**Optimized Solvated Conformers for the DrugBank Approved Library**

# Purpose:

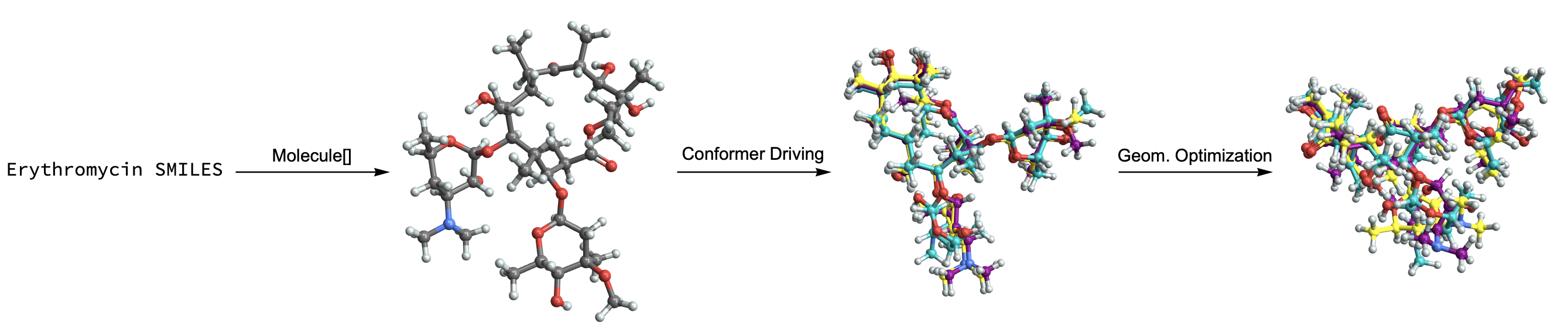
For DrugBank’s library of approved drugs, generate conformers, filter the conformers by energy and inter-conformer RMSD, then optimize their geometries using the PM7 semi-empirical Hamiltonian and a COSMO solvent model. In this way, make a library of conformers for these molecules which may represent geometries adopted in water, to facilitate *in silico* drug screening.

# Overview of workflow:

The final conformer library was constructed in six stages:

1. Remove molecules I don’t want to optimize based on defined criteria
2. Generate 3D starting geometries
   1. In the few cases that fails, download a 3D geometry from a source like ChemSpider, use ChimeraX or build manually
3. Run conformer driving and filtering
   1. Run conformer driving using OBabel
   2. Filter the first 100 conformers by ensuring no accepted conformer is within an RMSD of 2.0 Å2 from minimum energy accepted conformer, or has an energy estimate of more than +40 kcal/mol from the bottom.
4. Geometry optimize all the conformers
   1. Feed the accepted conformer geometries to MOPAC for geometry optimization using PM7 Hamiltonian and COSMO water implicit solvent model
5. Diagnose failed jobs
   1. Identify jobs with errors, fix errors and put them back through Stages III-IV
6. Extract final conformers, visualize
   1. Convert geometries to the .xyz and .sdf formats
   2. Upload these geometries to GitHub

Below is a graphical view of our workflow as applied to a single compound. Conformer driving and filtering led to three conformational states for this compound. Finally, the geometries were optimized semi-empirically in an implicit solvent.



# Stage I: Remove undesired molecules

Janina Kaderli took the Approved library from DrugBank and did the following:

* “Salts were desalted
* Duplicates were removed
* TM complexes should be intact. However, the 2D structure in the sdf-file sometimes does not show the interaction between metal and ligand, so for the fragment count they are counted as separate fragments. I tried to correct it manually in comparison with literature…”

I opened a Mathematica Notebook titled Processing\_JaninaNew.nb and imported the library I was given (*allsmiles\_DB\_APcomplex\_JaninaDesalted.txt*).

I took Janina’s library any made some additional modifications to start with. I removed compounds which were:

* Composed of < 3 heavy atoms (because for such compounds, conformer generation and refinement seems like a waste of time)
* Inorganic (based on having 0 Carbon atoms)
* Likely TM complexes (based on having a Transition or Post-Transition metal)

For the latter two cases, I just want to do some research to verify whether my existing workflow is appropriate.

That whole process yielded a library I named **compoundsToSimulate**. I exported the SMILES strings as *smilestosimulate.xlsx*. I also managed to preserve the molecule indices from the original library.

# Stage II: Generate 3D starting geometries

I split the **compoundsToSimulate** into 12 batches. I wrote a function that converts the SMILES strings of the compounds I actually wanted to simulate into starting geometries in *.xyz* format, and exports them into the appropriate sub-folder (*batchX*). **All geometries were converted to SDF format using Open Babel in a Terminal python environment. These .sdf files were used in Stage II.**

Finally, I located the indices of all the compounds Mathematica failed to build and started an Excel workbook where I’m keeping track of them, and noting the alternate way I’m obtaining 3D geometries. That workbook is *manual\_extractions\_and\_builds.xlsx.* The geometries I download/generate are going in the *extracts\_and\_builds* folder. Once these geometries are confirmed and converted to *sdf* (or “molfile”) format, I am placing them in the *BLOOPERS* folder.

# Stage III: Run conformer driving and filtering

For this stage, I used the script *sdf\_to\_confs\_2.py* to convert starting geometries into multiple conformers.

This script performs the following functions:

* Create necessary directories to organize the work
* Read in all molecules’ starting geometry files from a specified directory
* For each molecule:
  + Run a quick geometry optimization in the MMFF94 force field with the Conjugate Gradients optimization method
  + Run a conformer search using the GetConformers function of Open Babel’s [OBConformerSearch](http://openbabel.org/dev-api/classOpenBabel_1_1OBConformerSearch.shtml#a3d54183d8351d93903a41fb8e2d6eb7e) class, generating 100 conformers
* Determine conformers’ energies via the MMFF94 force field, determine the lowest energy conformer, and save that conformer’s coordinates in the geometry input file format for MOPAC, as “<name>-min-energy-conf.mopcrt”
* And then for each conformer, write an input geometry file for MOPAC *only* if the conformer geometry meets *both* these criteria:
  + The energy calculated in MMFF94 is <40 kcal/mol above the minimum energy conformer for this molecule
  + The atomic coordinates RMSD between this conformer and the minimum energy conformer is at least 2.0 Å
* Finally, for each of the conformers with a MOPAC geometry file, write a MOPAC input file according to our template *MOPAC\_template*

# Stage IV: Geometry optimize all the conformers

At this stage, MOPAC input files and starting geometries generated in Stage III were submitted to MOPAC for geometry optimization.

The MOPAC template mentioned above is simply the following:

|  |
| --- |
| \* THIS IS A PM7 CALC for $name \*  XYZ SINGLET CHECK CHARGE=$charge GEO\_DAT="$geo" &  PM7 OPT EF EPS=78.4 LET DDMIN=0 COSWRT PRTXYZ |

The keywords above are all documented in this [MOPAC manual](http://openmopac.net/manual/). The important ones:

* PM7: use the semi-empirical PM7 Hamiltonian for energy calculations
* OPT: perform geometry optimization
* EF: use the EigenFollowing routine for geometry optimization
* EPS=*n.nn*: Deploy the COSMO solvent model with the dielectric constant *n.nn*

I used the python script *monitor\_mopac\_3\_home.py* to automate the submission of jobs. This script simply submits jobs to MOPAC on my home machine in batches of some size I define, with break periods of defined length (i.e. 300 sec = 5 minutes) between batches. This allowed me to run the calculations overnight and interrupt the script the next day when my computer was needed for other tasks.

# Stage V: Diagnose failed jobs

I decided to focus on minimum-energy-conformer jobs that terminated prematurely since these were the most likely to be a real error in the setup, as opposed to something weird happening during geometry evolution. I extracted error messages the quick and dirty way in the Terminal with:

grep --after-context=3 "Error" batch<n>/MOPAC\_outfiles/out/\*-min-energy-conf.out > batch<n>errors.txt

Here n= the batch number.

Most of the errors were:

“SINGLET SPECIFIED WITH ODD NUMBER OF ELECTRONS, CORRECT FAULT”

In the case where we’re optimizing a library of closed-shell organic small molecules, that error means there’s something wrong with the input starting geometry. So far, there have been two sources of starting geometry errors:

1. OpenBabel making a mistake converting the .*xyz* geometry file exported from Mathematica to *.sdf* format.
2. A Mathematica bug wherein geometries exported as *.xyz* files don’t have atomic charges.

Both errors can be fixed thus far by exporting the geometries from Mathematica as *.sdf*’s. I’m never using *.xyz* format again actually. But I submitted a bug report to Wolfram Support along with a notebook documenting exactly what happened. Apparently, a bug fix is in the works.

The one exception error (not a “singlet”) was Compound 2154, with:

“ATOMS 38 AND 8 ARE SEPARATED BY 0.6133 ANGSTROMS.

GEOMETRY IN ERROR. TO CONTINUE CALCULATION SPECIFY 'GEO OK.”

This compound is [sulfotep](https://www.sigmaaldrich.com/US/en/product/sial/45664), an organophosphate cholinesterase inhibitor which is used (probably unwisely) as an insecticide. I fixed the error so sulfotep will appear in the final dataset, but note:

Sulfotep should not be used as a drug anyway. If you get a hit with it in screening, I recommend performing “scaffold hopping” or “ligand-based screening” to find a compound with similar 3D morphology to sulfotep, but that’s not a potentially-poisonous ecological hazard.

I compiled new starting geometries for all molecules with any of these classes of issues:

* Mathematica couldn’t write a 3D geometry, so I downloaded one from sources like ChemSpider
* None of the existing tools would build a 3D geometry except ChimeraX so I used that
* Starting geometry had an error related to my idiotic decision to export in *.xyz* format, so I exported as a *.sdf*.

These geometries were all added to what I called my “BLOOPERS” directory and the jobs were re-run.

One issue remains: Open Babel has a known set of [bugs](https://github.com/openbabel/openbabel/issues/1360) affecting the interpretation of (especially bridged) heterocycles. The structures in the MOPAC outfiles should be correct, but the .*sdf* and *.xyz* files I converted from the MOPAC outputs may have valence errors in the structures due to the Open Babel bugs.

# Stage VI: Extract final conformers, visualize

Optimized geometries generated at the end of MOPAC optimization runs were extracted from the MOPAC output files and converted to XYZ and SDF formats using Open Babel again. Some sets of finalized conformer geometry sets are displayed in the accompanying Mathematica notebook report *Solvated\_Approved\_library\_explanation.nb*.