BIOINFORMATICS APPLICATIONS NOTE Vol. 29 no. 13 2013, pages 1698–1699
doi:10.1093/bioinformatics/btt262

Structural bioinformatics Advance Access publication May 9, 2013

pyDockWEB: a web server for rigid-body protein–protein docking using electrostatics and desolvation scoring

Brian Jiménez-García[1], Carles Pons[1,2] and Juan Fernández-Recio[1,*]

[1]Joint BSC-IRB Research Programme in Computational Biology, Department of Life Sciences, Barcelona Supercomputing Center and [2]Computational Bioinformatics, National Institute of Bioinformatics (INB), Jordi Girona 29, 08034 Barcelona, Spain

Associate Editor: Anna Tramontano

ABSTRACT

Summary: pyDockWEB is a web server for the rigid-body docking prediction of protein–protein complex structures using a new version of the pyDock scoring algorithm. We use here a new custom parallel FTDock implementation, with adjusted grid size for optimal FFT calculations, and a new version of pyDock, which dramatically speeds up calculations while keeping the same predictive accuracy. Given the 3D coordinates of two interacting proteins, pyDockWEB returns the best docking orientations as scored mainly by electrostatics and desolvation energy.

Availability and implementation: The server does not require registration by the user and is freely accessible for academics at http://life.bsc.es/servlet/pydock

Contact: juanf@bsc.es

Supplementary information: Supplementary data are available at *Bioinformatics* online.

Received on January 23, 2013; revised on April 8, 2013; accepted on May 2, 2013

1 INTRODUCTION

Protein–protein interactions mediate most cellular functions; thus, a detailed description of the association process at molecular level is essential to comprehend the fundamental processes that sustain life. In such line, protein–protein docking tools aim to identify the native binding mode between two proteins (Ritchie, 2008). Such predictions are required to complement experimental techniques that cannot provide structural information at a proteomics scale given their current technical limitations. pyDock (Cheng *et al.*, 2007) is a rigid-body docking method in which sampling is performed by means of FTDock (Gabb *et al.*, 1997) and scoring implements an efficient empirical potential, composed of electrostatics and desolvation terms, with a limited contribution from van der Waals energy. The method has been successfully tested in CAPRI (Grosdidier *et al.*, 2007; Mendez *et al.*, 2003; Pons *et al.*, 2010a). Here, we present pyDockWEB server, a new fast implementation that allows easy access to non-expert users to state-of-the-art docking predictions.

*To whom correspondence should be addressed.

2 pyDockWEB SERVER

pyDockWEB server is a web application for the use of the protein–protein docking and scoring program pyDock. Users can easily send pyDock jobs to be executed in a five-step process via a user-friendly front-end (Fig. 1). In the first step, users have to introduce a project name and a notification email address. In the second step, the scoring algorithm is selected. In the third step, users can either upload their protein coordinate files or indicate the PDB code, in which case, PDB files will be automatically downloaded from RCSB Protein Data Bank. In both cases, PDB files are automatically parsed to select available receptor and ligand chains. An option to automatically set-up a docking job with example PDB files is also available. In the fourth step, users may specify optional distance restraints, which will be computed using pyDockRST (Chelliah *et al.*, 2006) module. Finally, in the fifth step, users will double-check whether data provided are correct and submit a docking job to the server queues. After job submission, user is redirected to a web page where project status is automatically updated and result files can be downloaded after computation is finished. In this web page, the top 10 models scored by pyDock are displayed using Jmol (http://jmol.sourceforge.net/).

pyDockWEB is technically constituted by three different components: a web front-end, *pydockd*, a daemon in charge of managing pyDock executions and a data storage system. The web front-end has been implemented using JSF (Java Server Faces, a Java-based web application framework), Ajax4sf (an open source framework that adds Ajax capabilities to JSF framework) and JSP (Java Server Pages) technologies. Data storage system has been implemented via one of the most popular choices in web applications databases, MySQL (http://www.mysql.com). Data tables have been designed to efficiently store the relevant job information and to gather a few statistics about usage and computation and queued times. The controller, *pydockd*, is an application written in Python version 2.7, which periodically polls job requests created from the web front-end and stored in the MySQL database and submits them as pyDock job instances to the Slurm batch queuing system (https://computing.llnl.gov/linux/slurm/slurm.html).

pyDockWEB uses an optimized pyDock version 3, which also includes a custom parallel version of FTDock, implemented using the MPI (Message Passing Interface) library MPICH2 (Bouteiller *et al.*, 2003) to generate docking poses, which is capable to scale to multiple processors/cores. Another optimization

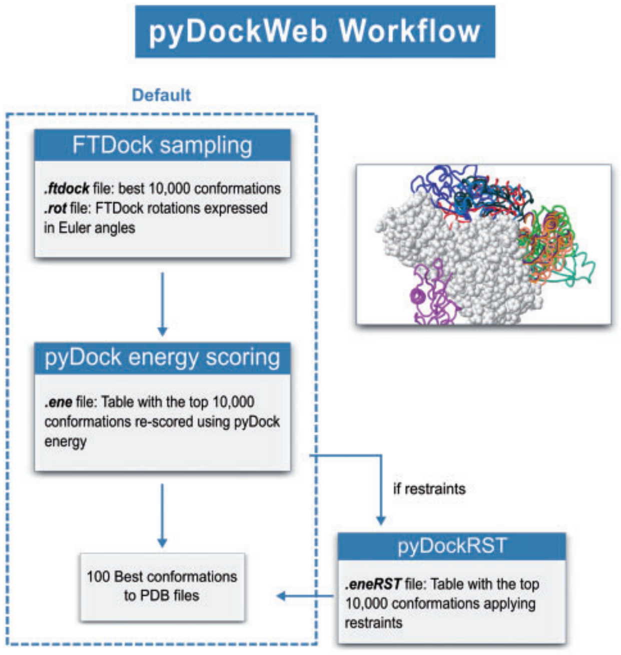
pyDockWeb Workflow
Default
FTDock sampling
.ftdock file: best 10,000 conformations
.rot file: FTDock rotations expressed in Euler angles
pyDock energy scoring
.ene file: Table with the top 10,000 conformations re-scored using pyDock energy
if restraints
pyDockRST
.eneRST file: Table with the top 10,000 conformations applying restraints
100 Best conformations to PDB files

Fig. 1. pyDockWEB workflow

has been implemented, as follows. FTDock makes use of the FFTW 2.1.5 (Frigo and Johnson, 1998) library to perform a global scan of translational and rotational space having the two molecules discretized onto orthogonal grids. The size of the transform in the FFTW scope is proportional to the FTDock grid size in number of cells, which was automatically calculated from the single grid unit size and the size of the proteins. However, according to FFTW's documentation, FFTW algorithms are optimal for sizes that follow Equation (1),

$$n = 2^a \cdot 3^b \cdot 5^c \cdot 7^d \cdot 11^e \cdot 13^f \qquad (1)$$

where $e+f$ is either 0 or 1, and the other exponents are arbitrary. Other sizes are calculated by FFTW using slower algorithms. Therefore, we have adjusted the FTDock grid size, n, to follow Equation (1). This grid size optimization has been implemented in the new custom parallel FTDock version. Supplementary Figure S1 shows the difference of execution times between the original and the grid optimized FTDock versions, as well as the stability in terms of time of the parallel version using the grid size optimization.

The server runs on a multi-user cluster with two nodes. Each node has 16 cores (4 Intel Xeon E5620 Quad Core) at 2.4 GHz. Two cores are reserved for MySQL, JBoss and interactive shells. Physical memory is 65 GB, with 11 TB of total available disk space.

3 BENCHMARKING AND DISCUSSION

The pyDockWEB server provides a user-friendly web front-end to allow the academic community to use the pyDock rigid-body docking and scoring method. The user is notified on completion of the execution and is able to visualize online the top 10 models of the predicted complex using Jmol. We have evaluated the performance of pyDockWEB server on the standard protein–protein docking benchmark 4.0 (Hwang *et al.*, 2010). The quality of the results in terms of generated near-native solutions (ligand RMSD within 10 Å from that in the X-ray complex structure) has not been affected by the optimization and implementation procedure, and the top 10 success rate (i.e. number of cases with near-native solutions within top 10 scored poses) reached 17.0% (Supplementary Table S1), in line with previous benchmarks (Cheng *et al.*, 2007; Pons *et al.*, 2010b). This performance is comparable with other reported servers, as shown on available protein–protein targets from current CAPRI edition (Supplementary Table S2). Interestingly, sampling with FTDock with the new custom parallel and variable grid size implementation achieved speed-ups of up to 181 (50 as average) with respect to the default FTDock distribution, whereas the scoring process based on the new pyDock version 3 achieved speed-ups of up to 40 (38 as average) with respect to the previously available version (Cheng *et al.*, 2007).

Additional pyDock modules and new developments are planned to be implemented in pyDockWEB in the future: patch prediction (pyDockNIP), optimal docking area (pyDockODA), domain–domain assembly (pyDockTET) and SIPPER scoring energy (pyDockSIPPER).

ACKNOWLEDGEMENTS

The authors would like to thank Dmitry Repchevsky for his invaluable help and experience designing and implementing Java Server Faces web applications, and all the different users that have collaborated in testing and giving feedback to improve the server.

Funding: Spanish Ministry of Science and Innovation [BIO2010-22324].

Conflict of Interest: none declared.

REFERENCES

Bouteiller,A. *et al.* (2003) MPICH-V2: a fault tolerant MPI for volatile nodes based on pessimistic sender based message logging. In: *Proceedings of the SC2003 ACM/IEEE conference on Supercomputing*. Phoenix, AZ, USA, p. 25.

Chelliah,V. *et al.* (2006) Efficient restraints for protein-protein docking by comparison of observed amino acid substitution patterns with those predicted from local environment. *J. Mol. Biol.*, **357**, 1669–1682.

Cheng,T.M. *et al.* (2007) pyDock: electrostatics and desolvation for effective scoring of rigid-body protein-protein docking. *Proteins*, **68**, 503–515.

Frigo,M. and Johnson,S.G. (1998) FFTW: an adaptive software architecture for the FFT. *Proc. IEEE Int. Conf. Acoust. Speech. Signal Process*, **3**, 1381–1384.

Gabb,H.A. *et al.* (1997) Modelling protein docking using shape complementarity, electrostatics and biochemical information. *J. Mol. Biol.*, **27**, 106–120.

Hwang,H. *et al.* (2010) Protein-protein docking benchmark version 4.0. *Proteins*, **78**, 3111–3114.

Grosdidier,S. *et al.* (2007) Prediction and scoring of docking poses with pyDock. *Proteins*, **69**, 852–858.

Mendez,R. *et al.* (2003) Assessment of blind predictions of protein-protein interactions: current status of docking methods. *Proteins*, **52**, 51–67.

Pons,C. *et al.* (2010a) Optimization of pyDock for the new CAPRI challenges: docking of homology-based models, domain-domain assembly and protein-RNA binding. *Proteins*, **78**, 3182–3188.

Pons,C. *et al.* (2010b) Present and future challenges and limitations in protein-protein docking. *Proteins*, **78**, 95–108.

Ritchie,D.W. (2008) Recent progress and future directions in protein-protein docking. *Curr. Protein Pept. Sci.*, **9**, 1–15.