Protein Docking and Molecular Shape Recognition using Polar Fourier Correlations

Dave Ritchie
INRIA Nancy – Grand Est

Why is Protein Docking Difficult?

- Protein docking = predicting protein interactions at the molecular level

- If proteins are rigid => six-dimensional search space
- But proteins are flexible => multi-dimensional space!
- Modeling protein-protein interactions accurately is difficult!

Protein Docking – To Predict Protein-Protein Interactions

- Protein-protein interactions (PPIs) define the “machinery” of life

- Humans have about 30,000 proteins, each having about 5 PPIs
- Understanding PPIs could lead to immense scientific advances
- Controlling PPIs could have huge therapeutic benefits (new drug molecules)

Protein Docking Using Fast Fourier Transforms

- Conventional approaches digitise proteins into 3D Cartesian grids...

Katchalski-Katzir et al. (1992) PNAS, 89 2195–2199

- ...and use FFTs to calculated TRANSLATIONAL correlations:

\[ C[\Delta x, \Delta y, \Delta z] = \sum_{x,y,z} A[x, y, z] \times B[x + \Delta x, y + \Delta y, z + \Delta z] \]

- BUT for docking, have to REPEAT for many rotations – EXPENSIVE!
- POLAR coords allow ROTATIONAL nature of problem to be exploited
Some Theory – 2D Spherical Harmonic Surfaces

Use spherical harmonics (SHs) as orthogonal shape “building blocks”

- Reals SHs $y_{lm}(\theta, \phi)$, and coefficients $a_{lm}$
- Encode distance from origin as SH series to order $L$
  \[ r(\theta, \phi) = \sum_{L=0}^{L} \sum_{l=-L}^{L} a_{lm} y_{lm}(\theta, \phi) \]
- Calculate coefficients by numerical integration
- ROTATIONS: $a_{lm}' = \sum_{m'=-l}^{l} R_{lm}(m', \alpha, \beta, \gamma) a_{lm}$

Good for shape-matching, not so good for docking...

Ritchie and Kemp (1999), J. Comp. Chem. 20 383–395

Small-Molecule Virtual Screening using SH Surface Shapes

Targeting AIDS using small-molecule HIV “entry-blockers”

Example – we analysed 602 known inhibitors for the CCR5 cell surface protein

We clustered the inhibitors into 4 main groups which bind in 3 CCR5 sub-pockets
This will help in the design of new HIV entry-blocking molecules


Docking Needs a 3D “Spherical Polar Fourier” Representation

Need to introduce special orthonormal Laguerre-Gaussian radial functions, $R_{nl}(r)$

\[ R_{nl}(r) = N_{nl} e^{-\rho/2} L_{n-1/2}(\rho); \quad \rho = r^2/q, \quad q = 20 \]

Surface Skin:
\[ \sigma(r) = \begin{cases} 1; & r \in \text{surface skin} \\ 0; & \text{otherwise} \end{cases} \]

Interior:
\[ \tau(r) = \begin{cases} 1; & r \in \text{protein atom} \\ 0; & \text{otherwise} \end{cases} \]

Parametrise as:
\[ \sigma(r) = \sum_{n=1}^{N} \sum_{l=0}^{n-1} \sum_{m=-l}^{l} a_{nlm} R_{nl}(r) y_{lm}(\theta, \phi) \]

Translations:
\[ a_{nlm}' = \sum_{n'=0}^{N} \sum_{l'=0}^{n'-1} \sum_{m'=-l'}^{l'} R_{nlm}(n', \alpha, \beta, \gamma) a_{nlm} \]

SPF Protein Shape-Density Reconstruction

Interior density:
\[ \tau(r) = \sum_{n=0}^{N} \sum_{l=0}^{n} a_{nlm}'' R_{nl}(r) y_{lm}(\theta, \phi) \]

\begin{tabular}{|c|c|c|}
\hline
Image & Order & Coefficients \\
\hline
A & Gaussians & - \\
B & N = 16 & 1,496 \\
C & N = 25 & 5,525 \\
D & N = 30 & 9,455 \\
\hline
\end{tabular}


3D-Blast – Comparing Protein Fold Family Consensus Shapes

• Sequence-independent protein structure comparison and clustering

http://threeddblast.loria.fr

Mavridis et al. (2012), Proteins, 80, 530–545

Protein Docking Using SPF Density Functions

\[ \int (\sigma_A(\tau_A)\tau_B(\tau_B) + \tau_A(\tau_A)\sigma_B(\tau_B))dV \]

\[ \int \tau_A(\tau_A)\tau_B(\tau_B)dV \]

Favourable:

\[ S_{AB} = \int (\sigma_A(\tau_B) + \tau_A\sigma_B - Q\tau_A\tau_B)dV \]

Penalty Factor: \( Q = 11 \)

Unfavourable:

\[ S_{AB} = \sum a_{nlm}\sigma_{nlm} + a_{nlm}(b_{nlm} + Qb_{nlm}) \]

Orthogonality:

\[ S_{AB} = \sum a_{nlm}\sigma_{nlm} + a_{nlm}(b_{nlm} + Qb_{nlm}) \]

Search: 6D space = 1 distance + 5 Euler rotations: \( (R, \beta_A, \gamma_A, \alpha_B, \beta_B, \gamma_B) \)


The CAPRI Experiment (Critical Assessment of PRedicted Interactions)

<table>
<thead>
<tr>
<th>Predictor</th>
<th>Software</th>
<th>Algorithm</th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>T4</th>
<th>T5</th>
<th>T6</th>
<th>T7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abagyan</td>
<td>ICM</td>
<td>FF</td>
<td>*</td>
<td>**</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Camacho</td>
<td>CHARMM</td>
<td>FF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Eisenstein</td>
<td>MolFit</td>
<td>FFT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sternberg</td>
<td>FTDOCK</td>
<td>FFT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ten Eyck</td>
<td>DOT</td>
<td>FFT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gray</td>
<td>MC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ritchie</td>
<td>Hex</td>
<td>SPF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Weng</td>
<td>ZDOCK</td>
<td>FFT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wolfson</td>
<td>BUDDA/PPD</td>
<td>GH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bates</td>
<td>Guided Docking</td>
<td>FF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Palma</td>
<td>BIGGER</td>
<td>GF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gardiner</td>
<td>GAPDOCK</td>
<td>GA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Olson</td>
<td>SurfDock</td>
<td>SH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Valencia</td>
<td>ANN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vakser</td>
<td>GRAMM</td>
<td>FFT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* low, ** medium, *** high accuracy prediction; — no prediction


```
Example: best prediction for CAPRI Target 3 – Hemagglutinin/HC63
```


**Best Hex Orientation for Target 6 – Amylase/AMD9**

![Image of molecular structure]

**Subsequent CAPRI Targets 8 – 19**

<table>
<thead>
<tr>
<th>Target</th>
<th>Description</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>T8</td>
<td>Nidogen-γ3 - Laminin</td>
<td>U/U</td>
</tr>
<tr>
<td>T9</td>
<td>LiCT homodimer</td>
<td>build from monomer – 12Å RMS deviation</td>
</tr>
<tr>
<td>T10</td>
<td>TBEV trimer</td>
<td>build from monomer – 11Å RMS deviation</td>
</tr>
<tr>
<td>T11</td>
<td>Cohesin - dockerin</td>
<td>U/U; model-build dockerin</td>
</tr>
<tr>
<td>T12</td>
<td>Cohesin - dockerin</td>
<td>U/B</td>
</tr>
<tr>
<td>T13</td>
<td>SAG1 - antibody Fab</td>
<td>SAG1 conformational change: 10Å RMS</td>
</tr>
<tr>
<td>T14</td>
<td>MYPT1 - PP1 δ</td>
<td>U/U; model-build PP1 α → PP1 δ</td>
</tr>
<tr>
<td>T18</td>
<td>TAXI - xylanase</td>
<td>U/B</td>
</tr>
<tr>
<td>T19</td>
<td>Ovine prion - antibody Fab</td>
<td>model-build prion</td>
</tr>
</tbody>
</table>

T15-T17 cancelled: solutions were on-line & found by Google !!!

T11, T14, T19 involved homology model-building step...

**Exploiting Proir Knowledge in SPF Docking**

Knowledge of even only one key residue can reduce search space enormously...

This accelerates the calculation and helps to reduce false-positive predictions


<table>
<thead>
<tr>
<th>Predictor</th>
<th>Software</th>
<th>T8</th>
<th>T9</th>
<th>T10</th>
<th>T11</th>
<th>T12</th>
<th>T13</th>
<th>T14</th>
<th>T15</th>
<th>T16</th>
<th>T17</th>
<th>T18</th>
<th>T19</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abagyan</td>
<td>ICM</td>
<td>**</td>
<td>*</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
</tr>
<tr>
<td>Wolfson</td>
<td>PatchDock</td>
<td>***</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>**</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>Weng</td>
<td>ZDOCK/RDOCK</td>
<td>**</td>
<td>*</td>
<td>**</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
</tr>
<tr>
<td>Bates</td>
<td>FTDOCK</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>Baker</td>
<td>RosettaDock</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>**</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
</tr>
<tr>
<td>Camacho</td>
<td>SmoothDock</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
</tr>
<tr>
<td>Gray</td>
<td>RosettaDock</td>
<td>***</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>Comeau</td>
<td>ClusPro</td>
<td>**</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>Sternberg</td>
<td>3D-DOCK</td>
<td>**</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>Eisenstein</td>
<td>MolFit</td>
<td>***</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>Ritchie</td>
<td>Hex</td>
<td>***</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>Zhou</td>
<td>DOT</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Ten Eyck</td>
<td>DOT</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Zacharias</td>
<td>ATTRACT</td>
<td>**</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>Valencia</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Vakser</td>
<td>GRAMM</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Umeyama</td>
<td>-</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>Kaznessis</td>
<td>-</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>Fano</td>
<td>Grid-Hex</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

*Aendez et al. (2005) Proteins Struct. Funct. Bioinf. 60 150-169*
“Hex” and “HexServer”

- Hex – interactive docking – http://hex.loria.fr/ – about 25,000 downloads
- Hexserver – http://hexserver.loria.fr/ – about 1,000 docking jobs per month


Macindoe et al. (2010), Nucleic acids Research, 38 W445–W449

Inside Hex – High Order FFTs, Multi-threading on GPUs

The SPF gives an analytic way to calculate TRANSLATIONAL + ROTATIONAL correlations:

\[ S_{AB} = \sum_{j=1}^{\infty} \sum_{l=1}^{\infty} A_{lm}^{(j)} B_{lm}^{(k)} \left( R \right) A_{lm}^{(j)} e^{-i(\beta A' + \gamma A' + \delta B' + \epsilon B')} \]

This allows high order FFTs to be used – 1D, 3D, and 5D
also allows calculations to be easily ported to modern GPUs
- Up to 512 arithmetic “cores”
- Up to 6 Gb memory
- Easy API with C++ syntax
- Grid of threads model (“SIMT”)

UT – for best results, need to understand the hardware...

Ritchie, Kozakov, Vajda (2008), Bioinformatics 24 1865–1873
Ritchie and Venkatraman (2010), Bioinformatics, 26, 2398–2405

CUDA Device Architecture

- Typically 8–16 multiprocessor blocks, each with 16 thread units

- Typically 8–16 multiprocessor blocks, each with 16 thread units

NB. only a very small amount of fast shared memory is available
NB. global memory is ~ 80x slower than shared memory
Strategy: aim for “high arithmetic intensity” in shared memory

CUDA Programming Example - Matrix Multiplication

- Matrix multiplication C = A * B
- Each thread is responsible for calculating one element: C[i,k]

\[ C[i, k] = \sum_{j=1}^{m} A[i,j] \times B[j,k] \]

- Conventional algorithm: rows and columns
- Thread-block algorithm working on TILES

- A tile size of 16x16 is just right!

- Threads co-operate by reading & sharing tiles of A & B
- Multi-processor launches multiple blocks to compute all of C
- Executing thread-blocks concurrently hides global memory latency
CUDA Programming Example – Matrix Multiplication Kernel

```c
__global__ void matmul(int wA, int wB, float *A, float *B, float *C)
{
    float Cik = 0.0; // thread-local result variable
    int bx = blockIdx.x, tx = threadIdx.x; // thread subscripts
    int by = blockIdx.y, ty = threadIdx.y; // ("this" thread is one of a 2-D grid)

    __shared__ float a_sub[16][16], b_sub[16][16]; // declare shared memory
    for (int j=0; j<wA; j+=16) { // thread-local loop over tiles of A and B
        int ij = (16*by+ty)*wA + (j+tx); // thread-local array subscripts
        int jk = (j+ty)*wB + (16*bx+tx);
        a_sub[ty][tx] = A[ij]; // copy global data to shared memory ("I/O")
        b_sub[ty][tx] = B[jk];
        __syncthreads(); // wait until all memory I/O has finished
        for (int jj=0; jj<16; jj++) {
            Cik += a_sub[ty][jj] * b_sub[jj][tx]; // multiply row*column in current tiles
        }
        __syncthreads(); // synchronise threads before starting more I/O
    }
    C[(16*by+ty)*wB + (16*bx+tx)] = Cik; // copy local result -> global memory
}
```

Results – GPU v’s CPU Docking Performance

Key Hex functions implemented using only 5 or 6 CUDA kernels
1D and 3D FFTs are calculated using Nvidia’s cuFFT library
Here, GPU = Nvidia FX-5800, CPU = Intel i7-965

Hex 1D correlations are up to 100x faster on FX-5800 than on iCore7
Overall, including set-up, Hex 1D FFT is about 45x faster on FX-5800 than on iCore7

Results – Multiple GPUs and CPUs

With Multi-threading, we can use as many GPUs and CPUs as are available

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Kallikrein A / BPTI (233 / 58 residues)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FFT</td>
<td>ZDOCK</td>
</tr>
<tr>
<td>1xCPU</td>
<td>7,172</td>
</tr>
<tr>
<td>4xCPU</td>
<td>1,195</td>
</tr>
</tbody>
</table>

Kallikrein A / BPTI (233 / 58 residues)

For best performance: use 2 GPUs alone, or 6 CPUs plus 2 GPUs
With 2 GPUs, docking takes only about 15 seconds – very important for large-scale!
**Speed Comparison with ZDOCK and PIPER**

Hex: 52000 x 812 rotations, 50 translations (0.8 Å steps)

ZDOCK: 54000 x 6 deg rotations, 92Å 3D grid (1.2 Å cells)

PIPER: 54000 x 6 deg rotations, 128Å 3D grid (1.0Å cells)

Hardware: GTX 285 (240 cores, 1.48 GHz)

<table>
<thead>
<tr>
<th></th>
<th>ZDOCK</th>
<th>PIPER</th>
<th>Hex</th>
<th>Hex</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kallikrein A / BPTI (233 / 58 residues)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FFT 3D</td>
<td>7,172</td>
<td>468,625</td>
<td>26,372</td>
<td>224</td>
</tr>
<tr>
<td>1D</td>
<td>1,195</td>
<td>42,602</td>
<td>2,398</td>
<td>676</td>
</tr>
</tbody>
</table>

# execution times in seconds
(times scaled to two-term potential, as in Hex)

What’s next?

Better energy functions & constraints...

Modeling flexibility...

• Using homology templates...

• Multi-component complexes...

**Knowledge-Based Protein Docking:**

**CAPRI Target 40 (2009) – API-A/Trypsin**

We searched SCOPPI and 3DID for similar domain interactions to the target. This helped to identify two key inhibitory loops on API-A around L87 and K145.

Performing focused Hex + MD refinement gave a total of 9 “acceptable” solutions.

**EigenHex – Flexible Docking using Elastic Network Model (ENM)**

- Apply eigenvector analysis to the top 1,000 Hex orientations

Overall approach

- Cα elastic network model (ENM)

- Sample up to 20 normal mode analysis (NMA) eigenvectors

- Search using particle-swarm optimisation (PSO)

- Score using “DARS” potential

Results so far

- DARS works very well...

- Still need a better scoring function


**The KBDOCK Database and Web Server**

Content: 2,721 non-redundant hetero DDIs involving 1,029 PFAM domain families

For each PFAM family, all DDIs are superposed and spatially clustered

http://kbdock.loria.fr/

Aim: to provide PFAM family-level structural templates for knowledge-based docking
**KBDOCK – Analysis of PFAM Domain Family Binding Sites**

Nearly 70% of PFAM domain families have just one binding site

Very few domains have more than two or three binding sites

This supports the notion that protein binding sites are often re-used...

---

**KBDOCK – Template-Based Protein Docking Results**

The Protein Docking Benchmark 4.0 contains 176 protein-protein complexes

We selected 73 single-domain complexes

A “Full-Homology” (FH) template matches both target domains

A “Semi-Homology” (SH) template matches just one target domain

<table>
<thead>
<tr>
<th>Target class</th>
<th>Total targets</th>
<th>FH templates</th>
<th>Two SH templates</th>
<th>One SH template</th>
<th>Zero templates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Without date filtering</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Enzyme</td>
<td>36</td>
<td>24 / 24</td>
<td>(3 + 1) / 5</td>
<td>3 / 5</td>
<td>2</td>
</tr>
<tr>
<td>Other</td>
<td>37</td>
<td>21 / 21</td>
<td>(0 + 0) / 3</td>
<td>5 / 11</td>
<td>2</td>
</tr>
<tr>
<td>Total</td>
<td>73</td>
<td>45 / 45</td>
<td>(3 + 1) / 8</td>
<td>8 / 16</td>
<td>4</td>
</tr>
</tbody>
</table>

With date filtering

<table>
<thead>
<tr>
<th>Target class</th>
<th>Total targets</th>
<th>FH templates</th>
<th>Two SH templates</th>
<th>One SH template</th>
<th>Zero templates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enzyme</td>
<td>36</td>
<td>13 / 13</td>
<td>(2 + 1) / 5</td>
<td>7 / 11</td>
<td>7</td>
</tr>
<tr>
<td>Other</td>
<td>37</td>
<td>13 / 13</td>
<td>(0 + 0) / 1</td>
<td>8 / 15</td>
<td>8</td>
</tr>
<tr>
<td>Total</td>
<td>73</td>
<td>26 / 26</td>
<td>(2 + 1) / 6</td>
<td>15 / 26</td>
<td>15</td>
</tr>
</tbody>
</table>

If a FH template exists, it is almost always correct

Even if there is no FH template, SH templates are still very useful

Ghoorah et al. (2011), Bioinformatics, 27, 2820–2827

---

**Assembling Multi-Component Protein Complexes**

› Multi-component assembly is a highly combinatorial problem

› How to generate and score candidate orientations efficiently?

› Here, we use Minimum Weight Spanning Trees (MSTs), (Inbar et al., 2003)

› ... with an ant colony particle swarm optimisation (PSO) search algorithm

---

**Minimum Energy Spanning Trees**

• Here, we have \( N = 5 \) proteins and \( K = N(N-1)/2 = 10 \) “edges”

• Each edge should consider many (e.g. \( P = 100 \)) docking solutions

• Naive enumeration would give \( P^{N(N-1)/2} \) possible combinations

• A spanning tree visits each node just once...

• ... there are only \( P^{N-1}N^{N-2} \) distinct spanning trees

• ... and when \( N < P \), we get \( P^{N-1}N^{N-2} \ll P^{N(N-1)/2} \)

• Strategy: search for the minimum energy spanning tree ...

› Getting technical: this is an “edge-weighted K-cardinality” problem...

Inbar et al. (2003), Bioinformatics, 2003, i158–i168
Multi-Component Docking using Ant-Colony Optimisation

Ant colony optimisation is based on the behaviour of real ants. When an ant finds food, it leaves a trail of pheromones. Other ants follow strong pheromone trails to reach the food quickly.

- Here, we use 10 ants in parallel for 1,000 iterations...
- Each ant is assigned to a randomly generated spanning tree.
- It must detect and score steric clashes, and update its trail.
- It then makes a new spanning tree using the latest pheromone trails...

MDOCK – Multi-Component Docking Results

There are not many multi-component examples in the PDB. Therefore, several ‘targets’ were made from the same complex...

1VCB = von Hippel-Lindau ElonginC-ElonginB tumor suppressor protein
1IKN = Transcription factor I-kappa-B-alpha / NF-kappa-B
1K8K = Bovine actin polymerisation initiation complex Arp2 / Arp3

<table>
<thead>
<tr>
<th>Target</th>
<th>Chains</th>
<th>Time (min)</th>
<th>Rank</th>
<th>RMSD (Å)</th>
<th>Best RMSD (Å)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1VCB</td>
<td>A,B,C</td>
<td>43.8</td>
<td>1</td>
<td>0.58</td>
<td>0.58</td>
</tr>
<tr>
<td>1IKN</td>
<td>A,C,D</td>
<td>77.3</td>
<td>1</td>
<td>9.17</td>
<td>0.88</td>
</tr>
<tr>
<td>1K8K</td>
<td>A,B,D,E</td>
<td>123.5</td>
<td>1</td>
<td>4.96</td>
<td>2.19</td>
</tr>
<tr>
<td>1K8K</td>
<td>A,B,D,E,F</td>
<td>168.6</td>
<td>2</td>
<td>9.48</td>
<td>2.99</td>
</tr>
<tr>
<td>1K8K</td>
<td>A,B,D,E,F,G</td>
<td>194.1</td>
<td>15</td>
<td>4.63</td>
<td>3.53</td>
</tr>
<tr>
<td>1K8K</td>
<td>A,B,C,D,E,F,G</td>
<td>366.9</td>
<td>–</td>
<td>–</td>
<td>10.21</td>
</tr>
</tbody>
</table>

Mostly good results, but why did we miss one?

However, it would be very expensive to apply this algorithm to blind docking ...


Conclusions and Future Prospects

(+) Rigid-body docking on a GPU now takes only a few seconds:
- This was implemented using only 5 or 6 GPU kernels
- But a lot of low-level CPU code had to be re-written

(+) Knowledge-based docking is becoming increasingly useful

(-) Modeling protein flexibility during docking is still difficult

(-) Multi-component assembly is a highly combinatorial problem

With SPF correlations, high-throughput shape comparison is now feasible:
- All-vs-all docking ?
- Electron-microscopy density fitting ?
- Assembling multi-component machines ?

(? ) The challenge for this decade – “the structural interactome”

Acknowledgments

BBSRC 1996 – 2000
EPSRC 2000 – 2006
ANR 2009 – 2010
ANR 2011 – 2014

Anisah Ghoorah
Matthieu Chavent
Violeta Pérez-Nueno
Vishwesh Venkatraman
Lazaros Mavridis

People, Papers, Programs:   http://www.loria.fr/~ritchied/
5D FFT Correlations from Complex Overlap Expressions

nplex SHs, $Y_{lm}$:

$$y_{lm}(	heta, \phi) = \sum_t U^{(l)}_{mt} Y_{lt}^{(l)}(\theta, \phi)$$

nplex coefficients:

$$A_{nlm} = \sum_t a_{nt} U^{(l)}_{lt}$$

nplex overlap:

$$S = \sum_{kj,me} D^{(j)}_{me}(0, \beta_A, \gamma_A) A_{kjs}^{(j)} T^{(i)}_{kjs,me}(R) D^{(i)}_{me}(\alpha_B, \beta_B, \gamma_B) B_{nlv}$$

lect coefficients:

$$S_{jls,me}(R) = \sum_{kj} A_{kjs}^{(j)} T^{(i)}_{kjs,me}(R) B_{nlv}$$

give:

$$S = \sum_{jmlve} D^{(j)}_{me}(0, \beta_A, \gamma_A) S^{(i)}_{jls,me}(R) D^{(i)}_{me}(\alpha_B, \beta_B, \gamma_B)$$

and as exponentials:

$$D^{(i)}_{me}(\alpha, \beta, \gamma) = \sum_t \Gamma^m_{te} e^{-i\alpha t} e^{-i\beta \gamma}$$

$$S = \sum_{jmlve} \Gamma^m_{te} S^{(j)}_{jls,me}(R) \Gamma^m_{te} e^{-i(r(\beta_A - \gamma_A + \alpha_B) + t(\beta_B + \gamma_B))}$$


Translation Matrices From Fourier-Bessel Transform Theory

Using spherical Bessel transforms:

$$\tilde{R}_{nl}(\beta) = \sqrt{\frac{2}{\pi}} \int_0^\infty R_{nl}(r) j_l(\beta r) r^2 dr; \quad R_{nl}(\beta) = \sqrt{\frac{2}{\pi}} \int_0^\infty \tilde{R}_{nl}(\beta) j_l(\beta r) r^2 d\beta$$

Can be shown that

$$T^{(j)}_{nlmi}(R) = \sum_{k=[l-L]}^{l+L} A^{(j)}_{k}^{[m]} \int_0^\infty \tilde{R}_{nl}(\beta) \tilde{R}_{nl'}(\beta) j_k(\beta R) \beta^2 d\beta$$

where

$$A^{(j)}_{k}^{[m]} = (-1)^{\frac{L+L'}{2}+m} \frac{(2k+1)[(2l+1)(2l'+1)]^{1/2}}{m \tau 0} \begin{pmatrix} l & l' & k \\ 0 & 0 & 0 \\ m & \tau & 0 \end{pmatrix}$$

Can derive analytic formulae for both GTO and ETO radial functions

GPU Implementation – Rotate and Translate Protein A

On CPU, calculate multiple $(\beta_A, \gamma_A)$ rotations of protein A

On CPU, re-index translation matrices and rotated coefficients into regular sparse arrays

On GPU, translate multiple protein A coefficients using tiled matrix multiplication

Can derive analytic formulae for both GTO and ETO radial functions

Requires high precision math library (GMP)...
GPU Implementation – Perform Multiple FFTs

Next, calculate multiple 1D FFTs of the form:

\[ S_{AB}(\alpha_B) = \sum_{m} \sum_{nl} e^{-im\alpha_B} A_{nlm}^*(R, \beta_A, \gamma_A) \times B_{nlm}^*(\beta_B, \gamma_B) \]

- On GPU, cross-multiply transformed A with rotated B coefficients (as above)
- On GPU, perform batch of 1D FFTs using cuFFT and save best orientations

3D FFTs in \((\alpha_B, \beta_B, \gamma_B)\) can be calculated in a similar way...