FLIPDock

User's Guide

Version 0.4 β

Dr. Yong Zhao and Dr. Michel F. Sanner
10550 North Torrey Pines Road, TPC26
The Scripps Research Institute,
La Jolla, CA 92037-1000, USA
July 2007

Contents

Background and Introduction

What is docking?

Docking is a molecular modeling technique to predict how any macromolecules (typically a protein) interact with other molecules (may be other proteins, nucleic acids or small drug-like molecules). The goal of protein-ligand docking is to predict the position and orientation of a ligand (a small molecule) when it is bound to a protein receptor or enzyme. Pharmaceutical research employs docking techniques for a variety of purposes, most notably in the virtual screening of large databases of available chemicals in order to select likely drug candidates.

Why FLIPDock?

Numerous protein-ligand docking packages are available. One of the major challenges the computational biologists are facing is how to simulate the induced fit effect observed when a protein accommodate various ligands. It can be small rearrangement of sidechains near the active site, small loop conformational changes or even large scale motions. The traditional “lock-and-key” model limits itself from describing such phenomena. Instead, a “hand-and-glove” model should be employed, i.e. both the ligand and the receptor should change conformation during a docking simulation. FLIPDock calculation is such a “hand-and-glove” analogue.

FLIPDock is short for **F**lexible **LI**gand-**P**rotein **Dock**ing. In FLIPDock we include the receptor conformational changes into the docking process, rather than using multiple receptor structures (NMR, X-ray crystal or snapshots from molecular dynamics) or a pre- or post-optimization process. It avoids the problem of choosing the representative conformers for receptor molecules. Instead, the user is in charge of specifying the conformational subspace to explore in the docking process. Details of the FLIPDock protocol can be found in the following sections.

FLIPDock is free for non-profit academic usage. As a component-based design, FLIPDock is open for new variable searching techniques and scoring functions. The source code will be available for interested contributors.

Flexibility Tree data structure

A Flexibility Tree (FT) data structure allows the hierarchical and multi-resolution representation of conformational changes in macromolecules [1]. A FT is built by recursively partitioning a molecule or molecular system into molecular fragments moving relative to each other. Inter-domain motion descriptors (hinge, shear, twist, screw, etc.) and intra-domain motion descriptors (rotameric side chains, normal modes, essential dynamics, etc.) can be assigned to any molecular fragment. A FT encodes relatively small, yet complex subspaces of a protein's conformational space by combining and nesting a variety of motion descriptors. It is essentially an internal coordinate representation. The internal coordinate has been applied to describe the conformational spaces of small organic molecules: instead of using Cartesian coordinates of each atom, the torsion angles are employed to describe the atomic positions. The merit of such description is the number of variables to encode the conformational space can be reduced significantly.

For example, Figure 1 shows a FT for a generic protein. Suppose this protein is composed of two domains that move relative to each other. The initial domain subdivision allows us to assign a "local perturbation" motion descriptor (e.g. the combination of a small translation and a constrained rotation) to domain B. This motion descriptor jitters domain B relative to domain A. Both domains are further subdivided into rigid and moving regions, allowing the assignment of a hinge motion to the moving regions and normal modes perturbations to the rigid regions. Finally, critical side chains in the active site are split out of the rigid regions to allow them to adopt rotameric conformations.

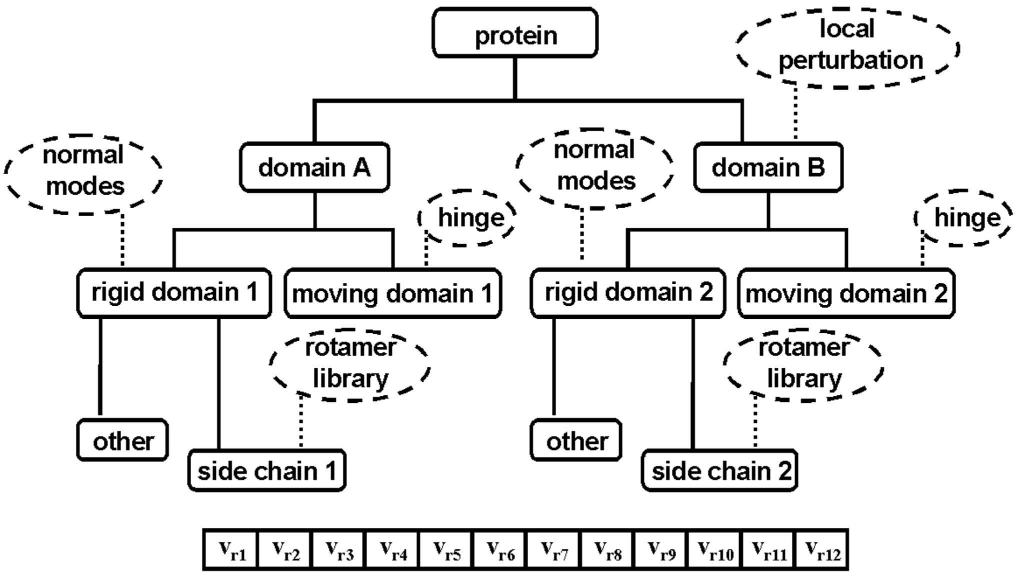


Figure 1. A FT data structure encoding some flexibility in a protein. FT nodes are depicted as boxes and motion descriptors are depicted in dashed ovals. Motion variables are listed at the bottom: 6 variables (vr1 – vr6) for the local perturbation, 1 variable for each: normal mode (vr7, vr8), hinge (vr9, vr10) and rotamer (vr11, vr12) motion objects. The main purpose here is to illustrate the variety of motion descriptors, their hierarchical nesting and their combinations. In docking studies the FT is constructed in a way to emphasize receptor flexibility that is known or assumed to be important for biological activity.

In a recent review on protein-protein docking (*Current Opinion in Structural Biology*, **16**, 194-200.), Bonvin summarized the state-of-the-art approaches in implicit and explicit treatment of flexibility in docking. For large conformational changes involving backbone atoms, the author stated that "Most probably, there will not be a unique solution; rather, it will be the proper combination of approaches for representing conformation changes and flexibility at several levels that will lead to success". This is precisely what the FT enables: combining the motions at various resolutions to encode a specific conformational sub-space. We believe that the ability to mix large conformational changes and local rearrangements will be important for modeling molecular interactions.

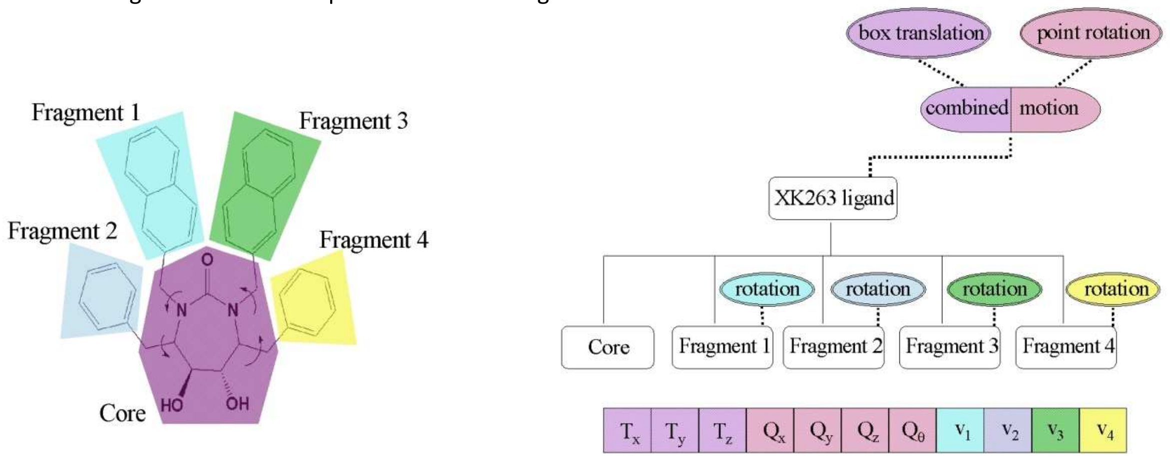


Figure 2. Representing a flexible ligand using a FT structure. If we allow four single bonds to be rotatable, the XK263 ligand is essentially partitioned into five fragments. The four fragments move relative to the fixed core. The conformation of these four fragments is described by four variables: v1-v4. By assigning the combined motion to the FT root node, the ligand can undergo free rotations encoded by the quaternion Qx, Qy, Qz, Qθ, and a constrained translation (Tx, Ty, Tz).

A FT can be used to represent a flexible ligand molecule as well. The FT is defined by identifying rigid moving fragments starting from a fixed core fragment. These rigid fragments are connected to each other by single bonds; hence their relative motion can be represented by an “axis rotation” motion descriptor in which the axis is set to be the rotatable bond. This approach is equivalent to an internal-coordinates parameterization of the ligand conformation, Figure 2.

The architecture of FLIPDock

The three main ingredients of a docking program are: the representation of the ligand-receptor complex, the search method and the method for ranking putative solutions. In FLIPDock we use Flexibility Tree data structures to represent the conformational spaces of both the ligand and the receptor. The solution space (conformations of the molecules, relative position and orientation of the ligand relative to the receptor) is searched using a genetic algorithm (GA) based approach. The putative dockings are evaluated using an atomic pair-wise scoring function based on the AutoDock 3.05 force-field.

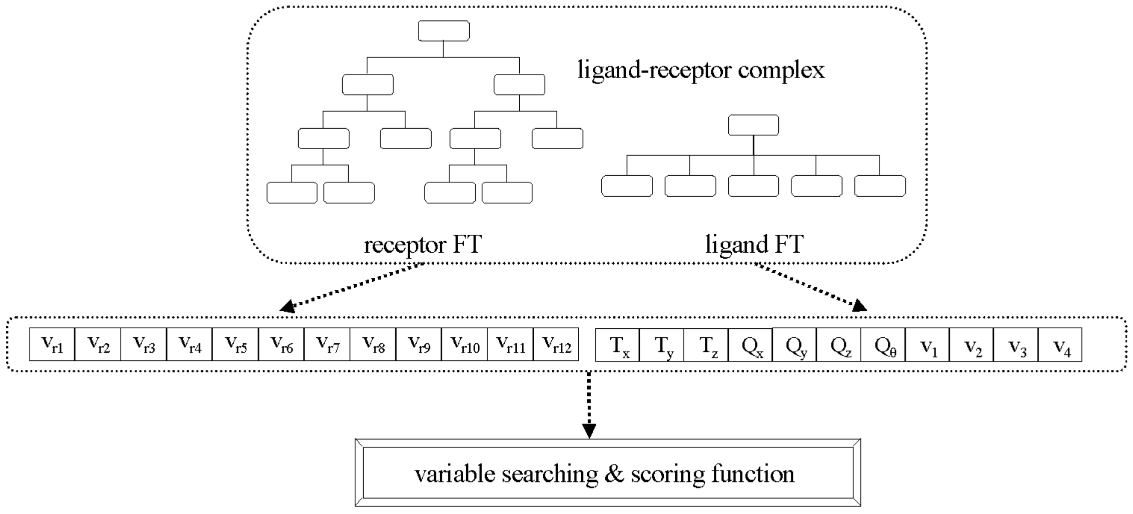

ligand-receptor complex
receptor FT
ligand FT
V_{r1} V_{r2} V_{r3} V_{r4} V_{r5} V_{r6} V_{r7} V_{r8} V_{r9} V_{r10} V_{r11} V_{r12}
T_x T_y T_z Q_x Q_y Q_z Q_θ V_1 V_2 V_3 V_4
variable searching & scoring function

Figure 3. FLIPDock architecture. The problem of docking a flexible ligand into a flexible receptor is encoded using two Flexibility Tree data structures (FTs). Randomizing the motions variables in the FTs provides random ligand-receptor conformations and relative positions and orientations. These putative dockings are evaluated using a scoring function. FLIPDock optimizes the motions variables to minimize the given scoring function.

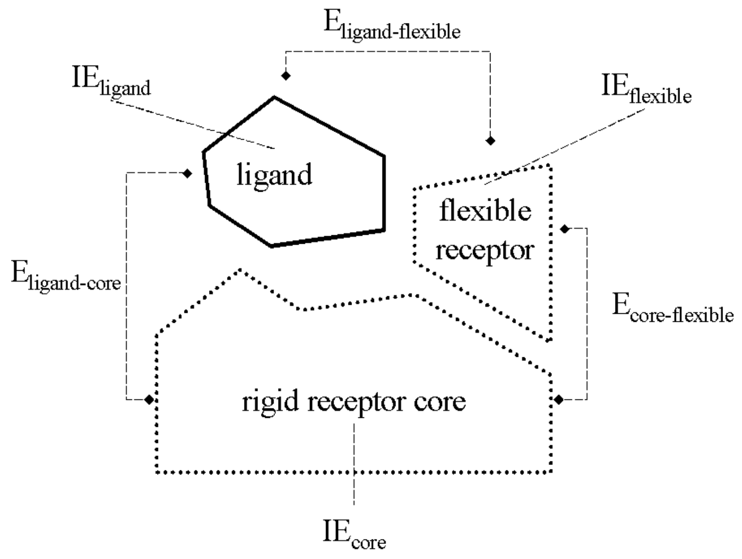
Note that in order to parameterize the problem of docking a flexible ligand into a flexible receptor, we need to encode the position and orientation of the ligand relative to the receptor. This is achieved by assigning a combination of a “box translation” and a “point rotation” to the root node of the ligand’s FT (Figure 2). The “box translation” will be configured to cover the active site of the receptor and will constrain the center of the ligand molecule. This combined rotation and translation of the ligand adds another 7 variables (3 for the translation and 4 for the quaternion representing the rotation).

FLIPDock is designed as a component-based framework. New search techniques and scoring functions can be added to the framework, thus facilitating the interchange and mixing of various search techniques and scoring functions. Currently, FLIPDock provides a simple genetic algorithm (GA) search engine as well as a two step "divide and conquer" genetic algorithm (DAC-GA). During a simple GA search, the center of a ligand is constrained to a 3-dimensional subspace called the docking box. In a DAC-GA search the docking box is partitioned into smaller boxes, typically 3 Å cubes and a short GA optimization is carried out in each one of these small boxes. In a second step, a GA optimization is carried out over the original docking box using a population containing the best individuals (motion variable sets) from the short optimizations carried out on the small boxes. A comparison between the simple GA and DAC-GA has shown that DAC-GA is more efficient than the simple GA when the docking box is large[2].

Scoring functions are also components in the FLIPDock framework. The simplest scoring function currently available in FLIPDock is the RMSD score, which measures how close the putative ligand-receptor complex is from a given reference complex conformation. While this scoring function is not useful for real docking problems (since it requires knowledge of the answer and the energy landscape is too simple) it is helpful for tasks such as debugging. More importantly, the existence of two interchangeable scoring functions allows us to demonstrate and verify the ability to easily interchange scoring functions in the FLIPDock framework.

Currently the scoring function in FLIPDock is based on the AutoDock v3.05 force-field [3]. This empirical force-field takes into consideration four non-bonded interactions: dispersion/repulsion, hydrogen bonding, electrostatics and desolvation upon binding. In AutoDock, grid maps are used to speed up the numerical evaluation of the interaction energy between the ligand and the receptor. Such grid maps cannot be pre-computed in FLIPDock since the receptor changes conformation during the docking. Instead, the atomic pair-wise interactions are evaluated based on the AutoDock force-field. The FLIPDock score is the sum of the internal energy of the ligand, the internal energy of the receptor and the interaction energy between ligand and receptor (equation 1). If only part of the receptor is flexible, the internal energy of the receptor can be obtained by adding the internal energy of the rigid part (a constant), the internal energy of the flexible part, and the interaction energies between the rigid and the moving part of the receptor. The 1-2 (covalent bond) and 1-3 pairwise interactions are not evaluated. The inclusion of 1-4 interactions in the score is optional. Non-bonded interactions between pairs of atoms that are further than a user-specified distance cutoff (default value: 8.0 Å) are not evaluated.

$$\text{score} = \mathrm{E}_{receptor-ligand} + \text{Internal Energy}_{\text{ligand}} + \text{Internal Energy}_{\text{receptor}} \tag{1}$$



$\mathrm{E}_{\text{ligand-flexible}}$
$\mathrm{IE}_{\text{ligand}}$
$\mathrm{IE}_{\text{flexible}}$
ligand
flexible receptor
$\mathrm{E}_{\text{ligand-core}}$
$\mathrm{E}_{\text{core-flexible}}$
rigid receptor core
$\mathrm{IE}_{\text{core}}$

Figure 4. A FLIPDock score is composed of empirical interaction energy (E) and internal energy (IE) terms. A receptor and a ligand are depicted as dashed and solid blocks respectively. When a part of the receptor is rigid the internal energy of the rigid part of the receptor is constant and omitted from the score.

$$\begin{aligned}\text{score} = {} & \mathrm{IE}_{\text{ligand}} + \mathrm{IE}_{\text{flexible receptor}} + \mathrm{IE}_{\text{rigid receptor core}} \\ & + \mathrm{E}_{\text{ligand-core}} + \mathrm{E}_{\text{ligand−flexible}} + \mathrm{E}_{\text{core−flexible}}\end{aligned} \tag{2}$$

Software details

FLIPDock framework is written in Python. The current scoring function is based on AutoDock3.05 force field, which is developed in C++. The SWIG package (version 1.3.21) was used to generate shared objects (binaries).

Python 2.4 is recommended for FLIPDock Version 0.4β. We developed the FLIPDock under Fedora Linux. The non-python code has been compiled for other platforms including Mac OS X, Windows, Sun and SGI. However, FLIPDock has not yet been tested or used exhaustively on these platforms.

Although FLIPDock is provided as command-line programs, we are integrating FLIPDock with our molecular viewer PMV and the visual programming environment Vision to provide a graphical interface for constructing FTs, setting up docking experiments and graphically analyzing the results. (http://mgltools.scripps.edu)

Installation

Obtain FLIPDock

FLIPDock binaries can be obtained from http://flipdock.scripps.edu. Note that a license agreement is required before downloading. Please follow the instructions on "the download" page. Currently the FLIPDock is distributed as Linux binaries. Support for other OS, such as Mac OS, SGI, Windows, will be available upon request.

Source code is also available for those who are interested in contributing to the FLIPDock project.

Install Linux binaries

After downloading the FLIPDock installer to your Linux box, you can either double click the file or run it from a terminal. Note that the ability to execute should be granted to the file. For example:

```
    chmod 755 FLIPDock-0.4-Linux-x86-Install
```

Follow the instruction on the screen to complete installation. Please add the FLIPDock executables to your $PATH. For example, if FLIPDock was installed to /usr/local/flipdock/bin,

bash users:

```
   PATH=$PATH: /usr/local/flipdock/bin
```

For tcsh users:

```
   set PATH ( $PATH : /usr/local/flipdock/bin)
```

Not sure which shell you are using? Try

```
   echo $SHELL
```

After setting up the path, type "flipdock" to make sure FLIPDock executables are ready to use.

In this manuscript, **$FLIPDock** always refers to where FLIPDock was installed.

Install from source code

We deliver FLIPDock source code to interested contributors. Python interpreter is not distributed with FLIPDock. User can download Python2.4 or above from http://www.python.org.

After downloading the FLIPDock source code:

```
   tar xzf flipdockSRC.tgz
   cd FLIPDock
   cd memoryobjectDIST
   python2.4 setup.py install --install-platlib=..
   cd ..
   cd cAutoDockDIST
   python2.4 setup.py install --install-platlib=..
   cd ..
```

The steps above were tested using python2.4 under Fedora Linux. Minor adaptations might be required on other Linux distributions.

Setup a FLIPDock Calculation

Docking parameters that control the calculation can be specified as command line options or saved in a python-style plaint-text parameter file. To see all the acceptable parameters, type

```
    flipdock
```

or

```
    flipdock -help
```

FLIPDock promotes a simple help page:

```
    usage: flipdock [-p parameter_file] [-parameter value] [-help]

    Use flipdock -help [topics] for more information.
    Valid topics are:
    GA repeat help ref searching score receptor file DACGA seed ligand
all
```

Try different topics with the "-help" option. Such as

```
    flipdock –help ligand
```

or

```
    flipdock –help receptor
```

Getting to know all the parameters

User can specify FLIPDock parameters as command-line inputs or as entries in a parameter file (with "-p filename"). Such as "flipdock –p my parameters.txt"

Parameter name	Default Value	Description
P	None	docking parameter file
Help	all	print help
LigandXML	None	XML file name for the ligand
ReceptorXML	None	XML file name for the receptor
Ligand	None	the ligand coordinates in PDBQ format
Receptor	None	the receptor coordinates in PDBQS format
movingSC	None	moving sidechains in the receptor
rand_seed	-1	the seed for random number generator, can be an integer or a float number
box_center	None	center of the docking box, such as [1,2,3]
box_dimensions	None	x,y,z dimensions of the docking box in Å, such as [20,30,25]
scoringFunction	AutoDock3	Scoring function, currently only AutoDock3. More functions will be available in the coming releases.
calcLigIE	True	calculate ligand internal energy? True/False
calcRecIE	True	calculate receptor internal energy? True/False
repeat	1	number of docking tests
rmsdRef	None	After docking, compute RMSD between the predicted ligand pose and these reference structures.
search	GA	Searching method, currently only genetic

		algorithm (GA) and are Divided and Conquer GA are supported. See Proteins, 68, 3 , 726 - 737 Other search engines will be available in the coming releases
GA_gens	100	number of generations
GA_pop_size	100	population size
GA_replace	0.5	replace rate in each generation, between 0 and 1
GA_crossover	0.5	probability of cross-over, between 0 and 1
GA_mutation	0.3	probability of mutation, between 0 and 1
GA_deviation	0.0001	stop evolution when the standard deviation of scores are below this value
GA_enableLocalSearch	False	enable the local search? True/False if enabled (True), it's essentially a Lamarckian GA analog. See: J. Computational Chemistry, 19: 1639-1662
GA_localsearchrate	0.3	local search probability, between 0 and 1
GA_max_eval	1000	maximum number of evaluations allowed.
DAC_x_div	1	divide X of docking box into x pieces
DAC_y_div	1	divide Y of docking box into y pieces
DAC_z_div	1	divide Z of docking box into z pieces
DAC_box_pop_size	100	population size for a DAC GA
DAC_box_gens	10	number of generations for a DAC GA
DAC_box_p_replace	0.5	replace rate in each generation
DAC_box_p_cross	0.5	probability of cross-over
DAC_box_mutation_rate	0.3	probability of mutation
DAC_box_p_deviation	0.0001	stop when deviation of scores is below this value
DAC_box_enableLocalSearch	True	enable the local search in a DAC GA
DAC_box_p_localsearch	0.3	local search probability in a DAC GA
DAC_box_max_evaluation	100000.0	maximum number of evaluations in a DAC GA.

In the following sections, we provide several Exercises as a brief FLIPDock tutorial.

Exercises 1: A test drive of FLIPDock, using parameter file

The FLIPDock needs at least two files as input: 3D structures of a ligand and a receptor. An XML file is used to save the Flexibility Tree data structure, which encodes the conformational space of a ligand (which bonds to be made rotatable) or a receptor (which part of the receptor to move during docking and how to move them). The two XML files can be specified by user or generated on-the-fly with FLIPDock.

Take a moment reading the docking parameter file “setting”, note that all lines begins with “#” are comments and therefore will be ignored by FLIPDock.

```
cd $FLIPDock/testRun
flipdock –p setting
```

The outputs of this docking test will be like the following:

```
##########################################################
#                                                        #
#   _____ _     ___ ____  ____             _             #
#  |  ___| |   |_ _|  _ \|  _ \  ___   ___| | __         #
#  | |_  | |    | || |_) | | | |/ _ \ / __| |/ /         #
#  |  _| | |___ | ||  __/| |_| | (_) | (__|   <          #
#  |_|   |_____|___|_|   |____/ \___/ \___|_|\_\         #
#                                                        #
#  Flexible LIgand Protein Dock (FLIPDock)               #
#     a Python-powered docking program   V 0.40          #
#                                                        #
#     Dr. Yong Zhao    (yongzhao@scripps.edu)            #
#     Dr. Michel Sanner  (sanner@scripps.edu)            #
#                                                        #
#     The Scripps Research Institute (TSRI), TPC26       #
#     La Jolla, CA 92037, USA                            #
#                                                        #
#     http://flipdock.scripps.edu                        #
#     http://dx.doi.org/10.1002/prot.21423               #
#     Copyright: Y.Zhao, M.Sanner and TSRI, 2004-2007    #
#                                                        #
##########################################################
Number of genes: 19
Setting: p: setting
Setting: LigandXML: 1HBV_Lig_example.xml
Setting: box_center: [3.2189999999999999, 4.7679999999999998, -
8.984]
Setting: box_dimensions: [3, 3, 3]
Setting: ReceptorXML: 1HBV_example.xml
Setting: rand_seed: 2007
Setting: scoringFunction: AutoDock3
Setting: calcLigIE: True
Setting: calcRecIE: True
Setting: repeat: 1
Setting: search: GA
Setting: rmsdRef: ['1HBV_Lig.pdbq', '1HBV_Lig_C2.pdbq']
Setting: GA_gens: 20
Setting: GA_pop_size: 150
Setting: GA_replace: 0.5
Setting: GA_crossover: 0.5
Setting: GA_mutation: 0.3
Setting: GA_deviation: 0.0001
Setting: GA_enableLocalSearch: False
```

```
    Setting: GA_localsearchrate: 0.3
    Setting: GA_max_eval: 10000

    Docking test # 1
    Docking begins at: Fri Jul 27 13:56:31 2007

    Using  2007  as random seed
    gen: 0 min: 41.1386086744 dev: 4.2067e+00 eval time:   11.07 sec
    gen: 1 min: 31.5243446349 dev: 4.2593e-01 eval time:    5.79 sec
    gen: 2 min: 22.2112841031 dev: 5.1620e-01 eval time:    5.84 sec
    gen: 3 min: 22.2112841031 dev: 5.9835e-01 eval time:    5.88 sec
    gen: 4 min: 14.2461425473 dev: 7.8190e-01 eval time:    7.24 sec
    gen: 5 min: 14.2461425473 dev: 1.1291e+00 eval time:    9.04 sec
    gen: 6 min: -4.30641124907 dev: 2.0720e+00 eval time:   10.90 sec
    gen: 7 min: -4.30641124907 dev: 9.0783e-01 eval time:   11.75 sec
    gen: 8 min: -4.30641124907 dev: 7.0403e-01 eval time:   12.41 sec
    gen: 9 min: -4.30641124907 dev: 5.3975e-01 eval time:   12.15 sec
    gen: 10 min: -15.2060913731 dev: 5.7632e-01 eval time:   12.20 sec
    gen: 11 min: -17.9376750188 dev: 7.0376e-01 eval time:   12.43 sec
    gen: 12 min: -17.9376750188 dev: 7.9206e-01 eval time:   12.84 sec
    gen: 13 min: -17.9376750188 dev: 8.9839e-01 eval time:   12.19 sec
    gen: 14 min: -17.9376750188 dev: 1.2407e+00 eval time:   13.02 sec
    gen: 15 min: -17.9376750188 dev: 2.1982e+00 eval time:   13.02 sec
    gen: 16 min: -18.12697331 dev: 1.0179e+01 eval time:   13.26 sec
    gen: 17 min: -18.12697331 dev: 3.2435e+00 eval time:   12.31 sec
    gen: 18 min: -18.12697331 dev: 1.2692e+00 eval time:   12.93 sec
    gen: 19 min: -18.5320185589 dev: 7.6731e-01 eval time:   12.88 sec
    gen: 20 min: -18.5320185589 dev: 5.0293e-01 eval time:   13.12 sec

    INFO: best_gene=[0.119481616156, 0.146735570024, 0.818155124069,
0.217404571666, 0.842091128654, 0.54374651253, 0.954628769084,
0.950217512888, 0.196088063991, 1.0, 0.0203395077858, 0.987055459068,
0.302245156332, 0.430909113573, 0.348121838962, 0.0, 0.0235069395284,
0.320164502688, 1.0]

    INFO: best_score=-18.5320185589
     ***  evolution finished at  Fri Jul 27 14:00:25 2007   ***
     ***  docking takes       3.90 minutes

    INFO: number_evaluation=1650
    -----------------

    1HBV_Lig.pdbq  ---> rmsd= 1.02013680715
    1HBV_Lig.pdbq  ---> rmsd= 1.02013680715
    INFO: best_RMSD = 1.02013680715

    ***********************  Summary  ***********************
    **  interaction energy between ligand and receptor: -13.543648
       -     hydrogen bonding:       -0.679487
       -          desolvation:       -1.101025
       -        Van der Waals:       -12.312161
       -       electrostatics:       0.549025

    **  internal energy of ligand: -3.651270

    **  internal energy of receptor: -1.337101
```

```
101         -   btw flex and rigid:       -2.897102
102         - i.e. flexible domain:       1.560001
103         - i.e. of rigid domain:       0.000000
104
105      **  total score: -18.532019
```

Line **1-23**:	logo, copy right, etc.
Line **24**:	number of variables to optimize
Line **25-46**:	FLIPDock parameters.
Line **48-51**:	Helpful information on starting time, random seed, etc.
Line **52-72**:	Evolution of Genetic Algorithm. "**gen**" refers to the generation number; "**min**" is the best (minimal) score achieved in this generation; "**dev**" is the standard deviation of scores in this generation. "**eval time**" is the time consumed by this generation.
Line **74-78**:	the "best gene", namely the optimized variables.
Line **80**:	best score achieved.
Line **81-82**:	information about simulation time.
Line **84**:	number of energy evaluations.
Line **87-89**:	RMSD values between the optimized ligand position and the reference position.
Line **91-105**:	The "break-downs" of each term in the FLIPDock score

Note: All the lines that begin with "Setting:" specify the docking parameters.
All the lines that begin with "gen:" record the GA based optimization.
All the lines that begin with "INFO:" specify the docking results.

Exercises 2: Using command line to specify docking parameters

In this exercise, we carry out the same docking as in Exercise 1.

```
flipdock -LigandXML 1HBV_Lig_example.xml -ReceptorXML
1HBV_example.xml -GA_pop_size 150 -GA_gens 20 -rand_seed 2007 -
GA_max_eval 10000
```

Here we specify the XML file encoding ligand conformational space (1HBV_Lig_example.xml), the XML file for the receptor (1HBV_example.xml), change the population size (150) and the number of generations (20) in GA evolution, use 2007 as the seed to generate random numbers and define the maximum allowed energy evaluations (10000).

Exercises 3: Command-line values overwrite the values in the parameter file

If both docking parameter files and command-line variables are specified, the ones in the command-line will overwrite the counterparts in the parameter file.

```
flipdock -p setting -GA_gens 20 -GA_pop_size 10 -rand_seed -1 -
repeat 3
```

Here we use all the parameters in the file "setting" and change the values of "GA_gens", "GA_pop_size", "rand_seed" and "repeat".

Exercises 4: Ligand provided in PDBQ or PDB format

PDBQ format was used in AutoDock3 suite to define the ligand. A set of rotatable bonds are saved in the "TorTree" data structure (More information about TorTree and how to select rotatable bonds in a ligand can be found at http://autodock.scripps.edu and in the ADT package at http://mgltools.scripp.edu)

If the user provides the ligand as a PDBQ file, FLIPDock will convert the TorTree data structure to Flexibility Tree and save it in an XML file.

```
flipdock -Ligand 1HBV_Lig.pdbq -ReceptorXML 1HBV_example.xml -
box_dimensions 3 4 5 -box_center 2 5 6 -calcRecIE False -GA_gens
10 -GA_pop_size 10
```

Note that the docking box information (such as center, dimensions in X,Y and Z) should also be provided so FLIPDock would know where to dock the ligand.

The conformational space of the ligand will be saved as an XML file. In this example, a file called “1HBV_Lig.xml” will be generated in the same directory. If a user wants to repeat the docking calculation, this XML can be reused by specifying the “-LigandXML” parameter.

When the ligand coordinates is available in PDB or MOL2 format, a user can use the “**pdb2pdbq**” utility to convert a ligand into the PDBQ format.

```
pdb2pdbq ligand.pdb [–o outpoutName.pdbq]
```

The default output filename is ligand.pdbq. Other output name can be specified with “-o” flag. In a pdbq file, polar hydrogen atoms and charges will be assigned. A torsion tree data structure will be saved in pdbq file as well. A set of single bonds will be marked as rotatable in this file. To visualize and customize rotatable bonds in the ligand, use AutoDockTools (ADT) to visualize and choose the torsions via graphical user interface. More information about ADT can be found at http://mgltools.scripps.edu

NOTE: The polar hydrogen and charge assignment is based on “PyBabel”, part of MGLTools distribution. Alternatively the user has the option to use OpenBabel for adding hydrogen atoms, please refer to the FAQ section.

Exercises 5: Rigid receptor docking

In FLIPDock, the receptor can be rigid or flexible. The simplest way to specify a rigid receptor is to use the “-Receptor” parameter and

```
flipdock -LigandXML 1HBV_Lig_example.xml -Receptor 1HBV.pdbqs -
GA_pop_size 15 -GA_gens 20 -calcRecIE False
```

Note that if a rigid receptor is specified, the “-calcRecIE” parameter should be set to “False”. i.e., no internal energy of the receptor will be computed since it’s a constant.

Exercises 6: Flexible receptor docking

In FLIPDock, the conformational changes in a receptor can be sidechain motions, small loop rearrangements, or even large scale domain motions. Since the conformational space encoded by Flexibility Tree data structure is always saved in an XML file, the best way to specify receptor flexibility is to construct such a file and use it as the “-ReceptorXML” parameter.

FLIPDock can describe the induced conformational changes in receptor, including domain motions (e.g. hinge), backbone vibrations (e.g. normal modes, essential dynamics) and/or side-chains rearrangements. Due to the variety of the motion descriptors, it’s demanding to provide a simple and uniform interface for each of the motions types as command-line parameters. Using the “-ReceptorXML” parameter is highly recommended when multiple motions descriptors are combined. However, when only sidechain motions are involved, the interface can be much simpler:

```
flipdock -LigandXML 1HBV_Lig_example.xml -Receptor 1HBV.pdbqs -
GA_pop_size 15 -GA_gens 20 -movingSC 1HBV:A:ARG8,ASP29,ASP30
```

In the docking test above, we use the "-movingSC" to specify three amino acid sidechains in chain A of 1HBV.pdbqs file. PDBQS format was used in AutoDock3 for the receptor. In a PDBQS file, polar hydrogen atoms, charges and desolvation parameters will be assigned. Alternatively, the PDBQS file can be generated with AutoDockTools (ADT) via the graphical user interface. More information about PDBQS format can be found at http://autodock.scripps.edu and in the ADT package from http://mgltools.scripps.edu.

In FLIPDock package, we provide a utility tool "**pdb2pdbqs**" to convert receptor in PDB format into the PDBQS format. To use it, try

```
pdb2pdbqs 1HBV.pdb [-o receptor.pdbqs]
```

The default output filename is receptor.pdbqs. Other output name can be specified with "-o" flag.

Note that the conformational space of the receptor will be saved as an XML file: 1HBV.xml. It can be reused for other FLIPDock calculations, using the "-ReceptorXML" flag.

In many cases multiple chains exist in a receptor structure. To specify flexible sidechains from different chains we use the same syntax as in MGLTools (http://mgltools.scripps.edu). For example,

```
1HBV:A:ARG8,ASP29;1HBV:B:ASP30,VAL82
```

refers to Arg8 and Asp29 from chain A of the molecule 1HBV and Asp30 and Val 82 from chain B of the same molecule. Note that

1. Residue names are in upper case
2. Different residues from different chains are separated by semi-column (;)
3. The residues in the same chain are separated by comma (,).

Cautious should be taken when using the "-movingSC" flag to specify flexible sidechains as command-line parameters. The semi-column is usually used by UNIX shell for other purposes. Under such a circumstance, we should use the double quote ("").

This is not a problem when "-movingSC" is used in a parameter file.

```
flipdock -LigandXML 1HBV_Lig_example.xml -Receptor 1HBV.pdbqs -
GA_pop_size 15 -GA_gens 20 -movingSC
"1HBV:A:ARG8,ASP29,ASP30,VAL82;1HBV:B:ARG8,ASP29,ASP30,VAL82"
```

Same as previous example, an XML file "1HBV.xml" will be generated.

POTENTIAL PROBLEM: If the chain's name was left blank, which is not uncommon in crystal structures, the blank, or the space character, will confuse FLIPDock.

```
1HBV: :ARG8,ASP29,ASP30
```

An easy fix will be manually changing the chain identifier (column 22 if in PDB format) to be "A".

Exercises 7: Direct FLIPDock output into a log file

In the following example, we use system time as seed (-1) for random number generator, repeat 3 short docking tests, and direct all FLIPDock standard outputs to a log file ('logFile').

```
flipdock -p setting -GA_gens 20 -GA_pop_size 10 -rand_seed -1 -
repeat 3 > logFile &
```

Exercises 8: Extract FLIPDock predictions from a log file

The final docking poses of the ligand and the receptor are not saved in the log file. Instead, the optimized variables, namely the best "gene" of GA evolution, are saved. The utility "**saveDockingResult**" will regenerate the FLIPDock prediction into two files, corresponding to the ligand and the receptor, respectively.

A reference ligand pose can be specified in the docking configuration file. At the end of FLIPDock simulation, the all atom RMSD between the predicted ligand pose and the reference conformation will be reported in the log file. This is helpful when the ligand poses have been observed in experiments.

The best score ever achieved by GA optimization will be reported as well, with break-downs of energetic contributions from various terms. Note that a FLIPDock score is the sum of several energetic terms calculated with AutoDock3.05 force field. Such a score does not correspond to the binding affinity observed in experiments.

```
saveFLIPDockResult logFile -op myTest
```

The "-op" flag specifies the prefix for output files. In this example, the log file contains the results of 3 docking tests (see Exercises 7). Six files will be generated, two for each prediction.

Further analyzing the docking predictions

We are currently working a graphical user's interface and visual analysis tools. It will be available as part of the future release. At this stage, the user is encouraged to use his favorite 3D visualization application to investigate the interactions between the ligand and receptor.

Summary

Run FLIPDock is as simple as typing one line at the shell:

```
flipdock mysetting > logfile &
```

The file "mysetting" is the docking parameter file, while the file "logfile" is used to save all the FLIPDock outputs. The final docking prediction can be derived from the log file.

Valid parameters can be found in page 7-8 or by typing

```
flipdock -help
```

The most demanding part of setting up a FLIPDock calculation is the flexibility definition of the receptor. When only sidechains are considered, please refer to Exercises 6. When more complicated motions are involved, please modify the XML sample files under $FLIPDOCK/examples.

We strongly recommend running an integrity check with "**readxml**" if you generated the XML file. For example, the following line will find errors in the XML file:

```
readxml  ligand.xml
```

To verify that there is no ambiguity of atoms and motions, use the "-v" flag:

```
readxml  receptor.xml -v
```

Frequently Asked Questions

- **How to run it on bluefish cluster (at The Scripps Research Institute)?**

Several configurations are needed. For Csh users, please add the following lines to your .cshrc file. Users of other shells can adjust it accordingly.

```
setenv PYTHONPATH /bluefish/people-a/sanner/MGLpackages:/bluefish/people-
a/sanner/MGLpackages/Numeric;

set path = ($path /bluefish/people-a/yongzhao/software/bin)
```

- **Is FLIPDock very slow, since it does not use any grid maps?**

Using an empirical pair-wise scoring function by nature is slower than the grid map approaches. In the "test drive" example, we witnessed 10-20 evaluations per second on single 3.0GHZ CPU (Linux RedHat 9). It also important to bury in mind that the time FLIPDock consumes also depends on the setting of genetic algorithms, especially the number of evaluations.

- **Should I change the default genetic algorithm parameters?**

Yes. We highly recommend users to adjust GA parameters depending on the problem under study. The FLIPDock performance depends on the setting of GA. The default values of mutation rates, population size and crossover rate were based on parameter tuning tests of a few docking tests. It may not be applied to any docking study. We believe there is no simple rule on the choice of GA parameters such as population size, mutation rate, etc. because of random number factor.

Some docking problems may be more challenging for GA than others. For example, the total number of variables can serve as a coarse measurement of how hard the docking problem is. Using a small population for large searching problem will be questionable. However there is no convincing evidence on the relationship between the docking accuracy and the choice of GA parameters. Same argument can be applied to other parameters such as mutation rate, cross-over rate, etc. In our docking studies, we usually make the population size to be 10 times of the number of variables.

Based on our experience, making a single long GA evolution is not as efficient as running many shorter GA runs.

We encourage users to explore the possible empirical rules and share them with other users.

- **How large is the rotamer library?**

The rotamer library is derived from the backbone-independent library by Dr. Dunbrack's team (http://dunbrack.fccc.edu/). The number of conformers highly depends on the amino acid. For example, there are 78 conformers for Lys, 81 for Arg, but only 6 for Phe.

- **How to restrict part of my ligand?**

In many cases, a user wants to dock a ligand with certain conditions. Take ligand optimization as an example, a covalent bond or hydrogen bond might be known to be crucial for ligand bonding. The goal of docking is to keep the hydrogen bond or covalent bond and investigate the bonding modes of other ligand derivatives. To do that,

1) Use ADT to choose the fixed end of the ligand as the ligand "root"
2) Save the ligand as PDBQ file (AutoDock3), e.g. ligand.pdbq
3) Generate docking parameters with "Ligand" parameter to be ligand.pdbq
4) Run the docking and terminate it after the ligand.xml is generated.
5) Edit the ligand.xml file, locate the following two motions: "FTMotion_BoxTranslate" and "FTMotions_RotationAboutPoint". Modify the "can_be_modified" values from 1 to 0.
6) Re-ran the FLIPDock with "–LigandXML=ligand.xml"

This will turn off the ligand translation and rotation in a 3D box. The interaction between the ligand and the receptor will be kept. Only internal torsions specified in the PDBQ file will be explored in the docking process.

- **How to specify flexibility other than sidechains in a receptor?**

  Any supported flexibility will eventually be written to a plain-text XML file. Several motion descriptors are available for backbone flexibility, such as hinge, normal mode, essential dynamics, etc. Due to the diversity of various descriptors, no simple interface is currently available for the motion combinations other than sidechain flexibility. Currently the only solution is to edit the XML file directly.

  We are working on the graphical user interface of FLIPDock.

- **How to add hydrogen atoms with OpenBabel via Python interface?**
  1) Install OpenBabel if you have not done so. go to http://openbabel.sourceforge.net, follow the instructions on http://openbabel.sourceforge.net/wiki/Install_%28source_code%29
  2) Install the OpenBabel - Python interface
     ```
     Python2.4 setup.py install
     ```
  3) Here is an example python script to add hydrogen atoms.

  ```
  import openbabel
  mol = openbabel.OBMol()
  obConversion = openbabel.OBConversion()
  obConversion.SetInAndOutFormats("pdb", "pdb")  ## input/output  formats
  obConversion.ReadFile(mol, "1FMO.pdb")
  mol.AddHydrogens()
  obConversion.WriteFile(mol, "1FMO_Hs.pdb")
  ```

  NOTE:
  - The added Hydrogen atoms are appended to the end of the file
  - Hydrogen atoms are named H, instead of HA, HB2.. etc.

- **How to generate a movie like the ones in FLIPDock home page?**

  Several movies in http://flipdock.scripps.edu/movies replay the docking process. We captured the screen shots via the PMV-Vision interface. Please contact with us if you are interested in this.

- **How do I report bugs, get support?**

  Any bugs should be sent to Dr. Yong Zhao, yongzhao@scirpps.edu, or post it to the forum at http://flipdock.scripps.edu/. A mailing list is available for users to exchange information. Email to flipdock@scripps.edu or visit http://mgldev.scripps.edu/mailman/listinfo/flipdock

References

[1] Y. Zhao, D. Stoffler and M. Sanner, Bioinformatics, 22 (2006) 2768.
[2] Y. Zhao and M.F. Sanner, Proteins: Structure, Function, and Bioinformatics, 68 (2007) 726.
[3] G. Morris, D. Goodsell, R. Halliday, R. Huey, W. Hart, R. Belew and A. Olson, Journal of Computational Chemistry, 19 (1999) 1639.