

## Chapter 2

# Chemoinformatics Approach for the Design and Screening of Focused Virtual Libraries

**Abstract** It is challenging to handle a large volume of molecular data without appropriate tools. Here, we describe the need and the approaches for the development of focussed virtual libraries to design efficient molecules and optimize them for lead generation. The experimental chemists and biologists are more interested in properties of chemicals and their response to biological system in both beneficial and adverse effects context rather than just their structures. In this chapter, the focus is to relate newly designed chemical structures to their predicted activity, property or toxicity. Property prediction tools save time, money and lives of experimental animals. They come in handy while taking informed decisions especially in certain cases involving pharmacodynamic studies of drug molecules in humans where there are inevitable ethical and safety concerns. Property prediction is an important component in virtual screening which is at the heart of drug design and the most important step where chemoinformatics plays a major role. The other fields where structure–activity relation-based principles hold good for virtual screening are agrochemicals and environmental science, specifically the toxicity and biodegradability prediction of pollutant molecules. In this chapter, we will show how to design software tools to handle generation of focussed virtual libraries from a given set of molecules with common features, fragments or bioactivity spectrum.

**Keywords** Descriptors • Chemical properties • Chemoinformatics • Drug design

## 2.1 Introduction to Structure–Property Correlations

Chemists are mainly interested in the structure of chemicals to know those properties which can be of some use to us. Physico-chemical properties, bioactivities and toxicity-related data of chemicals available from scientific literature or from experimental results are used for building predictive models applying advanced mathematical methods or machine learning techniques based on the principle of ‘similar structures possess similar property’ [1–3]. The quality of predictive models basically depends on the selection of relevant molecular descriptors and accuracy of experimental data [4]. Basically, molecular descriptors are the structural features

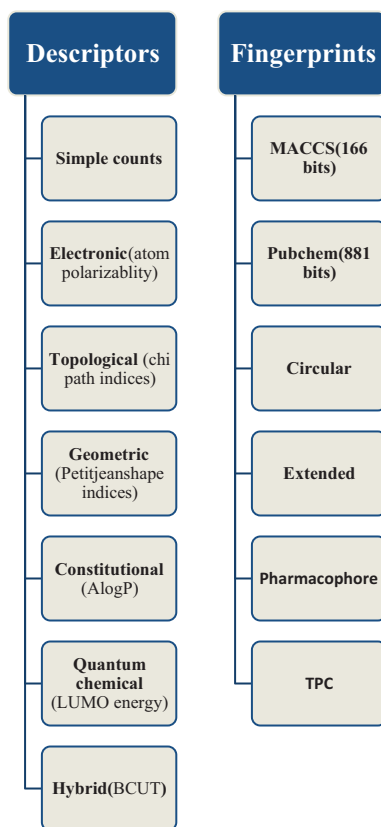
encoding independent property of interest such as activity, property and toxicity [5]. The relation between structure and property is studied by computing binary fingerprints and descriptors from the molecular graph and its three-dimensional (3D) chemical structure respectively [6].

### 2.1.1 Descriptors

Descriptors are properties that describe a molecule on the basis of either some physico-chemical property like melting point, boiling point or an algorithm like two-dimensional (2D) fingerprint [7]. There are several types of molecular descriptors and features used for establishing structure–property links. Most commonly used molecular descriptors are constitutional, surface, molecular connectivity, electrostatic, shape, geometry, quantum chemical, physico-chemical, hybrid, etc. which are all intimately related to each other [8]. The constitutional descriptors are the most simple and common ones that just provide information on the chemical composition of molecules [9]. The topological descriptors which encode the surface properties of a molecule are used to ascertain the solubility and permeability of a proposed drug. Electrostatic descriptors such as polarizability, dipole moment and ionization energy predict crystalline density [10]. Geometrical or 3D descriptors based on xyz coordinates provide rich information regarding a molecule's orientation in space and are often more useful than others in predicting biological activity [11]. Quantum chemical descriptors in theory encompass all the electronic and geometrical features of a molecule compared to empirical ones, the only drawback being the computational overload [12]. Some of the quantum chemical descriptors include lowest unoccupied molecular orbital (LUMO) energies, orbital electron density, delocalizability, etc. [13]. Hybrid descriptors such as BCUT [14] WHIM [15] were initially developed for chemical diversity but later found useful as inputs for building predictive models. Another class of descriptors include the binary bit string-based fingerprint descriptors which are employed for similarity searching in databases. The known literature fingerprints, viz. Molecular Design Limited Molecular ACCess System (MDL MACCS) 166-bit keys [16], circular fingerprints [17], Extended Convective Forecast Product (ECFP) [18], FCF2 [19], Unity [20], PubChem fingerprints [21] and TPC [22], have been applied to a wide range of applications including prediction of absorption, distribution, metabolism, excretion and toxicity properties (Fig. 2.1).

From a drug discovery point of view, the most important descriptor among molecular properties is the solubility of a compound [23]. This in turn impacts the oral bioavailability of a drug—an important pharmacokinetic parameter [24]. Solubility is also found to be an important parameter for lipid-based formulation excipients in pharmacy [25]. Another equally relevant descriptor is logP, i.e. the water/octanol partition coefficient [26]. The prior knowledge of these descriptors is considered important during the preclinical trial stage in the drug discovery pipeline. Currently, descriptors for target and ligand are computed simultaneously for predicting side effects in drugs and polypharmacology, an emerging concept in medicine, wherein other therapeutic options are explored for a known marketed drug [27]. Apart from drug design, another field where descriptors play an important role is material science where

**Fig. 2.1** Commonly employed descriptors and fingerprints in structure–property correlation studies



the selection of a right descriptor can lead to improved energetic substances [28]. By evaluating molecular, microscopic and structural descriptors of an adsorbate–adsorbent system, single-component adsorption isotherms can be predicted [29].

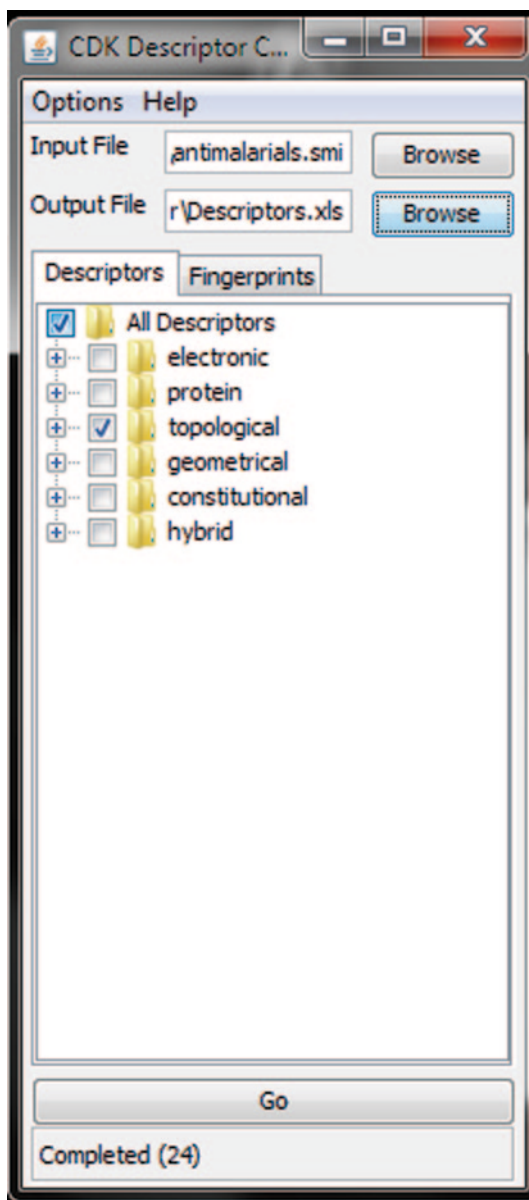
In this section, we shall practically see how to compute descriptors using open-source, free, commercial tools for a given set of molecules. The right choice of independent uncorrelated descriptors is the next important step. Genetic algorithm (GA)-based approaches are employed to select the optimal subset of descriptors [30]. Many linear and non-linear models to predict a physico-chemical property or bioactivity can be built using selected descriptors by employing machine learning methods like neural networks which are discussed in detail in the next chapter.

### 2.1.1.1 Open-Source Tools for Computing Descriptors

#### Chemistry Development Kit

SMILES notation of a molecule can be input to calculate properties/descriptors using open-source programs. The Chemistry Development Kit (CDK) is a scientific, Lesser General Public License (LGPL)-ed library for bio- and cheminformat-

**Fig. 2.2** Chemistry Development Kit (CDK) descriptors calculator



ics and computational chemistry written in Java [31]. A CDK descriptor calculator (v1.3.8) has been developed for cdk1.5 which calculates descriptors and fingerprints given a .smi or .sdf input file [32]. The user can select only the type of descriptor; the program currently uses a default parameter setting for each descriptor (Fig. 2.2).

## JOELib

It is an integrated chemoinformatics package governed by the GNU general public license [33]. The Java libraries are available at its homepage site. The descriptors include simple atom group counts which are good enough to build primitive quantitative structure–property relationships (QSPR) models but for predicting complex biological properties transformed descriptors should be computed. One can also write their own descriptor and classes into the program.

*Source Code for Computing JOELib Descriptors from Simplified Molecular-Input Line-Entry System format of Any Molecule*

```

public String[] getData(String smi) {
    BasicDescriptors bd = new BasicDescriptors();
    JOEMol mol = new JOEMol(IOTypeHolder.instance().getIOType("SMILES"),
    IOTypeHolder.instance().getIOType("SDF"));
    String[] out = new String[2];
    try {
        JOESmilesParser.smiToMol(mol, smi, "mol_name");
        double logP = 0;
        int promiscuous = 0;
        bd.computeDescriptors(mol, logP, promiscuous);
        DecimalFormat dfl = new DecimalFormat("####.#####");
        LogP lp = new LogP();
        bd.logP = lp.getDoubleValue(mol);
        out[0] = "";
        out[0] += "HBD:" + bd.hbd + ";LogP:" + dfl.format(bd.logP) + ";M.Wt:" +
        dfl.format(bd.mw) + ";Promiscuous:" + bd.promiscuous + ";TPSA:" +
        dfl.format(bd.tpsa) + ";Basic Score:" + bd.basicScore() + ";HBA:" + bd.hba + ";DL
        Failures:" + bd.drugLikeFailures() + ";LL Failures:";
        out[0] += bd.drugLikeFailures() + ";";
        out[0] += bd.basicScore() + ";PDL:" +
        dfl.format(bd.pdl) + ";PLL:" + dfl.format(bd.pll) + ";CFMS Penalties:" +
        bd.cfmsPenalties() + ";";
        out[1] = bd.stringsKey3DS;
        out[0] += ";numberOfBadAtoms :" + bd.numberBadAtoms;
        out[0] += ";numberOfCF3 :" + bd.numberCF3;
        out[0] += ";numberOfN :" + bd.numberN;
        out[0] += ";numberOfNO2 :" + bd.numberNO2;
        out[0] += ";numberOfO :" + bd.numberO;
        out[0] += ";numberOfS :" + bd.numberS;
        out[0] += ";numberOfSO2 :" + bd.numberSO2;
        out[0] += ";numberOfX: " + bd.numberX;
        String[] rp = bd.reactivePatterns;
        for (int i = 0; i < rp.length; i++) {
            out[0] += ";numberOf RP" + i + ":" + rp[i];
        }
        String[] wp = bd.warheadPatterns;
        for (int i = 0; i < wp.length; i++) {
            out[0] += ";numberOf WHP" + i + ":" + wp[i];
        }
        getSMPatterns sm = new getSMPatterns();
        String[] out1 = sm.getToxicophoreFP(smi); //toxicophoreFingerprints
        for (int i = 0; i < out1.length; i++) {
            out[0] += ";toxph FP:" + i + ":" + out1[i];
        }
        String[] out2 = sm.getChemClassFP(smi); //ChemicalClassFP
        for (int i = 0; i < out2.length; i++) {
            out[0] += ";chem FP:" + i + ":" + out2[i];
        }
    } catch (Exception e) {
        System.out.println(e);
    }

    return out;
}

```

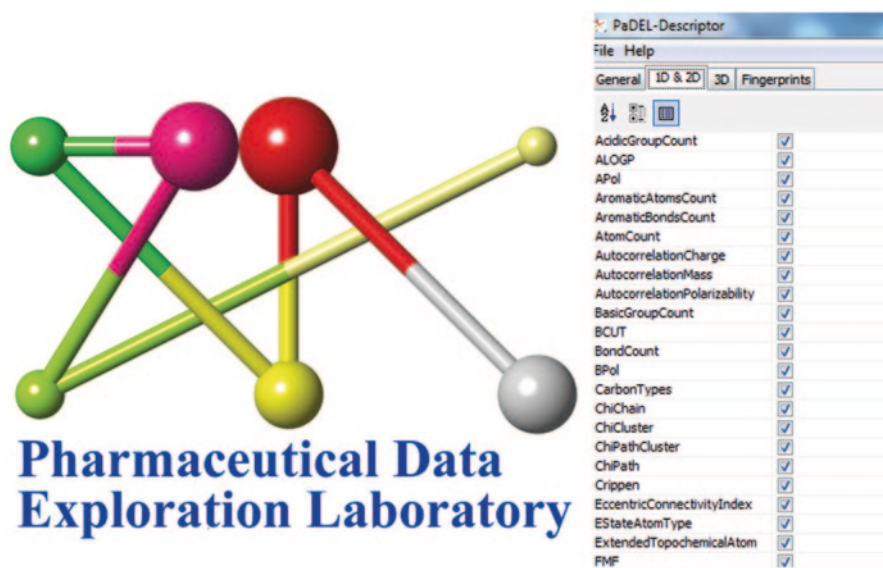


Fig. 2.3 PaDEL descriptor calculator graphical user interface (GUI)

## PaDEL

It is an open-source program that computes 797 descriptors and 10 types of fingerprints [34]. It uses the CDK library for computing descriptors; however, some new descriptors have been added, mainly electrotopological state descriptors [35]. Both graphical user interface (GUI) and command line options are available. The advantage of this software is the large number of file formats it supports, around 90 in number. Further, it surpasses the CDK calculator with regard to its speed due to its multithreaded nature (Fig. 2.3).

### 2.1.1.2 Free Programs

PowerMV is a descriptor generation and compound annotation tool designed biologists and statisticians for quickly screening their assay results and gaining some knowledge regarding their potential biological mechanism [36]. Four descriptor sets are used, four bit string and two continuous, which are used for nearest neighbour searching in annotated databases. The package is written in Visual C and C++ and runs on a .NET framework unlike previous Java-based programs. The program provides users with two versions: basic and affiliate with greater graphics and better descriptors in the latter [37]. One can build classification and regression models through graphical interface to the R program.

### 2.1.1.3 Tools Requiring An Academic License

Calculator plug-in in Marvin Beans from ChemAxon is used for calculating a number of descriptors and is available via an academic request [38]. It can be accessed from Marvin Sketch and Marvin view modules. For efficiency, it is advisable to run it using *cxcalc* command in batch mode from command prompt. A number of diverse descriptors can be computed in a short time.

#### A Practice Tutorial

Here, we compute some selected properties for a .smi file containing 100 molecules belonging to the well-known Ames data set [39]. Download this file and put in the Marvin Beans directory. We begin by calculating simple but powerful atomic descriptors like atom counts and atomic composition. The *cxcalc* commands are available in the original directory where ChemAxon is installed and then go to the sub-directory Marvin Beans docs users *cxcalc-calculations.html*. First, navigate to the directory containing Marvin Beans bin folder in command prompt and type *cxcalc -h* to list the commands. Then, type the commands *cxcalc atomcount -z 7 Ames100.smi* and then *cxcalc composition -S true Ames100.smi* to compute the atom counts and atomic composition for all the 100 molecules in the data set. Similarly, type *cxcalc atomicpolarizability test Ames100.smi* to calculate the polarizability of each atom in all 100 molecules (Fig. 2.4).

We can also compute 3D descriptors using the ‘*cxcalc*’ option. Draw a structure of aspirin molecule (acetyl salicylic acid) in Marvin Sketch and save it as .smi in the Marvin Beans folder. In the command window, type *cxcalc stereoisomers -v true aspirin.mol* to generate the stereoisomer of the molecule. Similarly, the command *cxcalc lowestenergyconformer -f mrv test aspirin.mol* calculates the lowest energy conformer of aspirin (Fig. 2.5).

Molecular graph-based descriptors can also be calculated using the *cxcalc* command. Here, let us compute Randic index [40] and Wiener index [41] which are important molecular connectivity descriptors. Randic index, also called bond index, is the sum of bond contributions in a molecule and Wiener path is a topologic index describing the shortest path between all pairs of vertices. The syntax of the commands is *cxcalc randicindex test ames100.smi* and *cxcalc wienerindex test ames100.smi* (Fig. 2.6).

Data processed in one program can be piped into another using the | vertical line command. Let us compute the logP values for 100 molecules in Ames data set and then pipe the output data to Marvin view to view the table alongside. The command to do so is *cxcalc -S -t myLOGP logP -a 0.15 -k 0.05 test ames100.smi | mview—* (Fig. 2.7).

Log p is the water/octanol partition coefficient [42]; there is another descriptor called logD [43] which is a distribution coefficient especially useful for determining lipophilicity of ionizable compounds as it accounts for pH dependence of molecules in aqueous solution.



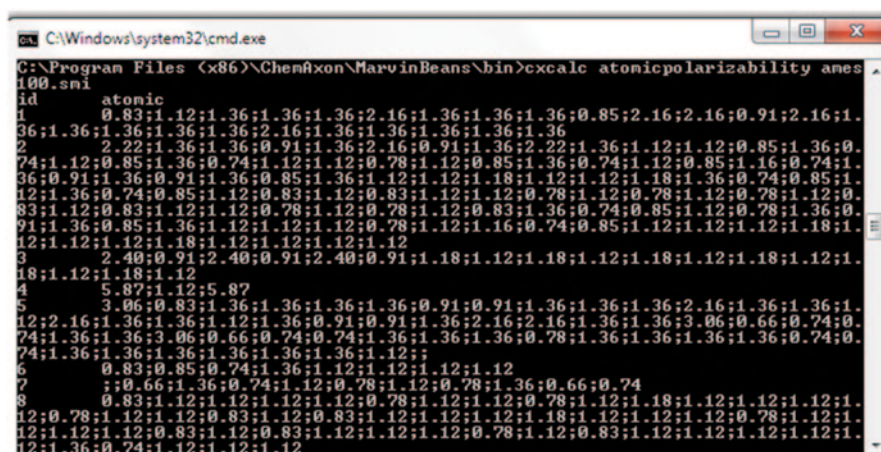


Fig. 2.4 Atomic polarizabilities for 100 molecules using the cxcalc program in Marvin Beans

## Code for Reading a Molecule from a Structure Data File and Printing LogD Values in a Given pH Range

```
plugin.setMolecule(mol);
plugin.run();
//get and print logD values
double[] pHs = plugin.getpHs();
double[] logDs=plugin.getLogDs();
for( int i=0; i<logDs.length; i++) {
double pH =pHs[i];
double logD = logDs[i];
System.out.println(pH+" "+logD);
}
```

### 2.1.1.4 Commercial Software to Calculate Molecular Properties

#### OpenEye

OpenEye company provides software to the pharmaceutical industry for molecular modelling and chemoinformatics. Their Shape TK module facilitates the calculation of molecular descriptors for shape volume overlap between molecules and spatial similarity of chemical groups [44].

#### Schrodinger

The QikProp module computes pharmaceutically relevant descriptors for a large data set containing million compounds in an hour in batch mode [45]. It is a quick,



The figure consists of two screenshots of a Windows command prompt window. The top screenshot shows the execution of the `cxcalc leconformer -f mrv aspirin.mol` command, which outputs a list of bond definitions for a molecule, including atom references and bond orders. The bottom screenshot shows the execution of the `cxcalc stereoisomers -v true aspirin.mol` command, which outputs a list of stereoisomers, including their IDs and various numerical values.

```

C:\Windows\system32\cmd.exe

<bond atomRefs2="a2 a3" order="2" />
<bond atomRefs2="a3 a4" order="1" />
<bond atomRefs2="a4 a5" order="2" />
<bond atomRefs2="a5 a6" order="1" />
<bond atomRefs2="a1 a7" order="1" />
<bond atomRefs2="a6 a8" order="1" />
<bond atomRefs2="a7 a9" order="1" />
<bond atomRefs2="a7 a10" order="2" />
<bond atomRefs2="a8 a11" order="1" />
<bond atomRefs2="a11 a12" order="1" />
<bond atomRefs2="a11 a13" order="2" />
<bond atomRefs2="a2 a14" order="1" />
<bond atomRefs2="a3 a15" order="1" />
<bond atomRefs2="a4 a16" order="1" />
<bond atomRefs2="a5 a17" order="1" />
<bond atomRefs2="a9 a18" order="1" />
<bond atomRefs2="a12 a19" order="1" />
<bond atomRefs2="a12 a20" order="1" />
<bond atomRefs2="a12 a21" order="1" />
</bondArray>
</molecule>
</MChemicalStruct>
</MDocument>
</cml>
C:\Program Files (x86)\ChemAxon\MarvinBeans\bin>cxcalc leconformer -f mrv aspirin.mol

C:\Windows\system32\cmd.exe

C:\Program Files (x86)\ChemAxon\MarvinBeans\bin>cxcalc stereoisomers -v true aspirin.mol

Mr00541 07151315562D

13 13 0 0 0 0 999 U2000
0.7145 1.2375 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1.4289 0.8250 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1.4289 -0.0000 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0.7145 -0.4125 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0.0000 0.0000 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0.0000 0.8250 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0.7145 2.0625 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
-0.7145 1.2375 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1.4289 2.4750 0.0000 O 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0.0000 2.4750 0.0000 O 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
-1.4289 0.8250 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
-1.4289 0.0000 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
-2.1434 1.2375 0.0000 O 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 2 1 0 0 0 0
1 6 2 0 0 0 0
2 3 2 0 0 0 0
3 4 1 0 0 0 0
4 5 2 0 0 0 0
5 6 1 0 0 0 0

```

Fig. 2.5 Computed stereoisomer and lowest energy conformer of aspirin using cxcalc command

The figure consists of two screenshots of a Windows command prompt window. The top screenshot shows the execution of the `cxcalc randicindex anes100.smi` command, which outputs a list of Randic index values for a dataset. The bottom screenshot shows the execution of the `cxcalc wienerindex anes100.smi` command, which outputs a list of Wiener index values for the same dataset.

```

C:\Windows\system32\cmd.exe

C:\Program Files (x86)\ChemAxon\MarvinBeans\bin>cxcalc randicindex anes100.smi
id Randic index
1 12.35
2 48.49
3 8.49
4 1.41
5 25.39
6 3.80

C:\Windows\system32\cmd.exe

C:\Program Files (x86)\ChemAxon\MarvinBeans\bin>cxcalc wienerindex anes100.smi
id Wiener index
1 1493
2 79045
3 525
4 4

```

Fig. 2.6 Randic and Wiener index values computed for the data set

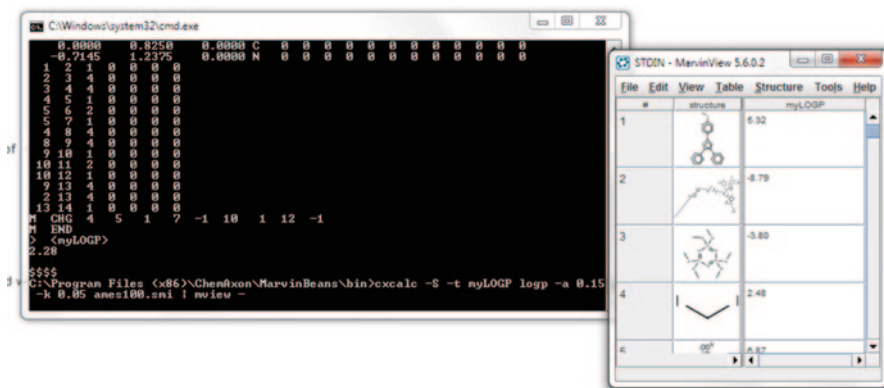


Fig. 2.7 LogP data for 100 molecules piped to Marvin View to visualize the tabulated results

accurate, easy-to-use absorption, distribution, metabolism and excretion (ADME) prediction program designed by Professor William L. Jorgensen [46]. It provides ranges for comparing a particular molecule's properties with those of 95% of known drugs. It can flag 30 types of reactive functional groups that may cause false positives in high-throughput screening (HTS) assays. QikProp input must be a file containing the 3D structure ( $x$ ,  $y$ , and  $z$  coordinates and atomic numbers) of one or more molecules.

### *A Practice Tutorial*

Let us compute the ADME properties of the previous Ames100 data set. First, download the data set from [www.chemoinformatics.org](http://www.chemoinformatics.org). It contains 100 molecules with the binary mutagenicity classification data. We will compute QikProp descriptors for them. Before submitting to QikProp, it is advisable to prepare the molecules using the LigPrep module in Schrodinger. LigPrep automatically converts them to 3D structures; also check for correct tautomeric and ionization variations. It performs energy minimization to generate a customized ligand library [47]. The .mae output file from LigPrep is input into the QikProp module by clicking applications and submitting the job (Figs. 2.8 and 2.9).

The output from the QikProp is obtained in four files, viz. qikpropames100.out, qikpropames100.mae, qikpropames100.qpsa and qikpropames100.csv. Apart from the usual physico-chemical properties, the comma-separated values (CSV) file shows the important descriptors like caco-2 and MDCK cell permeability, blood-brain barrier (logBB), HERG, CNS which are important ADME predicted parameters for a molecule to qualify as drug (Fig. 2.10).

Alternatively, a simple python script can be downloaded from the Schrodinger Script Center for generating molecular descriptors like topological, Molecular Orbital PACKAGE (MOPAC) and QuikProp (Script name: molecular\_descriptors.py

Fig. 2.8 LigPrep input screen

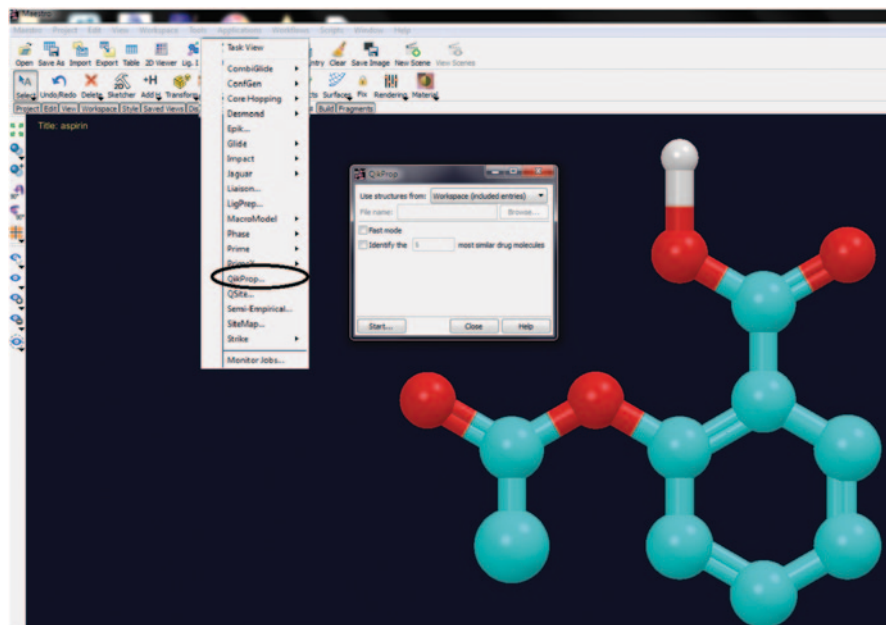
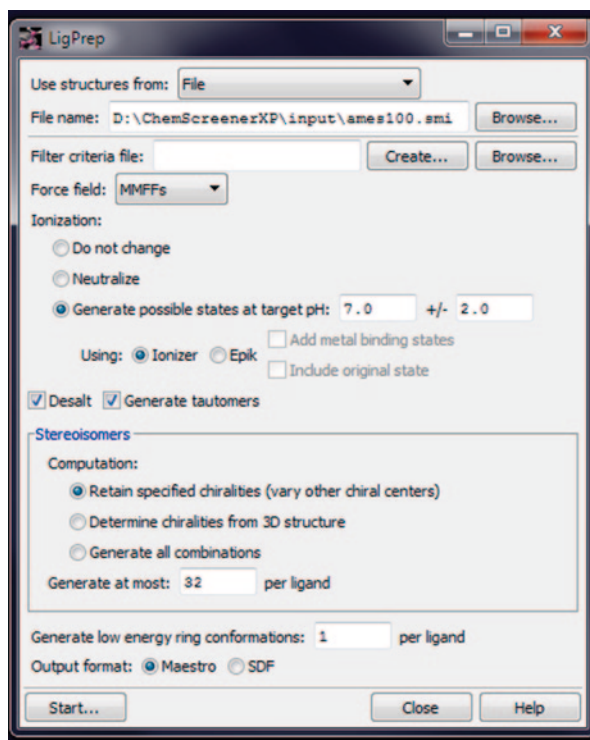


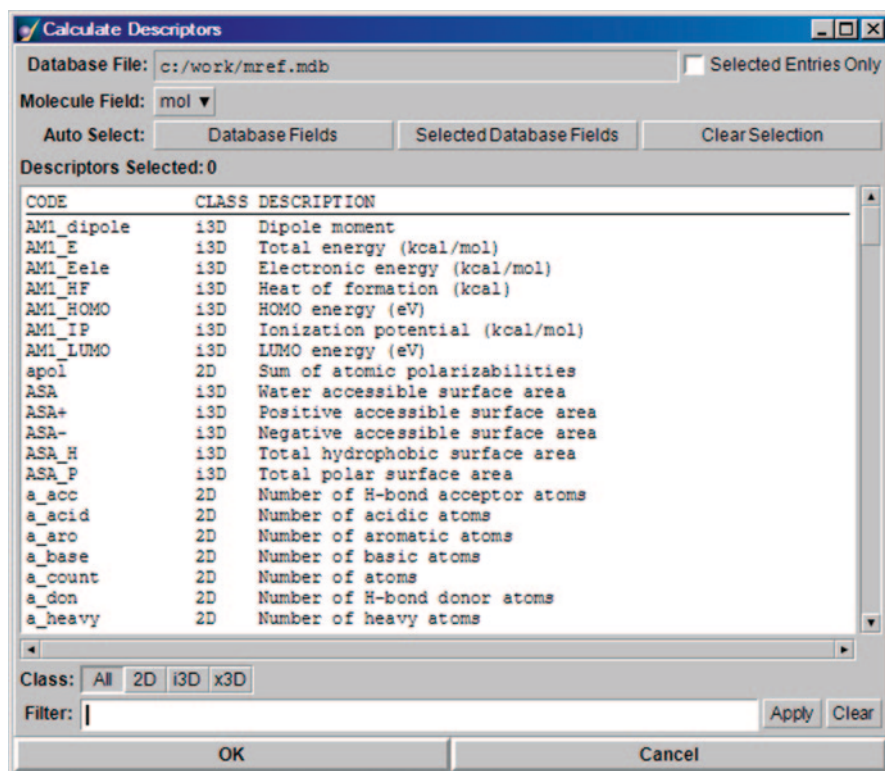
Fig. 2.9 QikProp input screen

**Fig. 2.10** QikProp computed descriptors for the Ames100 data set in comma-separated values (CSV) format

Molecular Operating Environment (MOE) from Chemical Computing Group (CCG) has many chemoinformatics modules; [48] 185 MOE 2D descriptors were calculated for the Ames data set as shown in the screen capture (Figs. 2.11 and 2.12). These descriptors can be input into to build linear regression models.

Dragon 6 is an application for the calculation of 4,885 molecular descriptors [49]. The latest version of Dragon includes new molecular descriptors such as CATS 2D, Klein TDB autocorrelations, atom-type E-state indices, extended topological atom (ETA) descriptors, P\_VSA descriptors, ring descriptors, several indices from different 2D and 3D matrices, drug-like and lead-like filters. These descriptors can be used to evaluate molecular structure–activity or structure–property relationships, as well as for similarity analysis and HTS of molecule databases.

ADME and molecular mechanics descriptors can be calculated using Accelerlys program. Their TOPKAT module is an established *in silico* method for assessing toxicity prediction of organic compounds [50]. TOPKAT can help assess environmental fate, ecotoxicity, toxicity, mutagenicity, and reproductive/developmental toxicity of chemicals. TOPKAT technology is currently used to optimize therapeutic ratios of



**Fig. 2.11** Descriptors list in Molecular Operating Environment (Chemical Computing Group) MOE(CCG)

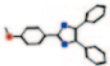
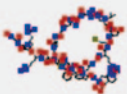
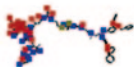

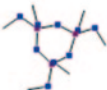
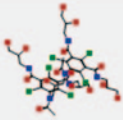

lead compounds, prioritise promising compounds for further development/investment, evaluate intermediates, metabolites and pollutants screen compounds generated via HTS systems, assess pharmaceutical, commercial, industrial and agricultural chemical products for potential safety problems and set dose ranges for animal assays.

### 2.1.1.5 In-House-Developed Open-Source Tool

Large-scale distributed computing of chemical properties has been carried out using ChemStar, wherein the Topological Polar Surface Area (TPSA) property of 18 million compounds was studied using Java Remote Method Invocation (JAVA RMI) [51].

Database Viewer : d:/moe2010/bin-i4w9/amesdataset.mdb

File Edit Display Compute Window Help

	mol	Ames test categorisation	FP:MACCS
1		mutagen	38 62 65 7
2		nonmutagen	14 25 36 4
3		mutagen	23 36 38 4
4		mutagen	23 25 36 3
5		nonmutagen	29 43 53 6
6		nonmutagen	27 53 54 7
7		nonmutagen	42 106 107

**Fig. 2.12** Molecular Design Limited Molecular ACCESS System (*MACCS*) fingerprints computed for Ames data set



Code for Distributed Computing Of Molecular Properties Using ChemStar<sup>1</sup>

```
Class:
Read Input file(String fname){

Distribute the tasks to Clients

Client Components (Parallel mode)

-Get List of Calculator Plugins (ChemAxon / PADEL / CDK / JOELib)
-LogP
-TPSA
-MWT
-HBA
-HBD
-WeinerPath
-Volume
-ADMET
-Toxicophores
-Chemophores
-Pharmacophores
-MACCS Keys
-nAtoms (C, H, N,S,O,Cl,Br,I,N,P)

Send the results to Server

}

class AppendFileStream extends OutputStream
{
    public AppendFileStream(String s)
        throws IOException
    {
        fd = new RandomAccessFile(s, "rw");
        fd.seek(fd.length());
    }

    public void close()
        throws IOException
    {
        fd.close();
    }

    public void write(byte abyte0[])
        throws IOException
    {
        fd.write(abyte0);
    }

    public void write(byte abyte0[], int i, int j)
        throws IOException
    {
        fd.write(abyte0, i, j);
    }

    public void write(int i)
        throws IOException
    {
        fd.write(i);
    }

    RandomAccessFile fd;
}
```

<sup>1</sup> Interested readers are encouraged to download the supporting materials related to ChemStar application (JCIM' 2008, ACS).



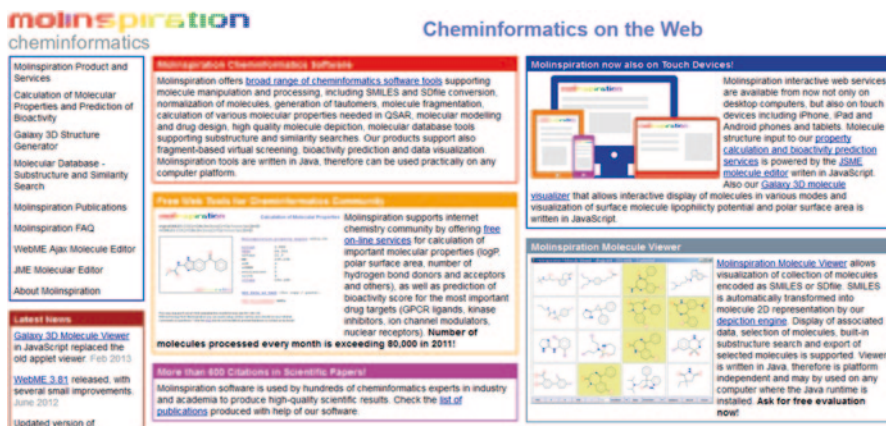


Fig. 2.13 Home page of molinspiration server on the web

## 2.1.2 Online Property Prediction Tools

All of them mostly employ any of the machine learning-based quantity structure–activity relationship (QSAR)/QSPR methods for property prediction.

### 2.1.2.1 Molinspiration

This site provides a range of tools for structure drawing, property prediction, etc. [52], Fig. 2.13.

### 2.1.2.2 Prediction of Activity Spectra for Substances

The acronym PASS stands for prediction of activity spectra for substances [53]. Upon entering a structural formula of a chemical substance, the program computes the potential biological activities of this compound. To execute the prediction, PASS requires a knowledge base about structure–activity relationships (SAR) for compounds with known biological activities. This is provided by SAR Base, containing the analysis results obtained with an in-house training set of more than 250,000 compounds with known biological activities. This training set is continuously curated and expanded. SAR Base can also be replaced by an exclusive knowledge base, which can be created using in-house data. The knowledge base together with the user-defined constraints of biological activities of interest and relevant parameters provides PASS the starting point for the computational prediction (Fig. 2.14).

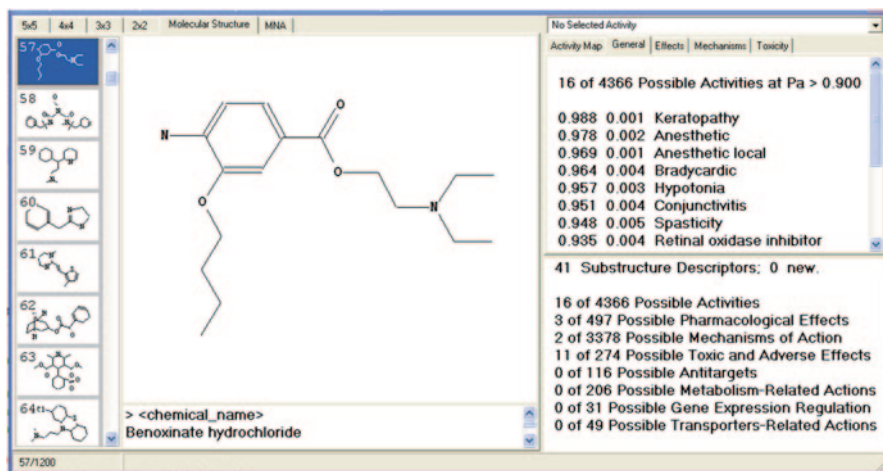


Fig. 2.14 Prediction of activity spectra for substances (PASS) property prediction server

### 2.1.2.3 AquaSol

A web-based predictor, AquaSol, is available online through the ChemDB portal that can be applied to the problem of predicting aqueous solubility [54]. Molpro, another module in the portal, predicts molecular properties other than 3D structures.

### 2.1.2.4 Molecular descriptor family prediction SAR

An algorithm for extracting useful information from the topological and geometrical representation of chemical compounds was developed and integrated to calculate molecular descriptor family (MDF) members [55]. The activity is predicted based on a learning set, a previously obtained MDF SAR model and a molecule submitted as HIN file by the user.

### 2.1.2.5 preADMET

It is a commercial website used to compute 2,000 descriptors including absorption, distribution, metabolism, elimination and toxicity (ADMET)-relevant properties like caco-2 cell permeability, blood–brain barrier, human intestinal absorption, etc. [56]. It also comes with a drawing tool and library builder.

**Fig. 2.15** Structure browser of Distributed Structure-Searchable Toxicity (*DSSTox*)

#### 2.1.2.6 Distributed Structure-Searchable Toxicity Prediction Server

It is hosted by Environmental Protection Agency (EPA) USA to predict the toxicity of compounds [57]. It encourages and uses the structure data file (sdf) format. It has a browser developed from open-source tools to search its data files. The files can be downloaded into any chemical relational database for chemical analog searching to enable model building (Fig. 2.15).

#### 2.1.2.7 Estimation Programs Interface Suite

The Estimation Programs Interface (EPI) suite is a free package to compute descriptors specifically to predict the biodegradability of compounds [58], Fig. 2.16.

The EPI Suite developed by EPA is a physical/chemical property and environmental fate estimation program. EPI Suite uses a set of several estimation programs like KOWWIN, AOPWIN, HENRYWIN, MPBPWIN, BIOWIN, BioHCwin, KOCWIN, WSKOWWIN, WATERNT, BCFBAF, HYDROWIN, KOAWIN and AEROWIN, WVOLWIN, STPWIN, LEV3EPI and ECOSAR. Every module in this program and similar programs has its own level of approximation and accuracy.

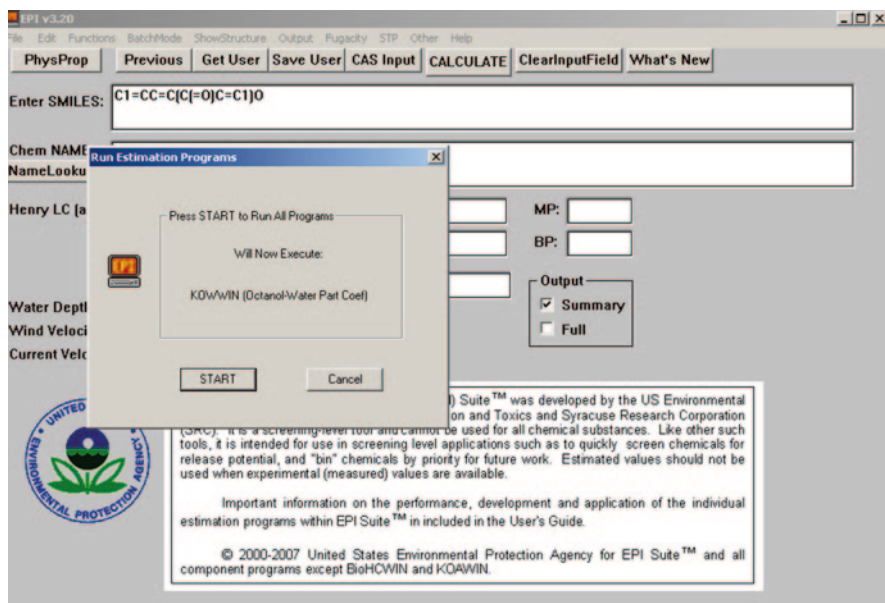


Fig. 2.16 EPI user interface

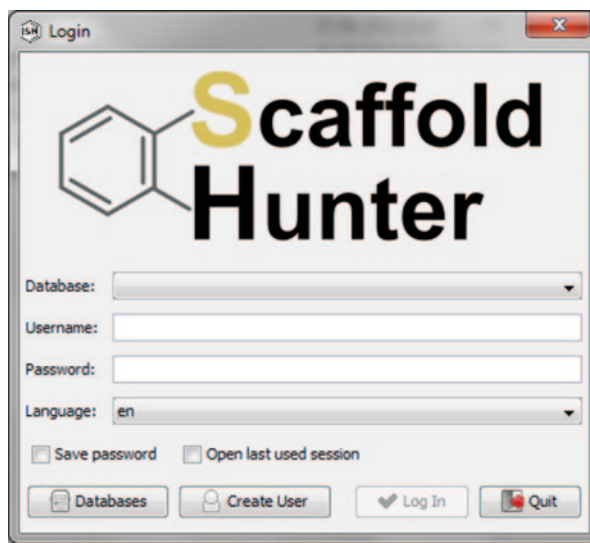
### 2.1.3 Virtual Library Generation (Enumeration)

The concept of designing virtual libraries to enhance the diversity of compounds for efficient lead generation is well known [59]. A virtual library is composed of scaffold, linkers and functional groups. First, let us see what is a scaffold and what are the known scaffold generation tools.

#### 2.1.3.1 Scaffold

The term ‘scaffold’ is used broadly in chemistry; the precise meaning of the word is context- and chemist-dependent. Bemis and Murcko outlined a popular method for computationally deriving scaffolds from molecules by removing side-chain atoms [60]. Atoms in ring systems or linking ring systems, and  $sp^2$  atoms directly bonded to these atoms, were preserved. Alternative scaffold definitions rely on abstraction or decomposing the framework into simpler substructural elements. For example, a molecular framework can be interpreted as a graph containing nodes and edges representing atom and bond types, respectively. Removing atom and bond labels or agglomerating nodes by chemotype yields a hierarchy of reduced graphs, or molecular equivalence classes, that represent sets of related molecules. Likewise, a framework can be further decomposed into individual rings (or the core ring assembly) using chemically intuitive rules; the rings can individually or jointly be considered as scaffolds derived from the original compound.

**Fig. 2.17** Scaffold Hunter start-up screen



Scaffolds are generally obtained by removing side-chain atoms from molecules with the definition of both ‘side chains’ and the equivalence of atoms and rings within the scaffolds being dependent on the particular implementation of the algorithm. Scaffolds constitute the major ‘denominator’ in drug design, as evident from approaches such as ‘scaffold hopping’, their link to bioactivity patterns and the fact that scaffold enumerations (Markush structures) are routinely used for patenting chemical series in the pharmaceutical context. From this, it becomes apparent that the scaffold is a truly relevant entity in synthetic organic as well as medicinal chemistry.

## Open-Source Tools for Scaffold Generation

### *Scaffold Hunter*

Scaffold Hunter is a Java-based open-source tool for the visual analysis of data sets with a focus on data from the life sciences, aiming at an intuitive access to large and complex data sets [61]. The tool offers a variety of views, e.g. graph, dendrogram and plot view, as well as analysis methods, e.g. for clustering and classification. Scaffold Hunter has its origin in drug discovery, which is still one of the main application areas and is evolved into a reusable open-source platform for a wider range of applications. The tool offers flexible plug-in and data integration mechanisms to allow adaption to new fields and data sets, e.g. from medical image retrieval. Scaffold Hunter is used worldwide in research, both academic and commercial, (Fig. 2.17).

*OpenEye*

**BROOD** is a software application designed to help project teams in drug discovery explore chemical and property space around their hit or lead molecule [62]. BROOD generates analogs of the lead by replacing selected fragments in the molecule with fragments that have similar shape and electrostatics, yet with selectively modified molecular properties. BROOD fragment searching has multiple applications, including lead hopping, side-chain enumeration, patent breaking, fragment merging, property manipulation and patent protection by SAR expansion.

*Code Optimized for Scaffold Generation Using Free And Open-Source Tools*

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package cheminfbook;

import chemaxon.struc.Molecule;
import chemaxon.util.MolHandler;
import chemaxon.sss.search.MolSearch;
import chemaxon.formats.MolImporter;
import java.util.Vector;
import java.io.*;
import joelib.io.*;
import joelib.smiles.*;
import joelib.molecule.JOEMol;
import joelib.molecule.JOEAtom;
import joelib.util.iterator.AtomIterator;
import joelib.molecule.JOEBond;
import joelib.util.iterator.BondIterator;
import java.util.*;
import chemaxon.util.MolHandler;
import chemaxon.struc.Molecule;

/**
 *
 * @author M Karthikeyan and Renu Vyas
 */
public class Cheminfbook {

    /**
     * @param args the command line arguments
     */
    Cheminfbook() {
    }

    public static void main(String[] args) {
        // TODO code application logic here
        Cheminfbook cb = new Cheminfbook();
        try {
            String smi = "C1C(Br)C(OC)CC(C1)C1C2=C(C=C)C=CC=C2C3N(C)C3";
            Molecule m = MolImporter.importMol(smi);
            // m.clean(3, null);
            // System.out.println(m.toFormat("
            String[] out = cb.getScaffold(smi, true, true);
            System.out.println(smi + ">>" + out[0]);
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

```

    }

    public static JOEMol ReadSMILES(String smiles, IOType inType, IOType outType) {
        JOEMol mol = new JOEMol(inType, outType);
        try {
            JOESmilesParser.smiToMol(mol, smiles, ".");
        } catch (Exception e) {
            System.out.println(e);
        }
        mol.addHydrogens();

        return mol;
    }

    /** Module to generate scaffold from SMILES format */
    public static String[] getScaffold(String smiles, boolean
removeAtomAndBondTypes, boolean c_atom) {

        String[] output = new String[5];
        output[0] = "";
        output[3] = "";
        int i = 0;
        JOEMol mol = ReadSMILES(smiles,
IOTypeHolder.instance().getIOType("SMILES"),
IOTypeHolder.instance().getIOType("SDF"));
        JOEMol framework = (JOEMol) mol.clone();
        JOEMol RGp = new JOEMol();
        int max = 100;
        String[] del_bond = new String[max];
        framework.deleteHydrogens();
        JOEAtom atom;
        JOEBond bond;
        int a_cnt = framework.numAtoms();
        int b_cnt = framework.numBonds();
        String[] at_inf = new String[a_cnt];
        int db_cnt = 0;
        int at_cnt = 0;
        int[][] da_inf = new int[b_cnt][5];
        String[][] db_inf = new String[b_cnt][4];
        for (int z = 0; z < b_cnt; z++) { //b_cnt
            bond = mol.getBond(z);
            da_inf[z][0] = bond.getBeginAtomIdx();
            da_inf[z][1] = bond.getEndAtomIdx();
            da_inf[z][2] = bond.getBondOrder();
            db_inf[z][0] = bond.getBeginAtom().toString();
            db_inf[z][1] = bond.getEndAtom().toString();
        }
        AtomIterator ait;
        JOEAtom h_atom = new JOEAtom();
        h_atom.setAtomicNum(1);
        boolean atomDeleted;
        String s = "";
        int d = 0;
        do {
            atomDeleted = false;
            ait = framework.atomIterator();
            while (ait.hasNext()) {
                StringBuffer sb = new StringBuffer();
                atom = ait.nextAtom();
                boolean m = atom.isInRing();
                atom.getCIdx();
                Vector vectBonds = atom.getBonds();
                if (m) {
                    JOEBond r_bond = (JOEBond) vectBonds.firstElement();
                    JOEAtom ra1 = r_bond.getBeginAtom();
                    JOEAtom ra2 = r_bond.getEndAtom();
                } else {
                    JOEBond nr_bond = (JOEBond) vectBonds.firstElement();

```



```

        JOEAtom ra1 = nr_bond.getBeginAtom();
        JOEAtom ra2 =
    }

    if (vectBonds.size() == 1 && d == 0) {
        bond = (JOEBond) vectBonds.firstElement();
        if (!removeAtomAndBondTypes && atom.isOxygen() &&
bond.isCarbonyl())) {
            atomDeleted = true;
            JOEAtom a1 = bond.getBeginAtom();
            JOEAtom a2 = bond.getEndAtom();
            int t1 = a1.getIdx();
            int t2 = a2.getIdx();
            int c1 = a1.getCIdx();
            int c2 = a2.getCIdx();
            if (a2.isInRing()) {
                da_inf[i][3] = t2;
                da_inf[i][4] = t1;
                db_inf[i][2] = a2.toString();
                db_inf[i][3] = a1.toString();
                at_inf[at_cnt] = a1.getType() + "_" + a2.getType();
                joelib.util.types.IntInt a = new
joelib.util.types.IntInt();
                a.i1 = a1.getIdx();
                a.i2 = a2.getIdx();
                RGp.beginModify();
                a1.setFormalCharge(0);
                RGp.addAtom(a1);
                a2.setFormalCharge(0);
                RGp.addAtom(a2);
                RGp.addBond(bond);
                RGp.endModify();
                d++;
                System.out.println("a2 " + framework + " d " + d);
            } else {
                da_inf[i][3] = t2;
                da_inf[i][4] = t1;
                db_inf[i][2] = a2.toString();
                db_inf[i][3] = a1.toString();
                at_inf[at_cnt] = a1.getType() + "_" + a2.getType();
                joelib.util.types.IntInt a = new
joelib.util.types.IntInt();
                a.i1 = a1.getIdx();
                a.i2 = a2.getIdx();
                RGp.beginModify();
                a1.setFormalCharge(0);
                RGp.addAtom(a1);
                a2.setFormalCharge
                RGp.addAtom(a2);
                RGp.addBond(bond);
                RGp.endModify();
                framework.deleteBond(bond);
                framework.deleteAtom(atom);
            }
            i++;
        }
    }
}
} while (atomDeleted && i < max);
JOEMol e_scaff = (JOEMol) framework.clone();
output[0] = e_scaff.toString(IOTyperHolder.instance().getIOTyper("SMILES"));
if (removeAtomAndBondTypes) {
    BondIterator bit = framework.bondIterator();
    JOEAtom atom1;
    JOEAtom atom2;
    int index;
    while (bit.hasNext()) {
        bond = bit.nextBond();

```

```

        atom1 = bond.getBeginAtom();
        atom2 = bond.getEndAtom();
        index = bond.getIdx();
        bond.set(index, atom1, atom2, 1, 0);
    }
    ait = framework.atomIterator();
    while (ait.hasNext()) {
        atom = ait.nextAtom();
        atom.setFormalCharge(0);
        atom.unsetStereo();
        if (!atom.isCarbon() && c_atom) {
            atom.setAtomicNum(6);
            boolean m = atom.isInRing();
            atom.getIdx();
        } else if (!atom.isCarbon() && !c_atom) {
            JOEMol f_scaff = (JOEMol) framework.clone();
        }
    }
    if (c_atom) {
        output[1] =
framework.toString(IOTypeHolder.instance().getIOType("SMILES"));
    } else if (!c_atom) {
        output[1] =
framework.toString(IOTypeHolder.instance().getIOType("SMILES"));
    }
    }
    framework.stripSalts();
    mol.deleteHydrogens();
    output[2] = "";
    for (int l = 1; l < RGP.numAtoms() + 1; l += 2) {
        int t1 = (l - 1) / 2;
        output[2] += db_inf[t1][2] + "," + db_inf[t1][3] + "," + da_inf[t1][3]
+ "," + da_inf[t1][4] + "\n";
    }
    output[3] = "";
    for (int l = 0; l < mol.numBonds(); l++) {
        output[3] += da_inf[l][0] + "," + da_inf[l][1] + "," + db_inf[l][0] +
", " + db_inf[l][1] + "," + da_inf[l][2] + "\n";
    }
    output[4] = (String)
RGP.toString(IOTypeHolder.instance().getIOType("SMILES"));
    return output;
}
}

```

The above code was used to extract scaffold B from molecule A.



## Commercial Tools

### ReCore

ReCore replaces a given core: Given a predefined central unit of a molecule (the core), fragments are searched in a 3D database for the best-possible replacement—

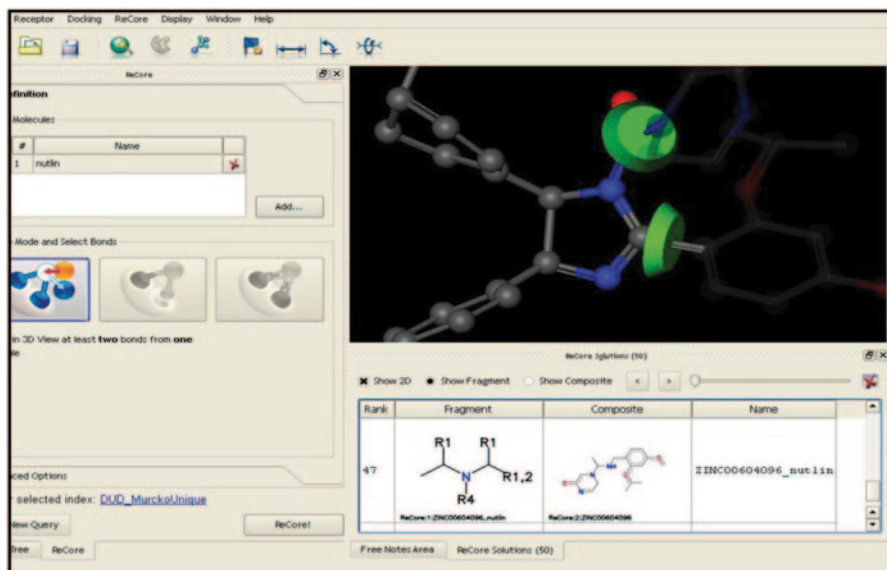


Fig. 2.18 Cutting points in a molecule defined using ReCore

while keeping all connected residues, i.e. the rest of the query compound in place [63]. Additionally, user-defined ‘pharmacophore’ constraints can be employed to restrict solutions. For further details, the reader is encouraged to download the manual from the website address <http://www.biosolveit.de/ReCore/> (Fig. 2.18).

### *Molecular Operating Environment Chemical Computing Group*

Scaffold Replacement (or scaffold hopping) is an approach used to discover new chemical classes by replacing a portion of a known compound (the scaffold), while preserving the remaining chemical groups [64]. This application is built upon MOE’s pharmacophore modelling tools. It generates novel structures from all or part of a ligand (possibly bound to a receptor). Three types of operations are supported:

1. Scaffold Replacement: replace a portion of the ligand with a linker
2. Link Multiple Fragments: connect separate fragments with a linker
3. Add Group to Ligand: extend the ligand with a linker

The user indicates the atoms or bonds to be replaced or extended and can specify QuaSAR Descriptor, Model file and/or pharmacophore query filters to limit the results. For example, a pharmacophore query can be used to enforce specific interactions (or restrictions) on the generated structures when growing in a receptor pocket. Scaffold Replacement can be used as part of a ligand-based or structure-based discovery methodology.

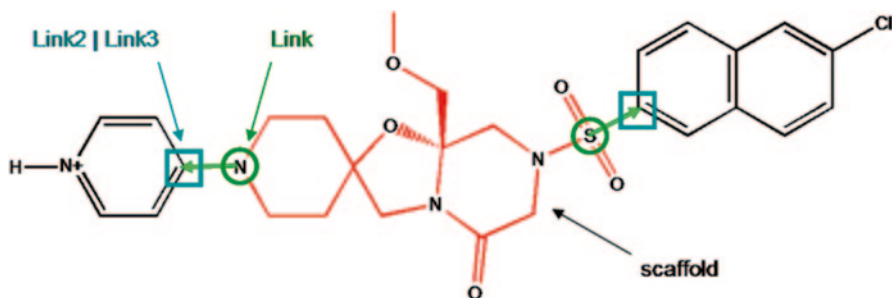


Fig. 2.19 Select red atoms for Replace Scaffold (Select Scaffold)

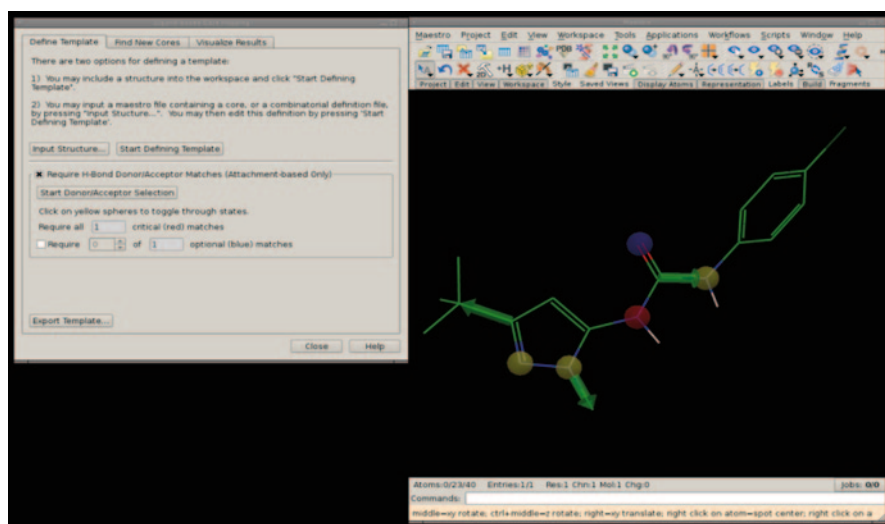


Fig. 2.20 Core hopping module of Schrodinger

Using ‘Replace Scaffold (Select Scaffold)’ and selecting the atoms indicated in red results in two connection points (indicated by arrows). The R-groups are indicated in black (Figs. 2.19 and 2.20).

### Schrodinger

The steps for the two core hopping strategies are given below [65]:

- Start with template with attachment bonds
- ... and with protocore with many possible attachments
- Find ways for protocore to align with template
  - Two alignments are shown in Fig. 2.21
- Add template’s R groups to the new core

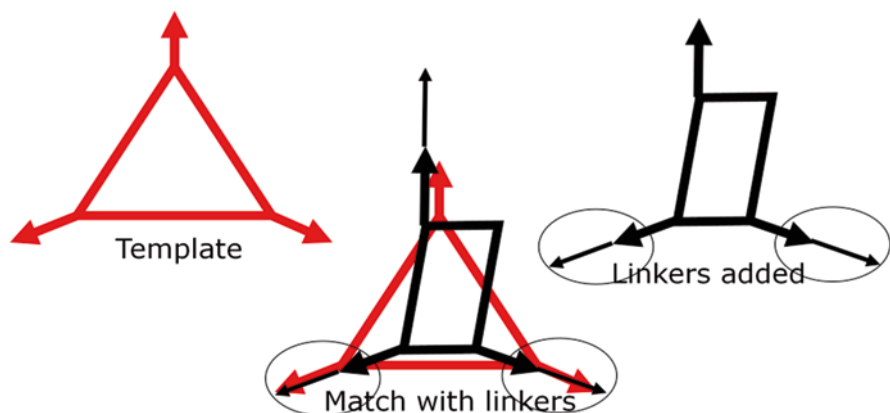
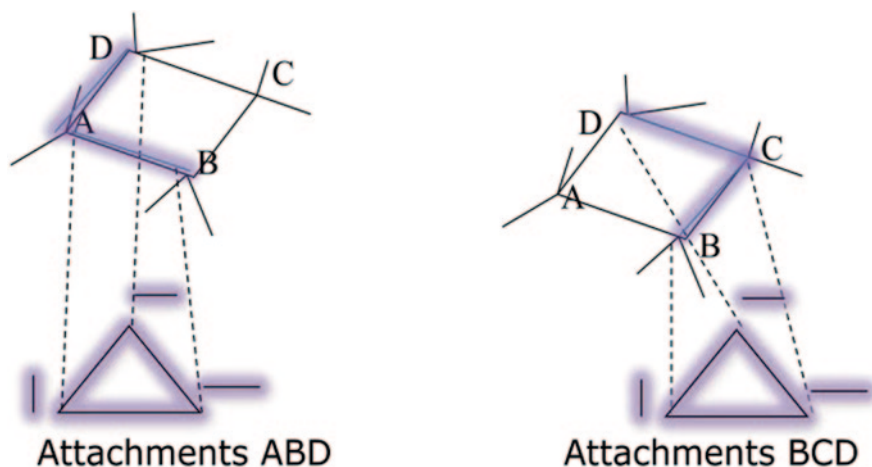


Fig. 2.21 Core hopping methods



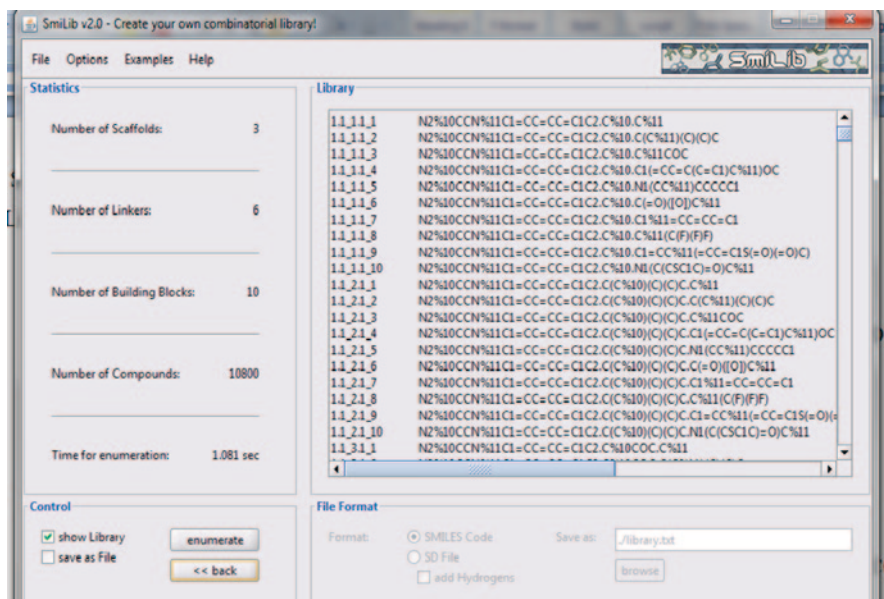
Show 1 linker (maximum) per attachment; default is 2

- All combinations of linkers in all attachment bonds are tested
  - Example shows that two attachment points used linkers
- Suite 2012: a variety of linkers available (2011: methylene)

### 2.1.3.2 Open-Source Tools for Virtual Library Synthesis

#### SmiLib

SmiLib is a Java-based combinatorial library enumeration tool developed by Andreas Schuller [66]. SmiLib v2.0 offers the possibility to construct very large com-



**Fig. 2.22** Graphical user interface (GUI) of SmiLib showing 10,800 molecules created by using three scaffolds, six linkers and ten building blocks

binatorial libraries using the flexible and portable SMILES format. Libraries can be created at rates of approximately 8,700,000 molecules per minute. Combinatorial building blocks are attached to scaffolds by means of linkers to allow for creation of customized libraries using linkers of different sizes and chemical nature. Important features include platform independence, correct handling of stereo chemistry, flexible reaction schemes, improved usability, a unique identifier for each molecule, the option to create libraries in SD format, a conformity check for SmiLib v2.0 SMILES notation restrictions and decreased library enumeration times. SmiLib v2.0 is available in both formats as interactive GUI application and command line tool. The main advantages of SmiLib are its simplicity to use, high flexibility in constructing combinatorial libraries (exact subset of molecules for virtual synthesis can be specified) and high speed of library construction [67]. The SMILES format is used as both input and output format. SmiLib uses a special syntax for ring closures, i.e. any two-digit number preceded by a percentage sign. For example, 'C%10.C%10'  $\equiv$  'C1.C1'  $\equiv$  'CC' (Ethane C<sub>2</sub>H<sub>6</sub>). In addition to normal SMILES, [R1], [R2], [R3], etc. are used as labels for sites of variability and [A] is used as a label for attachment sites. An attachment site is part of the molecule, which is to be attached to a scaffold or a linker. It is a platform-independent program written in Java; SmiLib is run with help of the Java virtual machine with '`java -jar SmiLib.jar`'. It requires three American Standard Code for Information Interchange (ASCII) files containing all scaffold, linker and building block molecule fragments in SMILES format (command line parameters `-s<scaffolds.smi>`, `-l<linkers.smi>`, `-b<buildingblocks.smi>`). A reaction scheme file for the enumeration of a combinatorial library can be specified with the option '`-r<reaction_scheme>`' (Fig. 2.22).

### Molecular Operating Environment Chemical Computing Group

In MOE, a proprietary software is also supplied with a combinatorial library generation tool [68]. A combinatorial library is specified by:

- A database of scaffold molecules or a single scaffold molecule
- Databases of functional groups
- Connection information specifying where the functional groups attach on each scaffold

**Attachment Points** A single combinatorial product is constructed by attaching *R-groups* to a scaffold at marked *attachment points*, called *ports*. The entire combinatorial library is enumerated by exhaustively cycling through all combinations of *R-groups* at every attachment point on every scaffold. The virtual library is written to an output database. Attachment points are terminal atoms named ‘*A<sub>n</sub>*’, where *n* is a positive integer. In the QuaSAR-CombiGen panel, *n* is limited to the range [0 ... 9]. When using the scientific vector language (SVL) command QuaSAR\_CombiGen, however, *n* can be in the range [0 ... 999]. If the terminal atom is attached to the main molecule by a higher-order bond, substitution will be made through a bond of the same order. Note that the bond order at the scaffold attachment point must agree with that at the *R-group* attachment point: Either at least one of the bond orders must be 1 (single bond) or both must be of the same order. Fragment molecules are created by appropriately naming atoms at the desired points of substitution. One can use the Builder to perform this operation and the Clip *R-Groups* application in a database can be used to create fragments with named attachment points.

Attachment points must be specified on both the *R-group* and the scaffold molecule (Fig. 2.23).

## 2.1.4 Virtual Screening

Bio- and chemoinformatics are crucial for the success of virtual screening of compound libraries which is an alternative and complementary approach to HTS in the lead discovery process [69]. A combination of drug-derived building blocks and a restricted set of reaction schemes is the key for the automatic development of novel, synthetically feasible structures that can be docked into the active site of a drug target for lead identification using computers which is the essence of virtual screening [70]. The virtual screening of combinatorial libraries is used to rationally select compounds for biological *in vitro* testing from databases of hundreds of thousands of compounds. In addition to structural descriptors, such as fingerprints and pharmacophores, the application of relatively simple structural descriptors traditionally used in quantitative structure–activity studies offers speed and efficiency for rapidly measuring the molecular diversity of such collections capable of screening large data sets of organic compounds for potential ligands. The methods described in this section are used for computationally prioritising candidate molecular libraries for synthesis and screening by using certain filters. These statistical methods are powerful because they provide a simple way to estimate the properties of the overall



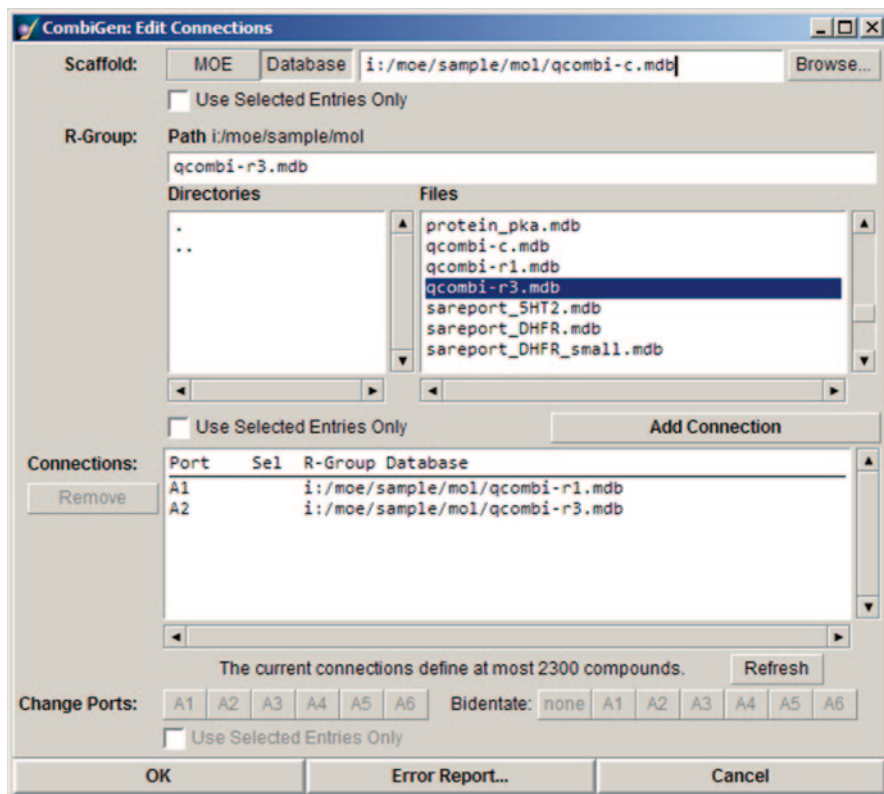


Fig. 2.23 Virtual library synthesis in Molecular Operating Environment (MOE) using CombiGen

library without explicitly enumerating all of the possible products. Current virtual screening applications focus not only on biological activity but also on other relevant properties of drug candidates, like ADME. In the first step of virtual screening, the prediction algorithm must be very fast because typically several millions of compounds have to be processed to generate hit lists of molecules which can be further subjected to actual experimental confirmation in laboratory.

A typical virtual screening workflow in a drug design experiment involves the following steps:

1. Scaffold extraction from a data set of molecules.
2. Use these scaffolds as seeds to enumerate a virtual library by supplying linkers and functional groups.
3. Apply any of the filters below either independently or in combination depending upon prior knowledge (Rule of five (RO5) Lipinski Pharmacophore model QSAR Docking Select Hits or no hits) (Fig. 2.24).

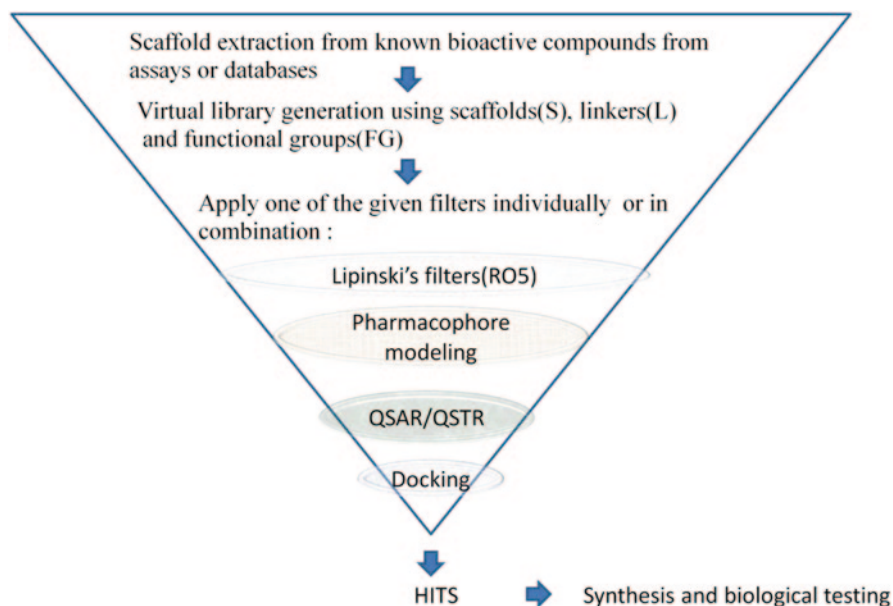


Fig. 2.24 A general virtual screening protocol

### 2.1.4.1 Free Virtual Library Screening Platforms

#### Screening Assistant 2

Screening Assistant 2 (SA2) is a modular software dedicated to perform various simple and advanced chemoinformatics analysis around chemical libraries [69], Fig. 2.25.

SA2 is a free and open-source Java software dedicated to the storage and the analysis of small to very large chemical libraries. SA2 stores unique chemical structures using a MySQL database and associates to the molecules various standard precomputed descriptors as well as user-defined properties/descriptors that can be imported in a flexible way. Various standard and advanced chemoinformatics methods have been implemented, including chemical space visualization/creation, substructure and similarity searches, diverse subset extraction and diversity indices calculation. Its modular architecture, based on the NetBeans Platform, eases the addition of new functionalities to the software. The program and source code are freely available (GPL), The system is programmed in Java and data are managed by a MySQL server. The software allows to calculate drug-like and lead-like properties, as well as to study the libraries in terms of uniqueness, of internal duplicates, diversity and frameworks (<http://www.univ-rleans.fr/icoa/screeningassistant/>). The software is available on Sourceforge: <http://sourceforge.net/projects/screenassistant>.

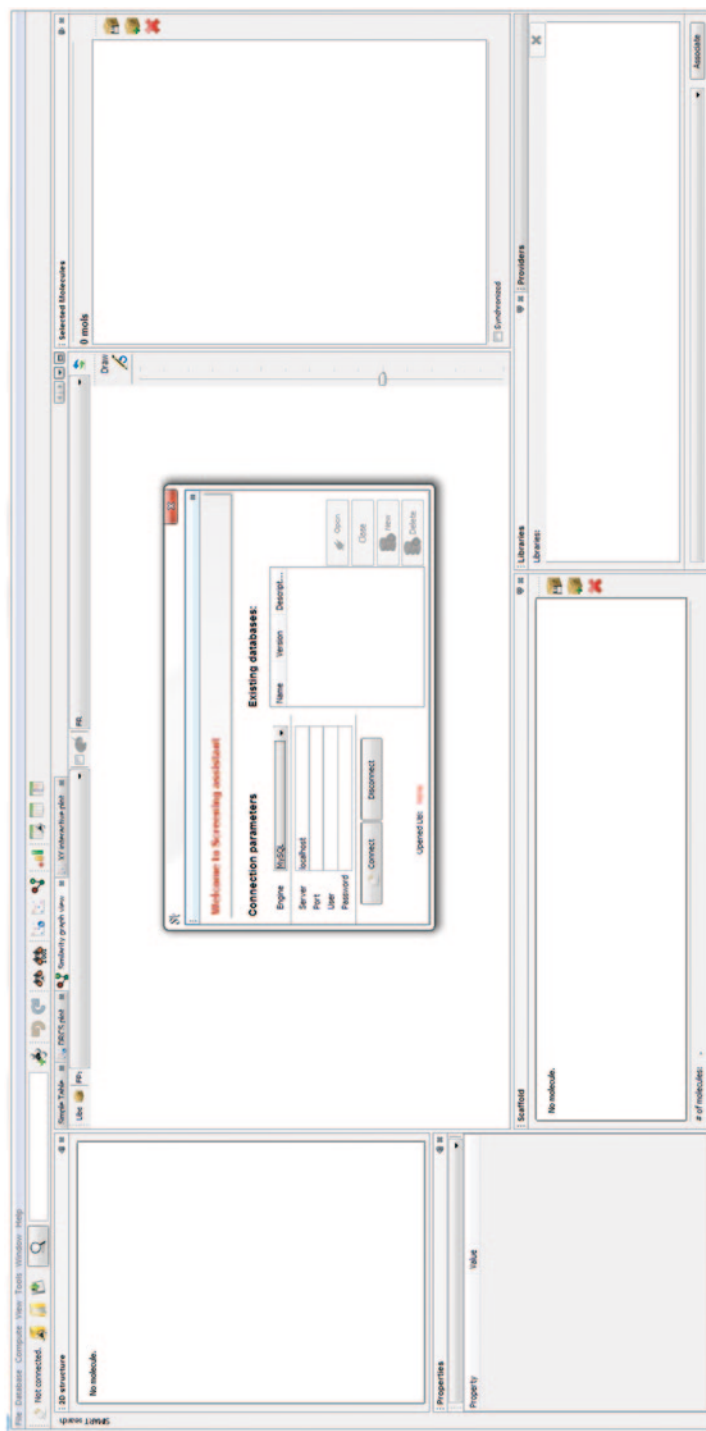


Fig. 2.25 The welcome screen of Screening Assistant 2

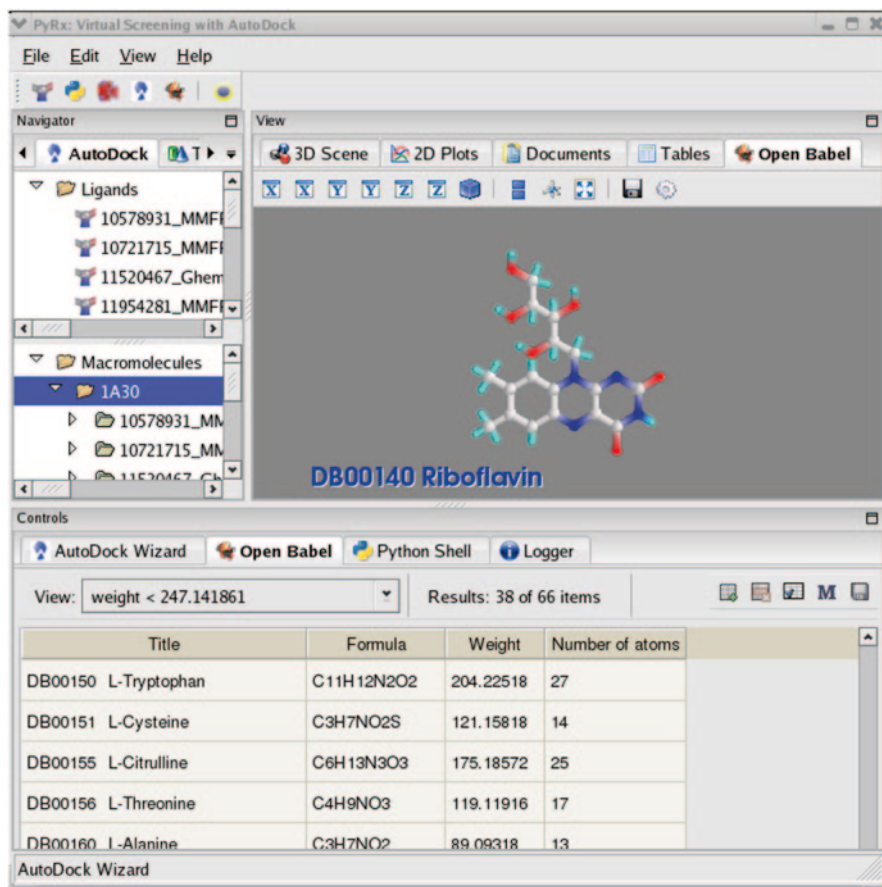


Fig. 2.26 A screenshot of PyRx platform

## PyRx

PyRx is a free and open-source software for computer-aided drug design distributed under Simplified BSD License [70]. PyRx is a Virtual Screening software for Computational Drug Discovery that can be used to screen libraries of compounds against potential drug targets. PyRx enables medicinal chemists to run virtual screening from any platform and helps users in every step of this process—from data preparation to job submission and analysis of the results. PyRx includes a docking wizard with easy-to-use user interface which makes it a valuable tool for computer-aided drug design. PyRx also includes chemical spreadsheet-like functionality and powerful visualization engine that are essential for rational drug design (Fig. 2.26).

A number of open-source software are used such as AutoDock 4 and AutoDock Vina for docking AutoDockTools to generate input files, Python as a programming/



Fig. 2.27 A comparison of Bemis Murcko scaffold and ChemScreener scaffold

scripting language, wxPython for cross-platform GUI, the Visualization ToolKit (VTK) by Kitware Inc, Enthought Tool Suite, including Opal Toolkit for running AutoDock remotely using web services, Open Babel for importing SDF files, removing salts and energy minimization and matplotlib for 2D plotting.

### In-House-Developed Virtual Screening Platform

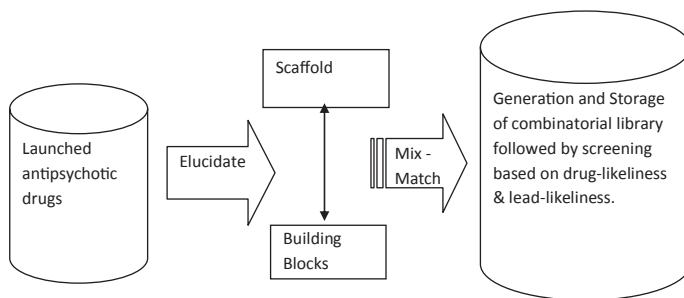
**ChemScreener** It is a Java-based platform developed to create diverse but focussed libraries. Tools like SmiLib do not take into account chemical or physico-chemical characteristics of products but rather simply concatenate scaffold molecules and building blocks with single bonds [71]. Often, the libraries created are huge but not chemically meaningful to develop a lead molecule. ChemScreener provides three main modules to use a scaffold extractor: library generator, a screener which screens the library for the presence of pharmacophoric, chemophoric and toxicophoric features.

The scaffold extractor generates scaffolds, extended scaffolds and frameworks. The extended scaffolds, unlike the conventional Bemis Murcko scaffold [72], retain connection information and are used in focussed library synthesis (Fig. 2.27).

In the medicinal chemistry literature, a number of substructures have been identified as toxicophores, such as some aromatic amines, azides, diazo structures, triazenes, aromatic azo moieties, aromatic hydroxylamines, aliphatic halides, etc. 'Chemophores' refer to substructural groups which are too reactive or inert or synthetically inaccessible, which would lead to practically irrelevant molecules. Medicinal chemists design compounds on the basis of chemophoric features; for instance, the presence of OMe group in a molecule is generally known to enhance its bioactivity, alkyl groups are introduced to increase selectivity, a fluoro group for metabolic stabilization whereas a nitro group will impact the activity in an adverse way which implies later side effects in drug efficacy. Toxicophores were collected from literature databases such as RTECS [73], NIOSHTIC [74], EPA and pharmacophores and chemophores were extracted from literature. This program provides an alert indicating the number of chemophore, toxicophore and pharmacophore matches to assist in fine-tuning the library generated. The virtual library can also be screened on the basis of binding affinity-based filters provided the target structure



A virtual library of 150 million antipsychotic molecules of 2 GB file size was generated from four seed scaffolds using the ChemScreener program which is currently not possible with the existing software. It could also reveal significant bioactivity data patterns from scaffold extraction in big databases like PubChem[75] and ChEMBL [76], (Fig. 2.29).



**Fig. 2.29** Flowchart to create a focussed and diverse library of antipsychotic molecules

### 2.1.5 Thumb Rules for Computing Molecular Properties

- Check the list of molecules for their proper connectivity
- Remove salts, multiple molecules (retain only the large molecule from a mixture of molecule)
- Avoid using too small or too big molecules in the collection (as input for automatic focussed virtual library generation)
- Compute basic descriptors related to Lipinski's rule of five in addition to TPSA, Volume, Weinerpath
- Do 2D and 3D PCA to evaluate diversity and similarity of molecules in the collection
- Do not put too many hydrogen bond acceptor and donor atoms into a molecule, otherwise it will not be absorbed from the intestine to the blood and fail in the preclinical trials
- Apart from the usual Lipinski and Oprea criteria for selection of lead molecules, also search in natural and marine products databases which offer more chemical diversity and unexplored rich functional group variety
- Design molecules with NP scaffolds and functional groups for better bioactivity (based on early reports) and more scope for patenting

### 2.1.6 Do it Yourself

1. Use the relevant code given in the text to extract scaffolds from SMILES of top ten drugs in the market
2. Retrieve ten drug molecules from drug bank database and ten known pesticides, calculate Lipinski's drug-like properties, ADMET and biodegradability parameters using any of the free online tools. Comment on the results
3. Generate a virtual library using SmiLib from molecules belonging to anti-angi-  
nal compounds



### 2.1.7 Questions

1. Write a brief essay on the known property prediction tools in chemoinformatics.
2. How is a virtual library constructed? What are the methods known to screen a virtual library?
3. How do you obtain a diverse but focussed virtual library for a class of therapeutic compounds?
4. Define scaffold hopping. Highlight the tools which can be used for scaffold hopping.
5. What is a scaffold? Elaborate on the known methods of scaffold extraction.

### References

1. Leo A, Hansch C, Church C (1969) Comparison of parameters currently used in the study of structure-activity relationships. *J Med Chem* 12:766–771
2. Admason GW, Bawdon D (1976) An empirical method of structure-activity correlation for polysubstituted cyclic compounds using wiswesser line notation. *J Chem Inf Comput Sci* 16(3):161–165
3. Choplin, F (1990) Computers and the medicinal chemist. In: Hansch C, Sammes PG, Taylor JB (eds) *Comprehensive Medicinal Chemistry* Pergamon Press, UK 4:33–58
4. Tropsha A, Gramatica P, Gombar V (2003) The importance of being earnest: validation is the absolute essential for successful application and interpretation of QSPR models. *Mol Inform* 22(1):69–77
5. <http://www.molecularDescriptors.eu/>
6. Seybold PG, May M, Bagel UA (1987) Molecular structure property relationships. *J Chem Educ* 64(7):575
7. Todeschini R, Consonni V (2009) *Molecular descriptors for chemoinformatics*, vol 2. Wiley-VCH
8. Karelson M (2000) *Molecular descriptors in QSAR/QSPR*. Wiley
9. <http://www.vcclab.org/lab/indexhlp/consdes.html>
10. <http://www.codessa-ro.com/descriptors/electrostatic/index.htm>
11. Balaban AT (1997) *From chemical topology to three dimensional geometry*. Plenum Press, New York, 1–24
12. Karelson M, Lobanov V, Katritzky AR (1996) Quantum chemical descriptors in QSAR/QSPR studies. *Chem Rev* 96:1027–1043
13. Enoch SJ (2010) The use of quantum mechanics derived descriptors in computational toxicology. In: Puzyn T et al (ed) *Challenges and advances in computational chemistry and physics*, vol 8. Springer Science pp 24–27
14. Stanton D (1999) Evaluation and use of BCUT descriptors in QSAR and QSPR studies. *J Chem Inf Comput Sci* 39(1):11–20
15. Ma SL, Joung JY, Lee S, Cho KH, No KT (2012) PXR ligand classification model with SFED weighted WHIM and CoMMA descriptors. *SAR QSAR Environ Res* 23(5–6):485–504
16. <http://rdkit.org/docs/api/rdkit.Chem.MACCSkeys-pysrc.html>
17. Todeschini R, Bettiol C, Giurin G, Gramatica P, Miana P, Argese E (1996) Modeling and prediction by using WHIM descriptors in QSAR studies: submitochondrial particles(SMP) as toxicity biosensors of chlorophenols. *Chemosphere* 33:71–79

18. Hinselmann G, Rosenbaum L, Jahn A, Fechner N, Zell AJ (2011) Compound Mapper: an open source JAVA library and command line tool for chemical fingerprints. *J Chemoinformatics* 3:3
19. Rogers D, Mathew H (2010) Extended connectivity fingerprints. *J Chem Inf Model* 50(5):742–754
20. Bender A, Hamse Y, Mussa HY, Glen C (2010) Similarity searching of chemical databases using atom environment descriptors (Molprint 2D) evaluation of performance. *J Chem Inf Comput Sci* 44:1708–1718
21. Deursen R, Blum Lorenz CB, Reymond JL (2010) A searchable map of PubChem. *J Chem Inf Model* 50(11):1924–1934
22. Chemscreener unpublished results
23. Jorgenson WL, Duffy EM (2002) Prediction of drug solubility from structure. *Adv Drug Deliv Rev* 54:355–366
24. Livingstone DJ, Waterbeemd VD, Han I (2009) In silico prediction of human oral bioavailability. *Method Prin Med Chem* 40:433–451
25. Persson LC, Porter CJ, Charman WN, Bergstrom CA (2013) Computational prediction of drug solubility in lipid based formulation excipients. *Pharm Res* PMID:23771564
26. Faller B, Ertl P (2007) Computational approaches to determine drug solubility. *Adv Drug Deliv Rev* 59:533–545
27. Cortes-Cabrera A, Morris GM, Finn PW, Morreale A, Gago F (2013) Comparison of ultra fast 2D and 3D descriptors for side effect prediction and network analysis in polypharmacology. *Br J Pharmacol*. doi:10.1111/bph.12294
28. Rice BM, Byrd EF (2013) Evaluation of electrostatic descriptors for crystalline density. *Langmuir*
29. Garcia EJ, Pellitero PJ, Jallut C, Pirmgruber GD (2013) Modeling adsorption properties on the basis of microscopic, molecular structural descriptors for non polar adsorbents. *J Chem Inf Model*
30. Wegner JK, Zell A (2003) Prediction of aqueous solubility and partition coefficient optimized by genetic algorithm based descriptors selection method. *J Chem Inf Comput Sci* 43(3):1077–1084
31. Steinbeck C, Hoppe C, Kuhn S, Matteo F, Guha R, Willighagen EL (2006) Recent development of the CDK (Chemistry Development Kit) an open source JAVA library for chemo and bioinformatics. *Curr Pharm Design* 12(17):2111–2120
32. <http://www.rguha.net/code/java/cdkdesc.html>
33. Steinbeck C (2008) Open toolkits and applications for chemoinformatics teaching Abstracts of Papers, 235th ACS National Meeting, New Orleans, LA, United States, April 6–10
34. <http://padel.nus.edu.sg/software/padeldescriptor/>
35. Yap CW (2011) Padel descriptor an open source software to calculate molecular descriptors and fingerprints. *J Comput Chem* 32(7):1466–1474
36. <http://nisl05.niss.org/PowerMV/?q=PowerMV>
37. Liu K, Feng J, Young SS (2005) A software environment for molecular viewing, descriptor generation, data analysis and hit evaluation. *J Chem Inf Model* 45(2):515–522
38. <http://www.chemaxon.com/marvin/help/calculations/calculator-plugins.html>
39. <http://cheminformatics.org/datasets/>
40. Xueliang L, Yongtang S, Wang L (2012) On a relation between randic index and algebraic connectivity. *Match* 68(3):843–839
41. Ivanciuc O, Ivanciuc T, Douglas KJ, William SA, Balaban T (2001) Wiener index extension by counting even/odd graph distances. *J Chem Inf Model* 41(3):536–549
42. Benet LZ, Broccatelli F, Oprea TI (2011) BDDCS applied to over 900 drugs. *AAPS J* 13(4):519–547
43. Lu D, Chambers P, Wipf P, Xie X-Q, Englert D, Weber S (2012) Lipophilicity screening of novel drug like compounds and comparison to clogp. *J Chromatogr A* 1258:161–167
44. <http://www.eyesopen.com/oechem-tk>
45. QikProp (2012) version 3.5, Schrödinger, LLC, New York

46. Kerns E, Li D (2010) Drug like properties, concepts, structure design and methods. Academic Press
47. LigPrep (2012) version 2.5, Schrödinger, LLC, New York
48. Molecular Operating Environment (MOE) (2012)10; Chemical Computing Group Inc., 1010 Montreal, QC, Canada, H3A 2R7, 2012
49. Gerardo CMM, Yovani MP, Khan MTH, Arjumand A, Khan KM, Torrens F, Rotondo R (2007) Dragon method for finding novel tyrosinase inhibitors biosilico identification and experimental in vitro assays. *Eur J Med Chem* 42(11–12):1370–1381
50. <http://accelrys.com/products/discovery-studio/admet.html>
51. Karthikeyan M, Krishnan S, Pandey AK, Bender A, Tropsha A (2008) Distributed chemical computing using ChemStar: An open source java remote method invocation architecture applied to large scale molecular data from pubchem. *J Chem Inf Model* 48(4):691–703
52. <http://www.molinspiration.com/>
53. <http://www.pharmaexpert.ru/passonline/>
54. Lusi A, Pollastri G, Baldi P (2013) Deep architectures and deep learning in Chemoinformatics: the prediction of aqueous solubility for drug like molecules. *J Chem Inf Model* 53(7):1563–1575
55. Sorana BD, Lorentz J (2011) Predictivity approach for quantitative structure prediction models: application for blood barrier permeation for diverse drug like compounds. *Int J Mol Sci* 12(7):4348–4386
56. [www.preadmet.bmdrc.org/](http://www.preadmet.bmdrc.org/)
57. <http://www.epa.gov/ncct/dsstox/>
58. <http://www.epa.gov/opptintr/exposure/pubs/episuite.htm>
59. Ulrich A, Koch C, Speitling M, Hansske FG (2002) Modern methods to produce natural-product libraries. *Curr Opin Chem Biol* 6(4):453–458
60. Bemis GW, Murcko MA (1999) Properties of known drugs, 2: Side chains. *J Med Chem* 42(25):5095–5099
61. Wetzel S, Karsten K, Renner S, Rauh D, Oprea TI, Mutzel P, Waldmann H (2009) Interactive exploration of chemical space with scaffold hunter. *Nat Chem Biol* 5(9):696
62. <http://www.eyesopen.com/brood>
63. Van Drie JH (2009) ReCore. *J Am Chem Soc* 131(4):1617
64. <http://www.chemcomp.com/journal/newscaffold.htm>
65. Core Hopping (2011), version 1.1, Schrödinger, LLC, New York
66. Schuller A, Hahnke V, Schneider G (2007) SmiLib v2.0: A Java-Based Tool for Rapid Combinatorial Library Enumeration. *QSAR Comb Sci* 3:407–410
67. <http://gecco.org.chemie.uni-frankfurt.de/smilib/>
68. [http://www.chemcomp.com/MOE-Cheminformatics\\_and\\_QSAR.htm#CombinatorialLibraryDesign](http://www.chemcomp.com/MOE-Cheminformatics_and_QSAR.htm#CombinatorialLibraryDesign)
69. Tropsha A (2008) Integrated chemo and bioinformatics approaches to virtual screening. In: Tropsha A, Varnek A (ed) *Chemoinformatics approaches to virtual screening*. SC Publishing, pp 295–325
70. Perola E, Xu K, Kollmeyer TM, Kaufmann SH, Prendergast FG, Pang Y-P (2000) Successful virtual screening of a chemical database for farnesyl transferase inhibitor leads. *J Med Chem* 43(3):401–408
71. Oprea TI (2002) Virtual screening in lead discovery a viewpoint. *Molecules* 7:51–62
72. Unpublished results
73. <http://www.cdc.gov/niosh/rtecs/default.html>
74. <http://www2a.cdc.gov/nioshtic-2/>
75. <http://pubchem.ncbi.nlm.nih.gov/>
76. <https://www.ebi.ac.uk/chembl/>

Practical Chemoinformatics

Karthikeyan, M.; Vyas, R.

2014, XXI, 533 p. 621 illus., Hardcover

ISBN: 978-81-322-1779-4