Chapter 10

Rosetta Ligand Docking with Flexible XML Protocols

Gordon Lemmon and Jens Meiler

Abstract

RosettaLigand is premiere software for predicting how a protein and a small molecule interact. Benchmark studies demonstrate that 70% of the top scoring RosettaLigand predicted interfaces are within 2 Å RMSD from the crystal structure [1]. The latest release of Rosetta ligand software includes many new features, such as (1) docking of multiple ligands simultaneously, (2) representing ligands as fragments for greater flexibility, (3) redesign of the interface during docking, and (4) an XML script based interface that gives the user full control of the ligand docking protocol.

Key words: Rosetta, RosettaLigand, Ligand, Docking, Small molecule, Flexible, Flexibility, Interface

1. Introduction

Rosetta is a suite of applications used in protein modeling (2). These applications have proven themselves in the areas of protein structure prediction (3), protein-protein docking (4), protein design (5), and protein-ligand docking (1). In 2006 RosettaLigand was introduced as premier software for modeling protein/small molecule interactions. RosettaLigand samples the rigid body position and orientation of the ligand as well as side-chain conformations using Monte Carlo minimization. Ensembles of ligand conformations and protein backbones were used to sample conformational flexibility. The models produced by RosettaLigand conformational sampling are evaluated with a scoring function that includes an electrostatics model, an explicit orientation-dependent hydrogen bonding potential, an implicit solvation model, and van der Walls interactions (1). Default ligand-centric score term weights are provided through "ligand.wts" and "ligand_soft_rep.wts" (see the SCOREFXNS section of

Riccardo Baron (ed.), *Computational Drug Discovery and Design*, Methods in Molecular Biology, vol. 819, DOI 10.1007/978-1-61779-465-0_10, © Springer Science+Business Media, LLC 2012

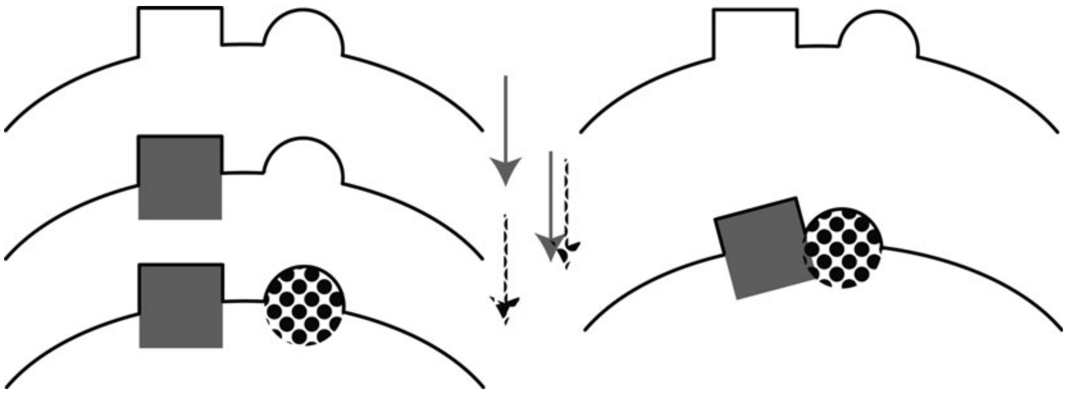

Fig. 1. Multiple ligand docking. *Black* curve represents a protein interface. *Square* and *circle* represent two ligands. Often multiple ligands, cofactors, water molecules, and ions interact with a protein in a synergistic manner to produce the resultant interface structure. Using ligand docking software to dock each of these components separately (*left*) may fail to capture protein induced-fit effects. Simultaneous docking of multiple ligands (*right*) with backbone and side-chain flexibility improves modeling of interfaces—especially those with induced-fit effects.

Fig. 2). However we have found that optimizing these score term weights for a particular class of protein/ligand complexes can greatly improve predictions (see Note 1).

RosettaLigand was later enhanced to allow receptor backbone flexibility as well as greater ligand flexibility (6). Both ligand flexibility and backbone flexibility were shown to improve self-docking and cross-docking scores and lead to better performance than the open-source competitor AutoDock. Ligand flexibility was modeled by sampling ligand conformers and minimizing ligand torsion angles. Backbone flexibility included selecting stretches of residues near the ligand and sampling phi/psi angles for those residues, using a gradient based minimization (6). Libraries of ligand conformers can be generated using methods presented by Kaufmann et al. (7). These features have enabled Rosetta to excel in predicting how pharmaceutically relevant compounds interact with their target (8).

In this chapter we present new features and enhancements to RosettaLigand. Multiple ligands, cofactors, ions, and key water molecules can now be docked simultaneously (Fig. 1). User provided ligand conformations are now sampled during docking, along with protein side-chain rotamer sampling. Interface residue identities can now be redesigned during docking. A new XML script format is used to describe the ligand docking protocol (Fig. 2). This adds great flexibility for the user to customize their docking study.

```
This protocol will simply do low-resolution followed by high-resolution docking.
It will also report the binding energy (ddg) and buried-surface area (sasa) in the score file.
<ROSETTASCRIPTS>
        <SCOREFXNS>
                <ligand_soft_rep weights=ligand_soft_rep>
                        <Reweight scoretype=hack_elec weight=0.42/>
                </ligand_soft_rep>
                <hard_rep weights=ligand>
                        <Reweight scoretype=hack_elec weight=0.42/>
                </hard_rep>
        </SCOREFXNS>
        <LIGAND_AREAS>
                <docking_sidechain chain=X cutoff=6.0 add_nbr_radius=true all_atom_mode=true minimize_ligand=10/>
                <final_sidechain chain=X cutoff=6.0 add_nbr_radius=true all_atom_mode=true/>
                <final_backbone chain=X cutoff=7.0 add_nbr_radius=false all_atom_mode=true Calpha_restraints=0.3/>
        </LIGAND_AREAS>
        <INTERFACE_BUILDERS>
                <side_chain_for_docking ligand_areas=docking_sidechain/>
                <side_chain_for_final ligand_areas=final_sidechain/>
                <backbone ligand_areas=final_backbone extension_window=3/>
        </INTERFACE_BUILDERS>
        <MOVEMAP_BUILDERS>
                <docking sc_interface=side_chain_for_docking minimize_water=true/>
                <final sc_interface=side_chain_for_final bb_interface=backbone minimize_water=true/>
        </MOVEMAP_BUILDERS>
        <MOVERS>
        single movers
                <StartFrom name=start_from chain=X>
                        <Coordinates x=-1.731 y=32.589 z=-5.039/>
                </StartFrom>
                <Translate name=translate chain=X distribution=uniform angstroms=0.01 cycles=50/>
                <Rotate name=rotate chain=X distribution=uniform degrees=360 cycles=1000/>
                <SlideTogether name=slide_together chain=X/>
                <HighResDocker name=high_res_docker cycles=6 repack_every_Nth=3 scorefxn=ligand_soft_rep movemap_builder=docking/>
                <FinalMinimizer name=final scorefxn=hard_rep movemap_builder=final/>
                <InterfaceScoreCalculator name=add_scores chains=X scorefxn=hard_rep native="inputs/7cpa_7cpa_native.pdb"/>
        compound movers
                <ParsedProtocol name=low_res_dock>
                        <Add mover_name=start_from/>
                        <Add mover_name=translate/>
                        <Add mover_name=rotate/>
                        <Add mover_name=slide_together/>
                </ParsedProtocol>
                <ParsedProtocol name=high_res_dock>
                        <Add mover_name=high_res_docker/>
                        <Add mover_name=final/>
                </ParsedProtocol>
        </MOVERS>
        <PROTOCOLS>
                <Add mover_name=low_res_dock/>
         <Add mover_name=high_res_dock/>
                <Add mover_name=add_scores/>
        </PROTOCOLS>
</ROSETTASCRIPTS>
```

Fig. 2. Ligand docking using rosetta_scripts compatible XML. This protocol will do low-resolution docking followed by high-resolution docking. “Compound movers” group simple movers for clarity. The parameters in this protocol replicate those used by Davis et al. (6).

2. Materials

RosettaLigand is part of the Rosetta software suite for protein structure prediction. Visit http://www.rosettacommons.org/ to obtain a license, download the latest release, and read the manual for help installing the software. The information in this tutorial applies to Rosetta version 3.2. Read the documentation about how to run Rosetta executables using command line or flag file options (http://www.rosettacommons.org/manuals/archive/rosetta3.1_user_guide/command_options.html). Read the tutorial entitled "Dock Design Parser Application" (http://www.rosettacommons.org/manuals/archive/rosetta3.1_user_guide/app_dock_design.html). This guide describes an XML format that is now used for all aspects of ligand docking.

2.1. Preparation of Protein PDB Input File

Assure that the protein PDB has at least one backbone heavy atom present for each residue. Rosetta can add missing atoms to incomplete residues. If a residue is completely missing use loop building to add its coordinates. Follow the loop building tutorial (http://www.rosettacommons.org/manuals/archive/rosetta3.1_user_guide/app_loop.html). Assure that residues are numbered in sequence. Rosetta will renumber residues if they are not. Assure that each ligand, cofactor, water molecule, or ion you wish to dock is assigned its own chain ID.

RosettaLigand has been successful in comparative modeling (9), where an experimental structure of the protein of interest is not available. In this case, a sequence alignment is made between the protein of interest and a homologous protein with similar sequence. The three-letter codes in the PDB file of the homologous protein are replaced with the three-letter codes of the protein of interest, according to the sequence alignment and side chain conformations are reconstructed using a rotamer library. If the protein of interest has insertions, loop modeling is used to fill in missing density.

Since ligand docking only repacks side-chain residues within the interface, we first repack all side-chain residues in the protein using the same score function that will be used in ligand docking. By optimizing unbound and bound protein structures using the same scoring function, we ensure that predicted binding affinity is based strictly on changes related to ligand docking. The following XML code can be used for repacking the unbound structure within rosetta_scripts.

```
<SCOREFUNCTION>

    <hard_rep weights=ligand>

</SCOREFUNCTION>

<MOVERS>

    <Repack name=repack score_function= hard_rep>

<MOVERS>
```

2.2. Preparation of Ligand PDB and "Params" Input Files

If you are starting with a ligand in PDB format, first convert it to .mol or .mol2 format. Use <rosetta_source>/src/python/apps/mol_to_params.py to generate a ligand params file and a ligand PDB file with Rosetta atom types. The .params file describes partial charges, atom types, bond lengths, bond angles, torsion angles, and atom types for each residue. Append the atoms in the generated ligand pdb file onto the end of the prepared protein PDB file.

If you are interested in large-scale ligand flexibility, generate conformations for your ligand using OpenEye's Omega (http://www.eyesopen.com/omega) or MOE (http://www.chemcomp.com). These conformations should be in one PDB format separated by TER statements. Add the line "PDB_ROTAMERS <location of PDB file with ligand conformations>" to the end of your .params file.

If your ligand has more than 7 rotatable bonds or if over 100 conformations are required to fully cover the conformational space of your ligand, split it into several smaller fragments. Specify split points at the bottom of your .mol or .mol2 file before running molfile_to_params.py in this fashion: "M SPLT <index 1> <index 2>" where indices 1 and 2 correspond to the atom number in the .mol or .mol2 file (the ATOM block line number). molfile_to_params.py will generate a .params file for each fragment.

2.3. Relevant Command Line or Flags File Options

Rosetta applications use a common set of options that can be specified either at the command line or in a file. Not all Rosetta options are relevant or accessed by each Rosetta application. The options below are most commonly used with ligand docking. An asterix signifies a required option.

1. –in:path:database <path to Rosetta database>. The Rosetta database directory is downloaded from www.rosettacommons.org and contains chemical descriptions of each amino acid as well default score term weights.
2. –in:file:s <space delimited list of PDB files containing protein and ligand(s)>. Alternatively use –in:file:list.
3. –in:file:list <text file with two or more PDB files listed on each line>. This option is especially useful for processing batches of proteins and ligands. PDBs on the same line are concatenated for docking.
4. –in:file:extra_res_fa <space delimited list of .params files for each ligand>. See Subheading 2.2 for preparation of these .params files. Alternatively use -in:file:extra_res_path.
5. –in:file:extra_res_path <path to find .params files>. All files in this directory that end with ".param" or ".params" will be included in docking.

6. –out:nstruct <number of models to produce per input PDB>. See Note 2 on determining how many models to produce.
7. –out:file:atom_tree_diff <name of output file>. In atom_tree_output files only differences from a reference structure are recorded. Since output models usually only differ within the interface region, much less disk space is used by only recording differences.
8. –parser:protocol <name of rosetta_scripts XML file>. This file allows the user to customize each step of ligand docking.
9. –packing:ex1, packing:ex2. These options provide larger (more fine-grained) rotomer libraries for conformational sampling of amino acid side chains. This can improve results but also increases compute times.

3. Methods

The RosettaLigand protocol has been implemented as an XML script used with rosetta_scripts. Instead of providing a separate RosettaLigand executable, the user creates an XML script that describes each of the pieces of ligand docking, and passes this script to the rosetta_scripts executable. This provides a large degree of flexibility to the user, and allows him or her to create novel approaches to ligand docking. In this section XML scriptable components directly related to ligand docking are described. Figure 1 combines these components into a complete ligand docking protocol that replicates the previously published protocol. Hundreds of additional components that are not ligand-centric are available and described in the rosetta_scripts documentation found in the user guide. The XML components below are presented in the order in which they would be used during ligand docking.

3.1. StartFrom

Provide a list of possible xyz starting `Coordinates` for your ligand. One of these points is chosen at random and the ligand specified by the `chain` parameter is recentered at this position.

```
<StartFrom name=(string) chain=(string)/>
    <Coordinates x=(float) y=(float) z=(float)/>
</StartFrom>
```

3.2. Translate

Randomly move the ligand up to a specified distance in any direction from its starting position. If you are confident about your ligand's starting position and seek only to fine tune this position, consider selecting from a `gaussian` distribution, where the specified `angstroms` represent one standard deviation from the starting point. If the random translation lands the ligand on top of another protein (as evaluated by the repulsive score term), then try another random translation. Repeat this `cycles` number of times before giving up and leaving the ligand at the starting point.

```
 <Translate name=(string) chain=(string)
distribution=[uniform|gaussian] angstroms=(float) cycles=(int)/>
```

3.3. Rotate

Randomly rotate the ligand through all rotational degrees of freedom. Specify 360° for full rotational freedom. `Cycles` in this case is much more complicated than seen in `Translate`. Perform up to `cycles` random rotations of the ligand. Only rotations that pass a Lennard-Jones attractive and repulsive score filter are stored. Also, rotations that are close in RMSD to other rotations are not stored. Once a minimum number of diverse structures are collected (this minimum is 5 times the number of ligand rotatable bonds) one of these structures is chosen at random as the starting structure. If no structures passed the attractive and repulsive filter just select the rotation with the best attractive and repulsive score.

This somewhat complicated rotation selection scheme is designed to enrich for hard to find poses, which fit in tight cavities for instance. By storing only rotations that pass an energy filter we limit ourselves to rotations that are close to the protein but do not clash with it. By storing only poses with a minimum RMSD from each other, we increase the probability of selecting "hard to find" poses (classes of similar ligand orientations that easily fit in the interface are only stored once). If you prefer to accept the first rotation, without filtering, just use `cycles = 1`.

```
<Rotate name=(string) chain=(string)
distribution=[uniform|gaussian] degrees=(int) cycles=(int)/>
```

3.4. SlideTogether

After an initial random positioning of the ligand, the ligand must be moved into close proximity to the protein. `SlideTogether` moves the ligand toward the protein, 2 Å at a time, until the two collide (as evidenced by a positive repulsive score). The step size is halved several times (1, 0.5, and 0.25 Å) to minimize the distance between the ligand and the protein. This step proves to be crucial to Rosetta ligand docking. Without it interactions between amino acid side chains and the ligand are rare.

```
<SlideTogether name="&string" chain="&string"/>
```

3.5. HighResDocker

During high resolution docking, `cycles` of rotamer trials (sampling of side chain rotamers, one side chain at a time) and repacking (simultaneous sampling of rotamers for multiple side chains) are combined with small movements of the ligand(s). The size of these movements is described by the `high_res_angstroms` and `high_res_degrees` options of `LIGAND_AREAS` (see Note 3). `LIGAND_AREAS` are part of `INTERFACE_BUILDERs` (see Note 4) which are part of `MOVEMAP_BUILDERs` (see Note 5).

The `movemap_builder` describes which protein residues to include in rotamer trials, repacking, and minimization. If a `resfile` is provided, interface residues are allowed to redesign (change amino acid identity), according to instructions provided in the specified file. Resfiles can also be specified through the command line flag "-packing:resfile." Resfile support allows protein interfaces to be optimized for particular ligands.

The user specifies how many `cycles` of docking and how often to do a full repack (`repack_every_Nth`—only rotamer trials occur in the other cycles). After each cycle the structure is minimized. If `minimize_ligand` values were specified in LIGAND_AREAS then ligand torsion angles are minimized as well. Monte Carlo sampling is used with a Boltzmann criterion to determine whether to accept or reject the new structure after each cycle. If a `tether_ligand` value greater than 0 is specified in `LIGAND_AREAS`, the ligand will be remain within the specified distance (in angstroms). `tether_ligand` prohibits multiple cycles of small translations in the same direction from moving the ligand farther than desired.

```
<HighResDocker name="string" cycles=(int) repack_every_Nth=(&int)
scorefxn="string" movemap_builder="string" resfile="string"/>
```

3.6. FinalMinimizer

Minimize the structure of the docked protein/ligand complex. This includes off-rotamer side-chain torsion angle sampling. The `movemap_builder` specifies which residues to minimize. If `Calpha_restraints` were specified in `LIGAND_AREAS` then backbone φ/Ψ angles are minimized as well.

```
<FinalMinimizer name=(string) chain=(string) scorefxn=(string)
movemap_builder=(string)>
</FinalMinimizer>
```

3.7. InterfaceScore Calculator

This component calculates a myriad of ligand specific scores and appends them to the output file. After scoring the complex the ligand is moved 1,000 Å away from the protein. The model is then scored again. An interface score is calculated for each score term by subtracting separated energy from complex energy. If a `native` structure is specified, four additional score terms are calculated:

1. ligand_centroid_travel. The distance between the native ligand and the ligand in our docked model.

2. ligand_radious_of_gyration. An outstretched conformation would have a high radius of gyration. Ligands tend to bind in outstretched conformations.
3. ligand_rms_no_super. RMSD between the native ligand and the docked ligand.
4. ligand_rms_with_super. RMSD between the native ligand and the docked ligand after aligning the two in XYZ space. This is useful for evaluating how much ligand flexibility was sampled.

```
<InterfaceScoreCalculator name=(string) chains=(comma separated
chars) scorefxn=(string) native=(string)/>
```

3.8. Putting It All Together

Figure 2 presents an XML script that replicates the protocol presented in Davis, 2009 (6). Because of the flexibility of ligand docking through RosettaScripts, it is easy to customize this protocol. For instance high throughput virtual screening of libraries of compounds can be accomplished by spending more time in low resolution docking. Results from low resolution docking can be filtering and used for high resolution docking. A variety of XML elements not specific to ligand docking can also be included as part of a docking study (see the Subheading 2).

A customized ligand docking protocol must take into consideration the number of desired output models (see Note 2), and the amount of time it will take to produce each model, given the available hardware (see Note 6). Best energy output models are then selected for further analysis (see Note 7), and used to generate testable hypotheses about protein/ligand interactions.

4. Notes

1. Score Term reweighting.
   The ligand weights specified in the database file "new.ligand.wts" perform well on a benchmark of diverse protein/ligand complexes. However results can be improved if weights are optimized for the class of protein/ligand interactions one is interested in. We recently used a leave-one-out analysis to improve the correlation between experimental binding energy and rosetta predicted binding energy for HIV-1 protease mutants bound to various protease inhibitors. The leave-one-out weight optimization improves the correlation coefficeint from 0.31 to 0.71.
2. How many models should I make?
   The number of models one should make is largely determined by how large of an interface one is sampling. For this reason

carefully describing the size and shape of an interface can save much compute time. By adjusting the `angstroms` parameter of `Translate` and adding more `StartFrom Coordinates`, a user can restrict sampling to a smaller area. Another strategy is to create a limited number of models, then cluster the results based on RMSD (see Subheading 4, step 4). Select several low energy clusters for further analysis. Select a model from each cluster. Use these models in ligand docking studies, after decreasing the size of `angstroms` in the `Translate` mover.

3. `LIGAND_AREAS`.

   `LIGAND_AREAS` describe parameters specific to each ligand, useful for multiple ligand docking studies (Fig. 1). `cutoff` is the distance in angstroms from the ligand an amino-acid's C-beta atom can be and that residue still be part of the interface. `all_atom_mode` can be `true` or `false`. If `all_atom_mode` is `true` than if any ligand atom is within `cutoff` angstroms of the C-beta atom, that residue becomes part of the interface. If `false`, only the ligand neighbor atom is used to decide if the protein residue is part of the interface. `add_nbr_radius` increases the `cutoff` by the size of the ligand neighbor atom's radius specified in the ligand .params file. This size can be adjusted to represent the size of the ligand, without entering `all_atom_mode`. Thus `all_atom_mode` should not be used with `add_nbr_radius`.

   Ligand minimization can be turned on by specifying a `minimize_ligand` value greater than 0. This value represents the size of one standard deviation of ligand torsion angle rotation (in degrees). By setting `Calpha_restraints` greater than 0, backbone flexibility is enabled. This value represents the size of one standard deviation of Calpha movement, in angstroms.

   During high resolution docking, small amounts of ligand translation and rotation are coupled with cycles of rotamer trials or repacking. These values can be controlled by the `high_res_angstrom` and `high_res_degrees` values respectively. Cycles of small ligand translations can lead to a large translation. In some cases the ligand can "walk away from the protein." The `tether_ligand` option prevents this by keeping the ligand close to its starting point before the high_res_docking step.

   ```
   <[name_of_this_ligand_area] chain="&string" cutoff=(float)
   add_nbr_radius=[true|false] all_atom_mode=[true|false] minimi
   ze_ligand=[float] Calpha_restraints=[float]
   high_res_angstroms=[float] high_res_degrees=[float]
   tether_ligand=[float]/>
   ```

4. `INTERFACE_BUILDERS`.

   An interface builder describes how to choose residues that will be part of a protein-ligand interface. These residues are chosen for repacking, rotamer trials, and backbone minimization during ligand docking. The initial XML parameter is the name of the `interface_builder` (for later reference). `ligand_areas` is a comma separated list of strings matching `LIGAND_AREAS` described previously. Finally `extension_window` surrounds interface residues with residues labeled as "near interface." This is important for backbone minimization, because a residue's backbone can't really move unless it is part of a stretch of residues that are flexible.

   By specifying multiple ligand areas, multiple ligand docking is enabled. Simultaneous docking of multiple ligands, cofactors, water molecules and ions may capture synergistic effects overlooked by serial docking (Fig. 2).

   ```
   <[name_of_this_interface_builder] ligand_areas=(comma separated
   list of predefined ligand_areas) extension_window=(int)/>
   ```

5. `MOVEMAP_BUILDERS`.

   A movemap builder constructs a movemap. A movemap is a $2 \times N$ table of true/false values, where N is the number of residues your protein/ligand complex. The two columns are for backbone and side-chain movements. The movemap builder combines previously constructed backbone and side-chain interfaces (see previous section). Leave out `bb_interface` if you do not want to minimize the backbone. The `minimize_water` option is a global option. If you are docking water molecules as separate ligands (multi-ligand docking) these should be described through `LIGAND_AREAS` and `INTERFACE_BUILDERS`.

   ```
   <[name_of_this_movemap_builder] sc_interface=(string)
   bb_interface=(string) minimize_water=[true|false]/>
   ```

6. How long will this take to run?

   Of course this question depends on many factors: how fast your computer is, how many processors you have access to, how large is your protein? Increasing amino acid rotamers and ligand conformers can increase run-time. Protein backbone and ligand torsion angle minimization also add increase run-time. We have found that the majority of the time is spent in full-repack cycles of ligand docking. Table 1 shows average times for modeling the interaction of Carboxypeptidase A with a phosphonate inhibitor. The XML script in Fig. 1 was used with the exception of modifications shown in column headings.

Table 1
Carboxypeptidase A was docked with a phosphonate inhibitor (PDB code: 7CPA)

Amino acid rotamers	Standard rotamers				Extended rotamers (ex1, ex2)			
Ligand conformations	1	10	100	500	1	10	100	500
rosetta_scripts startup	4.87	4.80	4.87	4.92	4.86	4.87	4.89	4.83
Only setup movers	5.81	5.73	5.76	5.72	5.71	5.77	5.91	5.72
Start From	5.84	5.80	5.80	5.72	5.88	5.74	5.76	5.80
Translate (5, 50)	6.05	6.04	5.88	5.84	5.94	6.04	5.83	5.85
Rotate (360, 1)	6.42	6.37	4.74	6.27	6.40	6.40	4.44	6.27
Rotate (360, 1,000)	76.32	44.81	78.42	40.50	82.94	42.31	68.18	39.71
SlideTogether	5.85	5.98	5.88	5.84	5.85	5.91	5.81	5.87
HighResDocker 1 RT	7.92	7.87	7.89	7.85	8.32	8.29	8.35	8.35
+ MinimizeLigand	8.23	8.21	8.22	8.43	8.32	8.26	8.20	8.34
HighResDocker 1 FR	6.37	6.30	6.38	6.33	11.93	11.85	12.00	11.81
+ Ligand flexibility	6.43	6.38	6.38	6.33	11.77	11.70	11.91	11.84
FinalMinimizer	8.95	8.89	8.98	9.06	8.90	8.89	8.97	9.17
+ Backbone flexibility	14.04	14.26	14.32	13.92	14.04	14.24	14.16	12.26
AddScores	6.02	5.87	5.84	5.95	5.88	5.87	5.77	6.05
Combined	86.77	87.20	95.88	83.35	104.19	98.40	68.36	53.46

The ligand has 9 rotatable bonds. Each datapoint represents the average time in seconds for 10 runs. The combined protocol uses rotate (360, 1,000), `HighResDocker` with ligand flexibility and 6 cycles of packing (full repacks at cycles 1 and 4), and `FinalMinimizer` with backbone flexibility

7. How do I analyze my results?
   When your docking study has finished you will have an output file (specified by the –out:file:atom_tree_diff option) which contains hundreds of models constructed and scored by Rosetta. You can extract these models to individual PDBs using rosetta_scripts. Prepare an XML script that is essentially empty. Under <PROTOCOLS> include this line: <Add mover_name=null/>. Run the XML script with the following command line or flags file options:
   (a) -in:file:atom_tree_diff <input file name>
   (b) -in:file:extra_res_fa <names of .params files>
   (c) –parser:protocol <name of XML file with null mover>
   (d) –database <directory of Rosetta Database>

You may only be interested in the best models by interface score or by total score. You can list the TAGs of the models you wish to extract at the end of the command line. These tags are found in the atom_tree_diff output file after "POSE_TAG." You can search the file for lines that start with "SCORES." By sorting these scores you can find the lowest energy models.

You can also use the Rosetta Cluster application to group your models by RMSD. Then you can choose one low energy model from several low energy clusters for further analysis. See the cluster documentation (http://www.rosettacommons.org/manuals/archive/rosetta3.1_user_guide/app_cluster.html) for more information.

References

1. Meiler, J., and Baker, D. (2006) ROSETTALIGAND: protein-small molecule docking with full side-chain flexibility, *Proteins 65*, 538–548.
2. Kaufmann, K. W., Lemmon, G. H., Deluca, S. L., Sheehan, J. H., and Meiler, J. (2010) Practically useful: what the Rosetta protein modeling suite can do for you, *Biochemistry 49*, 2987–2998.
3. Raman, S., Vernon, R., Thompson, J., Tyka, M., Sadreyev, R., Pei, J. M., Kim, D., Kellogg, E., DiMaio, F., Lange, O., Kinch, L., Sheffler, W., Kim, B. H., Das, R., Grishin, N. V., and Baker, D. (2009) Structure prediction for CASP8 with all-atom refinement using Rosetta, *Proteins-Structure Function and Bioinformatics 77*, 89–99.
4. Chaudhury, S., and Gray, J. J. (2008) Conformer selection and induced fit in flexible backbone protein-protein docking using computational and NMR ensembles, *J Mol Biol 381*, 1068–1087.
5. Jiang, L., Althoff, E. A., Clemente, F. R., Doyle, L., Rothlisberger, D., Zanghellini, A., Gallaher, J. L., Betker, J. L., Tanaka, F., Barbas, C. F., Hilvert, D., Houk, K. N., Stoddard, B. L., and Baker, D. (2008) De novo computational design of retro-aldol enzymes, *Science 319*, 1387–1391.
6. Davis, I. W., and Baker, D. (2009) RosettaLigand docking with full ligand and receptor flexibility, *J Mol Biol 385*, 381–392.
7. Kaufmann, K. W., Glab, K., Mueller, R., and Meiler, J. (2008) Small Molecule Rotamers Enable Simultaneous Optimization of Small Molecule and Protein Degrees of Freedom in ROSETTALIGAND Docking, In *German Conference on Bioinformatics* (Beyer, A., and Schroeder, M., Eds.), pp 148–157, Dresden.
8. Davis, I. W., Raha, K., Head, M. S., and Baker, D. (2009) Blind docking of pharmaceutically relevant compounds using RosettaLigand, *Protein Sci 18*, 1998–2002.
9. Kaufmann, K. W., Dawson, E. S., Henry, L. K., Field, J. R., Blakely, R. D., and Meiler, J. (2009) Structural determinants of species-selective substrate recognition in human and Drosophila serotonin transporters revealed through computational docking studies, *Proteins 74*, 630–642.