Parallel Protein Docking Tool

Ivan Sović Nino Antulov-Fantulin Igor Čanadi
Matija Piškorec Mile Šikić

Faculty of Electrical Engineering and Computing
University of Zagreb, Unska 3, 10000 Zagreb, Croatia
E-mail: ivan.sovic@gmail.com, nino.naf@gmail.com, igor.canadi@gmail.com
matija.piskorec@gmail.com, mile.sikic@fer.hr

February 22, 2010.

Abstract - Parallel Protein Docking Tool (PPDT) system is performing a shape based process of protein docking using spherical harmonics. The software is made from scratch in C++. Input data are two PDB (Protein Data Bank) files and desired docking parameters. The PDB files hold information about protein's atom coordinates in three dimensional space. Protein structures are transformed with spherical functions which allow fast rotation and translation of 3D shapes in space. By translation and rotation (with a given resolution of angles and distance) the surface complementarity is tested for each orientation of two proteins in 3D space. The orientations are then ranked according to the given scores. The output PDB file now holds orientations with best scores ready for visualization. As a part of the system a tool for visualization of molecules was developed. It's purpose is to provide a visualization of molecular structure in all phases of protein docking process. The visualizer also allows interactive translation and rotation of input proteins and evaluation of their complementarity in real time.

Although docking tools that use spherical harmonics already exist, we implemented a parallel version using MPI (message passing interface). In this implementation any worker MPI process independently makes docking search on six dimensional subspace determined by parameters in its task. PPDT has high scalability, modularity, variable granularity of tasks and provides significant speedup comparing to a non-parallel version of software.

Keywords - protein docking, spherical harmonics, PDB, MPI

I. Introduction

Majority of methods for protein docking include search over all possible conformations of two protein molecules and finding the conformations that meet some physically favourable criteria - surface complementarity being one of the simplest. Usually it includes searching over three rotational and three translational degrees of freedom. PPDT instead implements search over five rotational and one translational degree of freedom to exploit the features of SPF (Spherical Polar Fourier) correlations. The method is described by Ritchie and Kemp [3, 6] and allows fast correlation of various surface characteristics - geometry and electrostatics for example.

Our goal is to find the best few relative positions of two proteins. To do that, we need to:

- Discretize all the possible relative positions of two proteins. (search space)
- Evaluate one relative position of two proteins. (evaluation function)

Quick overview of SPF method is given in section II., following with the description of parallel MPI implementation in section III.. Verification of docking algorithm on real protein complex and performance analysis are given in section IV..

Most of the visualizations are generated with Vmol [8] - a specialized tool for molecule vizualization.

II. Predicting the complex structure

Problem of predicting the structure of a protein complex consists of testing the quality of all possible conformations that two proteins can find themselves in. The PPDT tool searches for the best geometric complementarity of protein surfaces, but it's also possible to correlate some other properties, such as parts of the surfaces determined by hydrostatics or hydrophobicity. To represent a protein surface PPDT expands it into set of spherical and radial basis functions. In order to be able to test every conformation in the search space, PPDT uses five rotational and one translational degree of freedom.

A. *Input data and data processing*

Protein structures, used as inputs for the docking process, are defined by two PDB (*Protein Data Bank*) files. Every PDB file consists of atom coordinates, bibliographic citations, primary and secondary structures, crystallographic structural factors, NMR experimental data and other information. Paths to the input PDB files, as well as other parameters (i.e. the order of spherical harmonics, output paths for the intermediate results, constants...) are set in configuration file. After loading all of the configuration settings input PDB files are parsed. Also, it is necessary to determine the radius of every atom. With this information, we can sample the surface of each protein using the MSMS program [7]. MSMS rapidly calculates solvent excluded surface of a molecule, and returns the results in a form of vertex coordinates. Vertices, which define the surface polygons, are later on used for the calculation of spherical harmonic coefficients.

B. *Surface representation with spherical harmonics*

Spherical harmonics are functions $y_{lm}(\phi, \theta)$ presented in spherical coordinates (ϕ, θ) with a condition that $m < l$. There are two important reasons why these functions are highly suitable for protein surface representation. First, every function on a sphere can be expressed as a linear combination of spherical harmonic functions:

$$\mu(\phi, \theta) = \sum_{l=0}^{\infty} \sum_{m=-l}^{m=l} a_{lm} y_{lm}(\phi, \theta) \tag{1}$$

In this case, the surface of a protein is parametrised with a set of coefficients $\{a_{0,0}, a_{1,-1}, a_{1,0}, \ldots\}$, instead of explicitly defining vertices on the surface. Second important property is that the space of spherical harmonic functions is closed in respect to rotation, and also that the functions of a specific order l transform amongst themselves in a predictable manner:

$$a_{lm}^{'} = \sum_{m'} a_{lm'} R_{mm'}^{(l)}(\alpha, \beta, \gamma) \tag{2}$$

where α, β and γ are Euler angles, and $R_{mm'}^{(l)}(\alpha, \beta, \gamma)$ is the Wigner rotation matrix as described by Ritchie in [3].

Further, in order to define a volume for the skins, the functions are additionally multiplied with selected radial functions. Suitable radial functions are defined by the following expression:

$$R_{nl}(r) = N_{nl} e^{-\rho/2} \rho^l L_{n+l}^{2l+1}(\rho) \tag{3}$$

where $L_{n+l}^{2l+1}(\rho)$ are Laguerre polynomials, ρ scaled distance to the origin and N_{nl} the normalization coefficient. New three-dimensional base functions are then:

$$F_{nlm}(\underline{r}) = R_n(r) y_{lm}(\phi, \theta) \tag{4}$$

Protein surface $\sigma(\underline{r})$ can now be displayed through development in a new base:

$$\sigma(\underline{r}) = \sum_{nlm}^{N} a_{nlm} F_{nlm}(\underline{r}) \tag{5}$$

In the process of calculation of the spherical harmonic coefficients the numerical integration is performed over a three-dimensional grid. This grid contains rasterized protein surface, obtained as described above.

Now every protein can be represented using two skins - interior and exterior. Interior skin is defined as the union of the van der Waals volumes of all atoms just inside the molecular surface, while the exterior skin is the volume bounded by the molecular and solvent-accessible surfaces.

C. *Search space*

With rotating one protein for the angles (α, β, γ), the other for (β, γ) and with changing the distance of their coordinate systems, we can define all the possible relative positions of two proteins. The icosahedral tessellation of sphere is used for obtaining the uniform distribution of β and γ angles. Angle α and intermolecular distance are chosen uniformly from desired interval.

D. *Rotation*

One of the main reasons for representing the protein surfaces with series of spherical harmonics is that they can be rotated fast. However, since translation is a much slower operation in this representation, we use five rotations and only one translation to cover all possible conformations. Expression for calculation of rotated coefficients for angles β and γ is given by Ritchie in [3]:

$$a'_{nlm} = \sum_{m'=-l}^{l} a_{nlm'} R^{(l)}_{mm'}(\beta, \gamma) \tag{6}$$

Where R is the element (m, m') of rotation matrix with order l and it depends on angles β and γ. For rotating around angle α we use simpler expression:

$$a'_{nlm} = a_{nlm} \cos m\alpha + a_{nl\overline{m}} \sin \overline{m}\alpha \tag{7}$$

E. *Translation*

During translation, only the relative distance of coordinate systems of two proteins is changed. Expression for changing the distance for R units is given by Ritchie in [4]:

$$a'_{nlm} = \sum_{n'l'm'}^{N} a_{n'l'm'} K_{nn'll'}\mathrm{m}(R) \delta_{mm'} \tag{8}$$

where K is a 5D translation matrix which depends on distance R. Calculation of translation matrices is complex and time consuming, so they are precalculated and saved in multiple files, which we load during runtime.

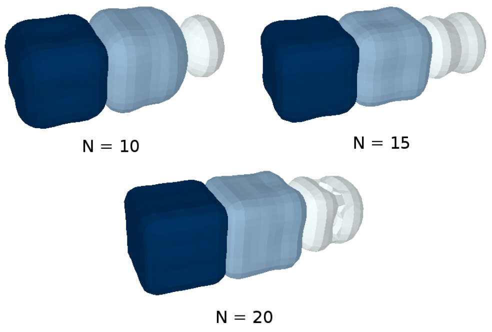

Figure 1: Deformations of original shape during translation are due to the localization properties of used radial functions. Using higher order coefficients helps to reduce them. The render is generated with Vmol [8].

F. *Evaluation function*

How will two proteins interact when they find themselves close to each other depends on various chemical and physical properties. However, when implementing docking, we have to simplify that properties and make them easier for calculations. Ritchie in [3] defines the evaluation function as a volume of intersection of outer and inner skins of protein in a following way:

$$S = \int \rho_A(\underline{r}_A)\tau_B(\underline{r}_B)dV + \int \tau_A(\underline{r}_A)\rho_B(\underline{r}_B)dV - Q\int \tau_A(\underline{r}_A)\tau_B(\underline{r}_B)dV \tag{9}$$

where ρ is the outer skin and τ is the inner skin of a protein. Q is a penalty for penetration of one protein into another (which is physically impossible). In our implementation we use $Q = 15$.

G. *Searching through parameter space*

After defining the rotation and translation of protein skins in SPF representation we are able to search through all possible conformations of two proteins and evaluate shape complementarity of their surfaces for each one. For searching through the parameter space we define the following algorithm:

1. (Rotation of protein A) For each (β_A, γ_A) calculate rotation of protein A and save it in memory.
2. (Rotation of protein B) For each $(\alpha, \beta_B, \gamma_B)$ calculate rotation of protein B and save it in memory.
3. (Translational search) For each molecular distance R do the following ...
   (a) Fetch precalculated (β_A, γ_A) rotation of protein A from memory and translate it R units.
   (b) For each $(\alpha, \beta_B, \gamma_B)$ fetch precalculated rotation of protein B from memory and evaluate the complex of two proteins.
   (c) Repeat last two steps for each (β_A, γ_A) of the protein A.

III. Parallel protein docking

In order to dock two proteins without prior knowledge of possible binding sites one must do finite exhaustive docking search with some predefined granularity over six-dimensional docking space. Six-dimensional docking search space consists of two-dimensional subspace of possible rotations of first protein, three-dimensional subspace of possible rotations of second protein and one-dimensional subspace of possible translations of first protein. Because exhaustive docking search over six-dimensional docking space is time consuming we have implemented parallel version of protein docking tool called *PPDT* by using MPI [2] *(Message Passing Interface)* communication protocol. MPI is used for communication among our processes running in parallel on a distributed memory system.

We have done data decomposition of six-dimensional docking search space into *pool of tasks* in which every task has fixed translation coordinate.

Functional decomposition of MPI processes is made by dividing processes into *master* and *worker* groups. All MPI processes in *master* group have the job of providing unfinished tasks to MPI processes from *worker* group. Master process waits for the new messages of type `get task()` from worker processes and provides an appropriate answer message. To obtain a task worker processes dynamically send messages of type `get task()` to master process. After receiving message `get task()` from worker, master process finds first non finished task in *pool of tasks* and resends tasks identifier to sender. After docking search on task subspace is done worker saves his docking results to disk and tries to get new task from master process. Master process continues to provide tasks as long as there are unfinished tasks in the *pool of tasks.* On emptying the *pool of tasks* master process sends message `stop working()` to all worker processes and assembles docking output from workers docking results.

IV. Results and analysis

Tests of individual parts of the system were performed on a test cube. Visualization of the results of translations were already shown on figure 1 in section E.. Reconstruction of isosurfaces from spherical coefficients is shown on figure 2.

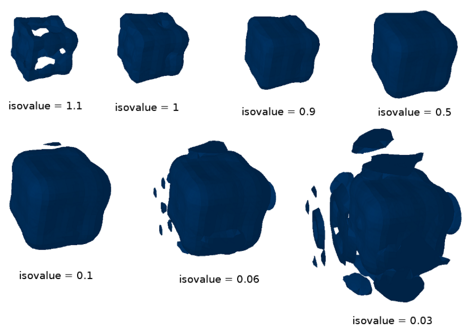

Figure 2: Reconstructing the molecular surface of the test cube from spherical coefficients. Isovalue of around 0.5 gives the required isosurface that approximates original test cube. Visualization is generated using Vmol [8].

Main test is performed on 3fhl complex that can be found in Protein Data Bank [1]. Figure 4 shows maximum score out of all $(\alpha, \beta_1, \gamma_1, \beta_2, \gamma_2)$ conformations on given molecular distance (radius) R.

Considering the molecular sizes of ligand and receptor protein the expected docking distance should be around 31 Å. As shown on figure 4 the maximum score is indeed around 31 Å.

The protein docking calculation is performed in parallel on distributed memory system. Memory occupation of each task can be expressed as

$$n_{conf} \cdot M_{coeff} + M_{trans} \tag{10}$$

where n_{conf} is number of distinct rotational conformations for each protein that are precalculated and stored in memory. It depends on chosen tessellation order T and number of α angles n_α that are investigated for each (β, γ) pair of ligand and receptor.

$$n_{conf} = 20 \cdot 2^{2T} \cdot n_\alpha \tag{11}$$

M_{coeff} is memory size of spherical coefficients and is composed of

$$M_{coeff} = 2 \cdot N \cdot (N+1) \cdot \frac{2N+1}{6} \tag{12}$$

double-size numbers. M_{trans} is the memory size of translation matrix for given order N and molecular distance R and is composed of

$$M_{trans} = N^2 \cdot (N+1)^2 \cdot \frac{2N+1}{12} \tag{13}$$

double-size numbers.

The whole conformation space for testing the 3hfl was composed of 320 distinct (β, γ) pairs for ligand and receptor, 36 α angles for ligand and 100 molecular distances totalling to 1152000 possible conformations that needed to be evaluated. Total time on one processor amounted to 59.22 minutes. The coefficients were of order 15 and occupied 19.38 Kb of memory space and the translation matrices for each molecular distance were 1163 Kb. The total memory occupation for one task was 219 MB.

A. *Scalability analysis*

We define total execution time T of the parallel program to be

$$T = \max\{T^i_{comp} + T^i_{comm} + T^i_{idle}\} \tag{14}$$

where T^i_{comp} is defined as total time spent for computing of processor number i , T^i_{comm} as total time spent for communication of processor number

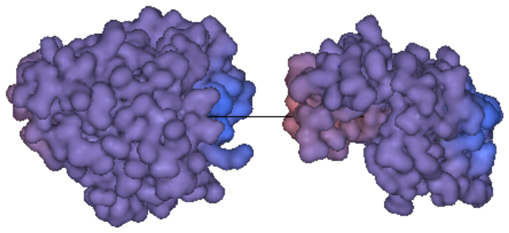

Figure 3: 3hfl complex in it's usual docking conformation. Ligand and receptor are separated to show the binding site. Centers of two molecules are joined with intermolecular axis along which the translation is performed during the docking search. The picture is generated with Hex [5].

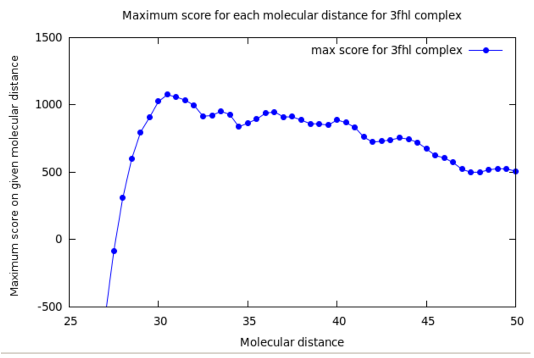

Figure 4: Maximum score for each given radius. Global maximum is reached at molecular distance of around 31 Å.

i and T^i_{idle} as total time spent for waiting of processor number i. Communication time T^i_{comm} is far less than T^i_{comp} and T^i_{idle} so we can reduce formula for total execution time to be

$$T = \max\{T^i_{comp} + T^i_{idle}\} \tag{15}$$

If the number of tasks in the *pool of tasks* is N, number of available processes is p and time for computing one task T_1 on one processor is constant than total execution time 5 is defined by formula

$$T = \lceil \frac{N}{p} \rceil T_1 \tag{16}$$

Parallel algorithm acceleration on figure 6 is defined as a ratio of total time of execution on one processor and total time of execution on p processors.

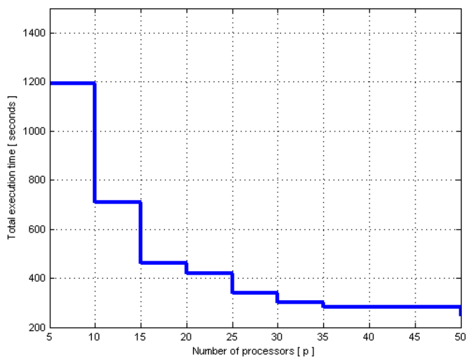

Figure 5: Scalability on variable total time of execution, order 15, N = 40

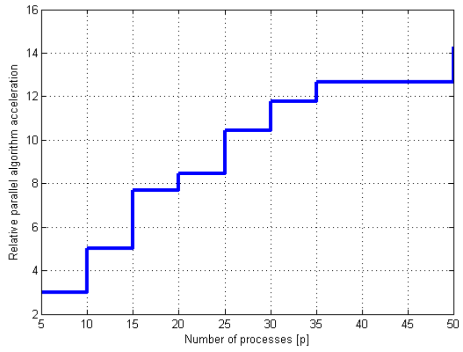

Figure 6: Scalability on variable algorithm acceleration, order 15, N = 40

V. Conclusion

We have presented an implementation of SPF based protein docking and showed that significant speedup could be attained by running docking procedure on multiple processors with MPI protocol. The parallelization is performed by functional decomposition of the one-dimensional translational subspace of the ligand protein. The *master* process creates the *pool of tasks* and dynamically allocates unfinished tasks to the *worker* processes. The method should prove useful in future implementations of the SPF based protein docking methods.

References

[1] N. Berman, Henrick. Worldwide protein data bank, December 2009.

[2] J. Dongarra, D. Walker, E. Lusk, B. Knighten, M. Snir, W. Gropp, E. Lusk, A. Geist, M. Snir, S. Otto, R. Hempel, E. Lusk, W. Gropp, J. Cownie, T. Skjellum, L. Clarke, M. Snir, R. Littlefield, M. Sears, and S. Huss-Lederman. The message passing interface (mpi) standard, June 1995.

[3] D. Ritchie. Parametric protein shape recognition. Phd thesis, University of Aberdeen, September 1998.

[4] D. Ritchie. High-order analytic translation matrix elements for real-space six-dimensional polar fourier correlations. *Journal of Applied Crystallography*, 2005.

[5] D. Ritchie. Hex protein docking 5.0, December 2009.

[6] D. Ritchie and G. Kemp. Protein docking using spherical polar fourier correlations. *PROTEINS: Structure Function and Genetics*, 1999.

[7] M. Sanner. Molecular surfaces computation, February 1996.

[8] I. Sović. Vizualizacija makromolekula. Bachelor thesis, Faculty of Electrical Engineering and Computing, June 2008.