structure




## Determination of $C_i$s and $\theta_j^i$s

- Automated
  - Generate all possible trees of height 3 and $C_i$s as *sum* rules of up to 3 components.
  - Determine $\theta_j^i$s using a greedy optimization that minimizes impurities of nodes level by level.
  - Disadvantage: overfits the data
- Manual
  - Determine $C_i$s by examining individual component's performances
  - Determine $\theta_j^i$s considering two levels of the tree simultaneously and considering only the values between score clusters to avoid overfitting.

## decision tree: *superfamily* level

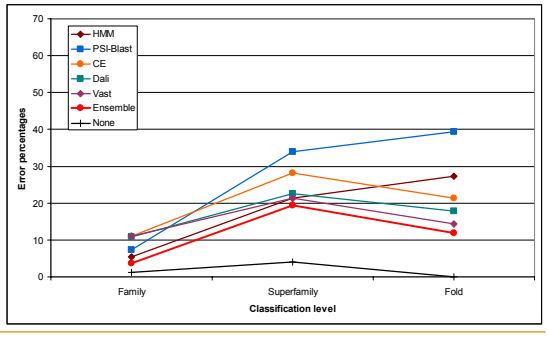



## Experimental evaluation

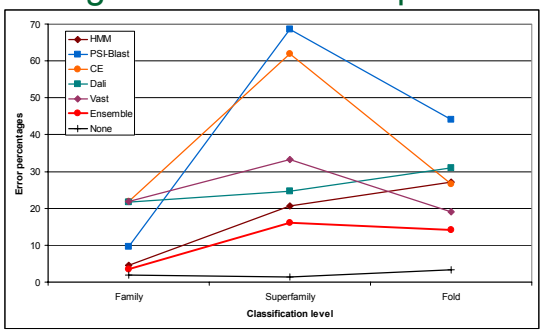
- The dataset:

| | Training | Evaluation |
|---|---|---|
| Database | **v1.59 (*20449*)** | **v1.61 (*22724*)** |
| Query | **v1.61 – v1.59 (*2241*)** | **v1.63 – v1.59 (*2825*)** |
| **new *family*** | ***248*** | ***618*** |
| **new *superfamily*** | ***84*** | ***424*** |
| **new *fold*** | ***47*** | ***339*** |

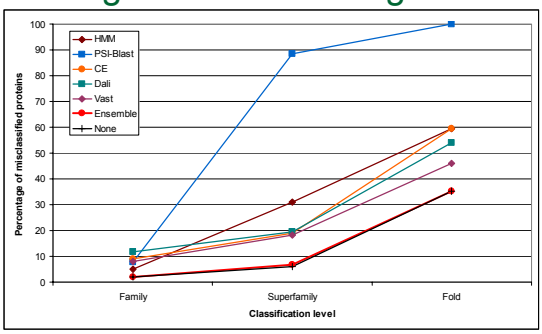
## Training: class membership



## Testing: class membership




## Training: class label assignment




## Testing: class label assignment