
HelmholtzZentrum münchen
Deutsches Forschungszentrum für Gesundheit und Umwelt


**Marie Curie Initial Training Network**

**Environmental Chemoinformatics (ECO)**

**Final Report**

**19 January 2012**

# *Development of docking approaches for toxicity prediction*

**Early stage researcher:**
Ehret Jacques

**Project supervisors:**
Dr. Igor V.Tetko and Mr Sergey Novotarskyi

**Research Institution:**
Helmholtz Zentrum München

# Literature

## Principles of docking

Molecular docking is a computer simulation procedure that attempts to find the "best" matching between two molecules: a receptor and a ligand. One can say a "complex" when the ligand is docked into a receptor. Generally, receptors are proteins or nucleic acid molecules (DNA or RNA). Ligands can either be proteins or small molecules (this last case can be called protein-drug docking). The molecular docking problem can be defined as follows: Given the atomic coordinates of a receptor and a ligand, predict their "correct" bound association i.e. close to experimental measurements (Halperin, 2002). To solve this problem, each docking program uses one or more search algorithms and scoring functions. Search algorithms are the methods used to predict the possible conformations of the complex. Scoring functions are employed in order to optimize and rank results. There are several possible applications to docking and the most common is drug design. Drugs are usually small molecules that alter the property or the activity of biological entities in the body in order to provide a specific effect. It can be for instance the activation of a protein that intervenes in the generation of antibodies. Interactions between a receptor and a ligand lead to activate or inactivate the receptor. That is why docking is an important computational tool in drug discovery.

There are three key ingredients in docking: (1) representation of the system, (2) conformational space search, and (3) ranking of potential solutions (Halperin, 2002). The basic description of the receptor (or ligand) surface is the atomic representation of exposed residues. More often, its geometric features such as Connoly surfaces represent the surface. One can use a grid representation as well. It was shown that small changes in the ligand representation could lead to drastic changes in the docking results so using multiple inputs is recommended (Yuriev, 2011). Then there are many ways of putting two molecules together (three translational and three rotational degrees of freedom for each evaluated conformation of each complex). The number of possibilities grows exponentially with the size of the components, so docking calculations can be really computationally intensive time. Moreover, the more flexibility is taken into consideration, the more calculation time it will need. In fact,

each rotatable bond increases the degree of liberty of the molecule, so the amount of possible conformations. Three levels of approximations can be used to reduce the calculation time: (1) rigid body docking. Rigid body is a highly simplistic model that regards the receptor and the ligand as two rigid solid bodies. It is useful though when ligands are tested with their experimentally observed conformation. (2) Semi-flexible docking. One of the molecules, usually the ligand, is considered flexible while the other is regarded as rigid. (3) Flexible docking. Both molecules are considered flexible, although usually not integrally. The flexibility of one or both is necessarily limited, or simplified. For instance only flexibility of the binding site and its close environment can be taken into account. Rigid docking has the advantage of speed while flexible docking provides more accurate results. The search for candidate solutions in a docking problem is addressed in two essentially different approaches: (1) a full solution space search in contrast to (2) a gradual guided progression through solution space. The second approach consists mainly of Monte Carlo (MC), simulated annealing, molecular dynamics (MD), and evolutionary algorithms such as genetic algorithms (GA) and Tabu search (Dias 2008). A search algorithm returns many possible orientations (poses) of a ligand in the target's biding site, unmanageable for any practical need. Scoring functions, which are able to evaluate intermolecular binding affinity or binding free energy, are employed in order to optimize and rank results, obtaining the best orientation after the docking procedure. The two critical elements in a search procedure are speed and effectiveness in covering the relevant conformational space. On the other hand, the scoring function should be fast enough to allow its application to a large number of potential solutions. Ideally, the best matching algorithms and scoring schemes should be combined (Halperin 2002).

Depending on the algorithms, matching and scoring can be expansive in computational time. To screen a huge library, it is common to filter it first using fast algorithm, and then investigate deeper with slower ones. Hereafter stands a general scheme that can be used for virtual screening (Dias 2008):

1. Setup of Small molecules Data Base
2. Rigid-body Docking

3. Specific Library with best ranked Results
4. Flexible Docking
5. Ligand-binding Affinity Evaluation
6. Molecular Dynamics Simulations

## Existing tools

Many docking programs exist, with different algorithm. None is universal, so to solve a docking problem one should use several software / approaches. Hereafter stands a non-exhaustive table containing some existing programs with their search method.

| Program | Search Method |
|---|---|
| Autodock 4.0 | Genetic Algorithm |
| Autodock Vina | BFGS + quasi-newton method |
| ICM | Monte Carlo simulations |
| DOCK | Incremental Construction |
| MS-DOCK | Fast Shape Matching |
| GOLD | Genetic Algorithm |
| Surflex | Incremental Construction |

Table 01: Some docking software and their search algorithm

## OCHEM

The host laboratory developed OCHEM (Online Chemical Database with Modeling Environment). It is a web application (http://ochem.eu/) programmed in java. The OCHEM is an online database of experimental measurements integrated with the modeling environment. One can submit experimental data or use the data uploaded by other users to build predictive QSAR models for physical-chemical or biological properties (I. Sushko and all, 2011). One can build QSAR/QSPR models using descriptors that are dependent of conformations (3D descriptors for instance). If those conformations are predicted by docking simulations, it can lead to more accurate predictions of activity. That is why we chose to implement the docking workflow in OCHEM. Moreover in the host laboratory docking-based descriptors considering atoms pair between the protein and the ligand were developed. Those can be used to calculate predictive model applicable to docking issues.

### Bibliography

*Principles of Docking: An Overview of Search Algorithms and a Guide to Scoring Functions*; Inbal Halperin, Buyong Ma, Haim Wolfson, and Ruth Nussinov; PROTEINS: Structure, Function, and Genetics 47:409–443 (**2002**).

*Optimizing the use of open-source software applications in drug discovery*; Werner J. eldenhuys, Kevin E. Gaasch, Mark Watson, David D. Allen and Cornelis J. Van der Schyf; DDT Volume 11, Number 3/4 ; **February 2006**.

*Molecular Docking Algorithms*; Raquel Dias and Walter Filgueira de Azevedo Jr.; Current Drug Targets, 9:1040-1047 (2008).

*Challenges and advances in computational docking: 2009 in review*; Elizabeth Yurieva, Mark Agostino and Paul A. Ramsland; J. Mol. Recognit. 24: 149–164 (**2011**)

*Online chemical modeling environment (OCHEM): web platform for data storage, model development and publishing of chemical information*; Sushko I, Novotarskyi S, Körner R, Pandey AK, Rupp M, Teetz W, Brandmaier S, Abdelaziz A, Prokopenko VV, Tanchuk VY, Todeschini R, Varnek A, Marcou G, Ertl P, Potemkin V, Grishina M, Gasteiger J, Schwab C, Baskin II, Palyulin VA, Radchenko EV, Welsh WJ, Kholodovych V, Chekmarev D, Cherkasov A, Aires-de-Sousa J, Zhang QY, Bender A, Nigsch F, Patiny L, Williams A, Tkachenko V, Tetko IV; J Comput Aided Mol Des. 25(6): 533-554 (**2011**)

## Docking software to implement

### AutoDock Vina

Autodock is a software package programmed in python and C++, which contains several tools related to docking issues. Some of the scripts used by it can preprocess files that will be used by another tool developed by Autodock: vina. It is an open-source docking program using same format and preprocessing as its predecessor, Autodock4.0. It has been designed and implemented by Dr. Oleg Trott in the Molecular Graphics Lab at The Scripps Research Institute.

Vina uses a sophisticated gradient optimization method in its local optimization procedure. The calculation of the gradient effectively gives the optimization algorithm

a “sense of direction” from a single evaluation. By using multithreading, Vina can further speed up the execution by taking advantage of multiple CPUs or CPU cores. The binding affinity formula (Figure 1) is an empirical scoring function, and so takes into account enthalpic and entropic effects.

$$\Delta G = f_{vdw} \sum \left( \frac{A_{ij}}{r_{ij}^{12}} - \frac{B_{ij}}{r_{ij}^{6}} \right) + f_{hbond} \left[ \sum E(\theta) \left( \frac{\alpha_{iH}}{r_{iH}^{12}} - \frac{\beta_{iH}}{r_{iH}^{10}} \right) + E_{Hbond} \right]$$
$$+ f_{elec} \sum \left( \frac{q_i q_j}{e(r_{ij}) r_{ij}} \right) + \Delta G_{tors} N_{tors} + f_{solv} \sum \left( S_i V_j + S_j V_i \right) e^{-r_{ij}^2}$$

**Figure 1:** Binding Affinity formula in vina

The software is available from the website http://vina.scripps.edu.

We chose it because it usually reaches good results (Oleg TROTT, 2010), it is lightweight (2MB), it is available for Mac and Linux system that makes it easier to dispatch on OCHEM cluster, and it is under an Apache license.

The inconvenient of this software is that the preprocessing is not straightforward. In fact, ligands and protein have to be preprocessed before being submitted to Vina docking. Figure 3 represents this process. Each tool on this scheme are described further.

## Bibliography:

*AutoDock Vina: Improving the Speed and Accuracy of Docking with a New Scoring Function, Efficient Optimization, and Multithreading*; Oleg TROTT, Athur J. OLSON; Journal of Computational Chemistry 31(2): 455-461 (**2010**)

# Database

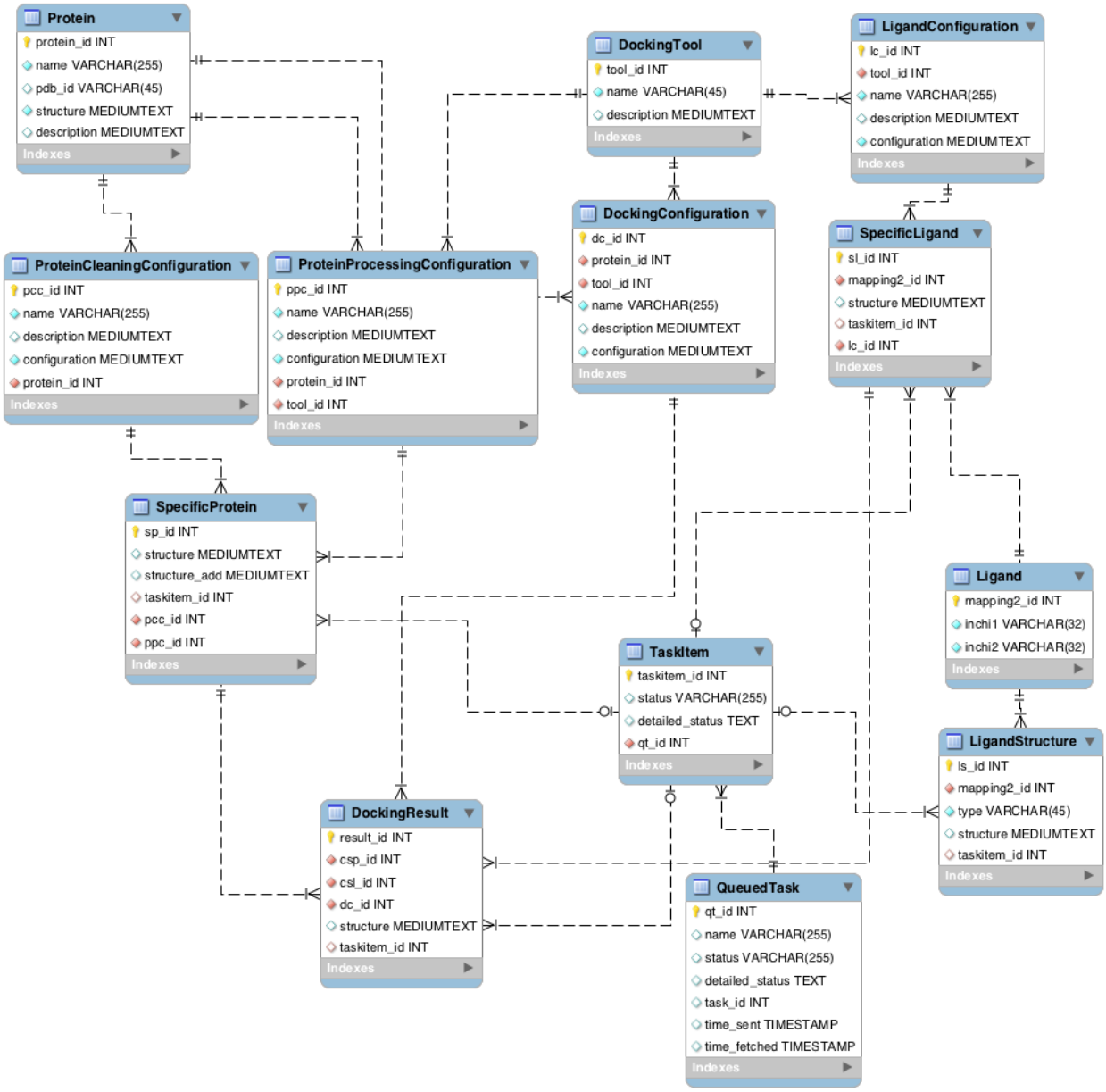


**Figure 2:** Database conceived fort he docking workflow

It was necessary to build our own database for this project. The database schema is illustrated on Figure 2. It was developed to support several docking tools, and store a maximum of information about each docking result to avoid useless redundant calculations.

There are 13 tables in the database. All relationships between them can be seen on Figure 2. Hereafter stands a brief description of each table.

Storage of Proteins: Table “Protein” is used to store basic data on each protein, namely name, protein data bank id, and raw structure, without any preprocessing. Usually the

structure is taken directly from the PDBank. Two configuration tables contain information used to preprocess the raw protein structures before docking. The first table, “Protein Cleaning Configuration” contains parameters used to refine protein structure (i.e. to remove any docked ligands or other heteroatoms). The second table “Protein Processing Configuration” contains the configuration used to preprocess a specific protein (with its flexible residues if there are some, which kind of charges were added and so on). Those two tables can be used to check if such configurations have ever been used for docking calculations or not. In case if it was used, there is no need to recalculate it. Table “Specific Protein” contains the preprocessed structure after cleaning and preprocessing the file, depending on the docking tool to be used. As foreign key are the id of the initial protein and the corresponding cleaning and preprocessing configuration’s id. The structure field for this table will be the pdbqt file content in case of preprocessing for vina docking, and the “structure_add” field will be the flexible residues pdbqt file content for a same case.

Storage of Ligands: Table “Ligand” is used to store ID and INCHI key of each ligand. Table “Ligand Structure” contains the initial structure of the ligand. Table “Ligand Configuration” contains the configuration used to preprocess ligands. It includes rotatable bonds, type of 3D optimization, type of tools used to calculate charges of atoms (e.g., using vina docking tool, external ones, etc.) and other information. Table “Specific Ligand” contains the structure of each ligand after the preprocessing. It is linked with a foreign key (“lc_id”) to the corresponding configuration.

Configuration of Docking: Table “Docking Tool” contains names of each tool existing in the docking workflow. It can be the docking tool by itself or a tool used during the preprocessing of data. Table “Docking Configuration” contains the arguments that were used with which docking software in which case, with for instance binding pocket’s grid, size and center of the calculation box, etc. Table “Docking Result” contains the output of the docking tool for each docked ligand, in which binding site, and using which software.

Tasks: Table “Task Item” contains each task that is supposed to be done. Table “Queued Task” contains all posted tasks that are waiting for some results or some place on the server. The field “status” provides details about the status of the task such as “waiting” if the task has to be calculated or “done” if the calculations are finished.

# Implementation

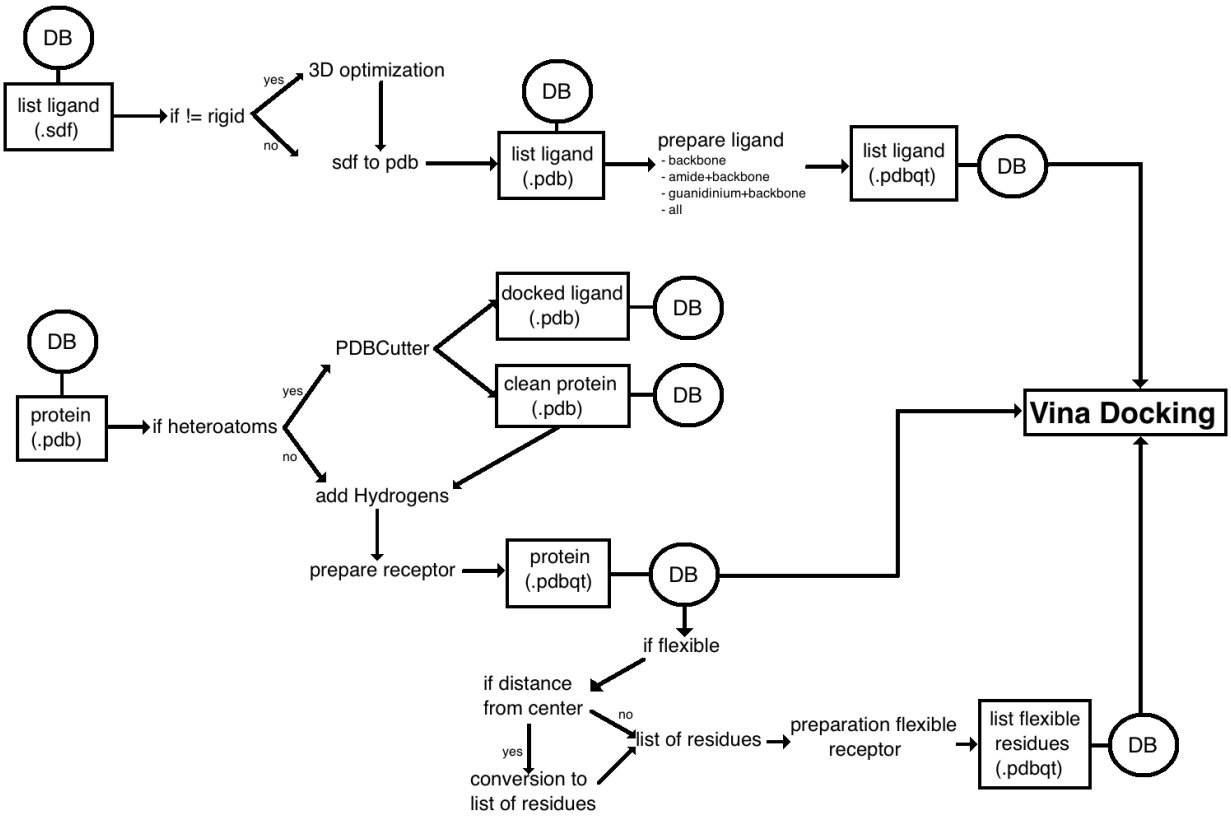


**Figure 3:** Scheme of the workflow for Vina Docking

## Autodock Tools:

To use Vina, one must preprocess files using Autodock tools. There are 4 scripts or binaries that are used for that purpose. Those are prepareligand4.py, preparereceptor4.py, prepareflexreceptor4.py, and Vina docking. To maximize the accuracy of calculations, we decided to let some choices to the user, usually about the rotability of ligand bonds and protein residues. The following tables contain the arguments of each script that are supported by our workflow. Further details on the use of each script stand after the four tables.

Prepare ligand script's arguments:

| Option | Argument | Specificities |
|---|---|---|
| -l | Ligand filename | .pdb format |
| -o | Output filename | .pdbqt format |
| -B | Type(s) of bonds to allow to rotate | Can be backbone, backbone + amide, backbone + guanidinium, backbone + guanidinium + amide. |
| -Z | Inactivate all torsions | |

Prepare receptor script's arguments:

| Option | Argument | Specificities |
|---|---|---|
| -r | Receptor filename | .pdb format |
| -o | Output filename | .pdbqt format |
| -A | Type of repair: defined as checkhydrogens | Add hydrogen only if there are none already in the file |

Prepare flexible receptor script's arguments:

| Option | Argument | Specificities |
|---|---|---|
| -l | Ligand filename | .pdbqt format |
| -r | Receptor filename | .pdbqt format |
| -s | Names of flex residues | |
| -g | Rigid output filename | .pdbqt format |
| -x | Flexible output filename | .pdbqt format |

Vina docking arguments:

| Option | Argument | Specificities |
|---|---|---|
| -receptor | Rigid part of the receptor | .pdbqt format |
| -flex | File containing receptor's flexible side chains, if any | .pdbqt format |
| -ligand | File containing ligand | .pdbqt format |
| -center_x | X coordinate of the docking box's center | |
| -center_y | Y coordinate of the docking box's center | |
| -center_z | Z coordinate of the docking box's center | |
| -size_x | Size of the docking box, X axe | Angstrom |
| -size_y | Size of the docking box, Y axe | Angstrom |
| -size_z | Size of the docking box, Z axe | Angstrom |
| -cpu | Number of CPUs to use | Set at 1 |
| -num_modes | Max binding modes to generate | Set at 10 |

Autodock vina needs pdbqt files for ligand and proteins as input. This format is an extension of pdb format, developed for Autodock tools. It contains rotatable bonds explicitly defined (for the ligand file) and partial charges of each atom. For flexible docking, an additional file containing the rotatable residues is required. Moreover, it needs coordinates and size (in Angstrom) of the box that has to be considered during the docking (i.e. containing the site where the ligand is supposed to fit). In our database, ligands are stored in sdf format and proteins in pdb, so the formats are modified during the preprocessing to finally have pdbqt for both ligand and protein.

We chose to offer several possibilities to the user that could modify the result of the preprocessing and thus the docking results. With respect to ligands, the user can choose flexible or rigid ligands. A rigid case would be useful for specific studies, i.e., analysis of ligands with conformation experimentally observed in a complex for instance. For such cases, 3D optimization should not be done. In flexible cases (most typical cases), the Autodock script offers three possibilities: set as rotatable a) the backbones, b) the amide chains, or c) the guanidinium chains. We kept the possibility to choose one or combine some of those choices in our workflow. Moreover, hydrogen atoms are explicitly added in the file in case if they were not present for the ligands. Similarly for proteins, we check a presence of hydrogen atoms and add them if they are absent. In case of a flexible docking, a third script is called to build the third pdbqt file. This method requires information which residues should be considered as flexible ones. We provide two choices: user can either supply the list of flexible residues or provide a distance (in Angstrom) from the center of the box to automatically detect flexible residues for docking.

**CADDSuite tools:** PDBCutter, ProteinProtonator.

CADDSuite tool as its name suggests, is a suite of tools that are useful for computational chemistry analyses. Some of them are specially designed for molecular docking. We used two of them: PDBCutter and ProteinProtonator.

PDBCutter arguments that are supported by our workflow:

| Option | Argument | Specificities |
|---|---|---|
| -i | Input file | .pdb format |
| -rec | Receptor output file | .pdb format |
| -lig | Ligand output file | .pdb format |
| -lig_chain | Ligand chain-name | |
| -lig_name | Ligand residue name | |
| -rm_ch | id of chains to delete | As A, B, C, … |
| -rm_res | Name of residues to delete | As HOH, ions, … |

Protein files can be found in the Protein Data Bank (PDB, available on www.rcsb.org/). This database includes a lot of experimental complexes, which means

that the file contains the protein with bound ligand(s). Often it also contains some other heteroatoms such as water molecules or metal atoms. In a docking procedure, it is required to use a free binding site, i.e. without the ligand. Moreover, the presence of heteroatoms in the pocket can disturb interactions of the analyzed ligands. In addition, the bound metals can modify the shape and the rigidity of a protein. That is why depending on the studied case/medium/conditions it can be useful to filter all those heteroatoms. PDBCutter is then used during the preprocessing. It can separate a complex into protein and ligand, and take off heteroatoms such as water or bound metals. In principle bound ligand can be removed from PDB file as part of standard pre-processing procedure. However, some docking tools (as IMGDock from CADDSuite) need the docked ligand atoms coordinates to identify the active site and map a grid around it. Therefore, we store in our database entire ligand-protein complex if a docked ligand is available in the initial PDB file.

ProteinProtonator, as its name suggests, is used during the data preprocessing to add hydrogen atoms in proteins. Hydrogen atoms are usually implicit in pdb files. During a docking procedure, steric and electrostatic interactions interact with the score calculation. If hydrogen atoms are not taken into account during the scoring procedure, the result could be wrong. That is why it is necessary to add hydrogen to the studied receptor. Some, but not all, software tools do it by default (as prepareреceptor4.py script in Autodock tools). That is why this preprocessing is a necessary step in our docking workflow.

# Application

OCHEM's database contains a variety of datasets, including 15983 molecules with known activity (quantitative and qualitative) to the human aryl-hydrocarbon receptor (hAhR). The initial data for this activity (activation of AhR) were taken from the pubchem bioassay database (AID: 2796). The structure of the receptor is not experimentally known, but M. Salzano propounds in her paper a structure deduced by homology modeling techniques (M. Salzano, 2011). She suggests the presence of multiple binding sites (A, B, C, D) in hAhR. Moreover, it is supposed that the A pocket

maybe describes not a real pocket but a crevice on the surface. The author provides the positions of the residues describing the pockets, and the protein file. There are several differences in data described in the article and in the provided protein file: the id number is different between the residue in the file and in the publication. This difference is about 274 id (the first residue is number 1 in the file and 274 in the publication).Then the residue LEU342 (in the paper) is LYS68 (in the file) due to an error with the naming of the amino acid in the paper.

The study of Salzano deals only with ten molecules. To test our workflow, we decided to dock all ligands with known activity to this receptor from OCHEM to all four pockets. In our workflow, using vina docking, the output contains the binding affinity with the receptor (formula on Figure 1) and also the coordinates of each docked ligand. The binding energy can be directly correlated with activity to bind ligands. The coordinates could be used to calculate 3D or docking-based descriptors in order to build QSAR/QSPR models. But the latter feature is not fully implemented yet, the link between the docking results and the descriptors calculation workflow will be added in the future. Threfore, we checked if there was a significant correlation between the activity and the ligand/protein affinity in one of the binding sites. Let consider the multiple binding sites hypothesis as true and the provided structure as similar to the real one. The more interaction between ligand and receptor there is, the more chances to modify an activity there should be. So one can think that an active ligand should fit with high affinity in binding pockets (B, C, or D). Moreover, the pocket A is supposed to be only a crevice so no significant correlation between the activity of the ligand and the affinity should be found. However, we found out that calculated affinity did not correlate with activity (quantitatively or qualitatively speaking). Figures 4-7 contain graphs plotting the binding affinity in abscissa and the measured property in ordinates. One can see that there is no apparent correlation between the quantitative activity and the calculated binding affinity. A paper (A. Marabotti, 2011) mentioned that docking tools usually provide good poses but “scoring functions seem to perform less well than e.g. machine learning methods”. We showed that scoring functions are not directly correlated with the activity. That can be explained by the fact that molecules with strong affinity will fit but not necessarily modify the shape of the protein, so there

will be no effect. On the other way around, a molecule with a weak affinity, when approaching the binding site could modify the structure of it that would lead to an activation of the protein and a new affinity, stronger. Moreover, so many effects intervene in biological activity that a description based only on such affinity is not enough: more parameters should be taken into account. Using docked coordinates could lead to better results, and will be tested in a future. Another explanation can be that activation of AhR receptors is not correlated with binding activity to four aforementioned binding sites.

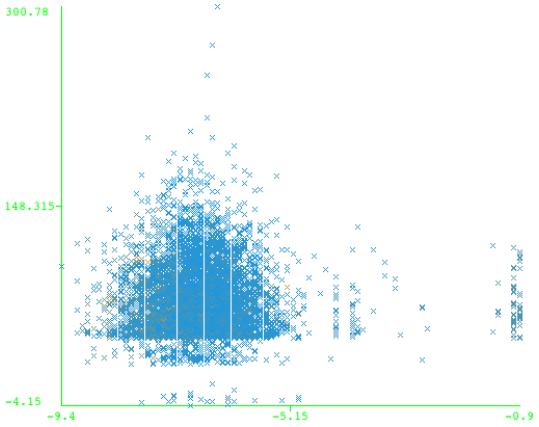


**Figure 4:** Binding Affinity/Property; siteA

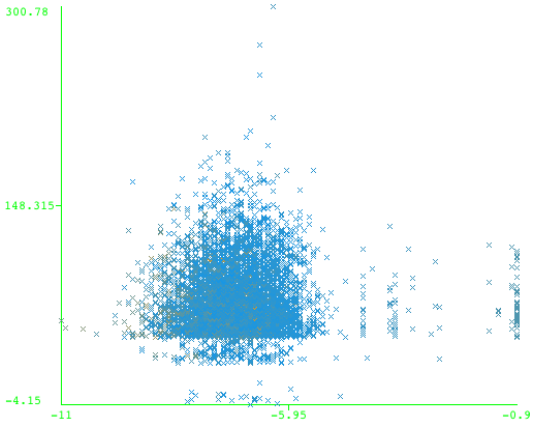


**Figure 5:** Binding Affinity/Property; site B

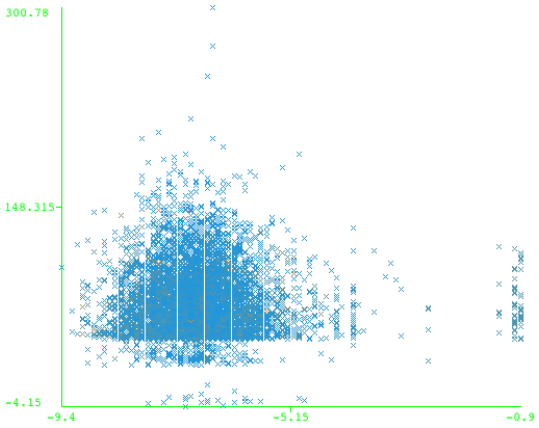


**Figure 6:** Binding Affinity/Property; site C

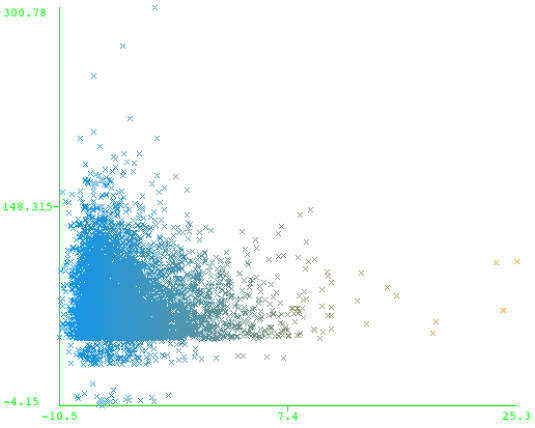


**Figure 7:** Binding Affinity/Property; site D

## Bibliography:

*Human aryl-hydrocarbon receptor and its interaction with dioxin and physiological ligands investigated by molecular modeling and docking simulations*; Maria Salzano, Anna Marabotti, Luciano Milanesi, Angelo Facchiano; Biochemical and Biophysical Research Communications 413:176–181 (2011);

*Virtual Screening and Prediction of Site of Metabolism for Cytochrome P450 1A2 Ligands*; Poongavanam Vasanthanathan, Jozef Hritz, Olivier Taboureau, Lars Olsen, Flemming Steen Jørgensen, Nico P. E. Vermeulen, and Chris Oostenbrink; Journal of Chemical Information and Modeling 49 (1): 43-52 (**2009**)

# Outcomes

The database and the docking workflow were developed and can be launched using command lines. Therefore, current calculations require some programming efforts. A graphical interface will need to be added, so the web users can use the workflow as well. Moreover, a link between calculation of descriptors and the docking workflow will be added as well to allow inclusion of docking results in QSAR predictions.

The database and development was designed to handle several docking tools. So other software could be easily added. In order to do it, several Java classes including calculation server and configuration should be added as well as a proper preprocessing of structures and ligands required for each respective tool. Considering that docking results are depend on the used software the use of several docking tools with different scoring and posing scenarios can increase quality of docking and provide better results.

It would be also interesting to continue analysis of AhR ligands using QSAR. The use of the binding energy with 3D descriptors to represent docked structures can provide a good set of descriptors for such study. Moreover, the use of chemical expert knowledge to identify the active binding conformation of ligands on one of the four possible binding sites can further increase the accuracy of model.