molecular informatics

Supporting Information

PrepFlow: A Toolkit for Chemical Library Preparation and Management for Virtual Screening

Marion Sisquellas and Marco Cecchini*

Supporting Information

Average preparation time per ligand

Using a random selection of 600 ligands extracted from different libraries, we compared the median preparation time per ligand on three different computing architectures, a desktop computer in the lab (i.e. Intel(R) Core(TM) i7-4770K CPU @ 3.50GHz 8 cores), and two HPC centers, i.e. the Mésocentre de Calcul of the University of Strasbourg (*HPC1*, Intel(R) Xeon(R) CPU E5-2640 v3 @ 2.60GHz 16 cores) and the French national supercomputer GENCI-IDRIS (*HPC2*, Intel(R) Xeon(R) Gold 6248 CPU @ 2.50GHz 40 cores). In each set (FDA, Prestwick, ICSN, Maybridge, CN and ChEMBL), 100 ligands were picked randomly and merged to build a benchmark library in SDF format. The 600 ligands were then prepared simultaneously on the different machines on a single CPU for comparison. Once the preparation was finished, the Time.dat file was analyzed and the preparation time per ligand DONE (see below) extracted. This information was then used to evaluate the statistical distribution of the preparation time per ligand; see Figure S1. Since the distributions are peaked but present long tails at large preparation times, the characteristic preparation time per architecture was estimated from the median of the distribution. The median preparation time per ligand was 10.47, 26.99, and 22.93 seconds on the *Lab computer*, *HPC1*, and *HPC2*, respectively.

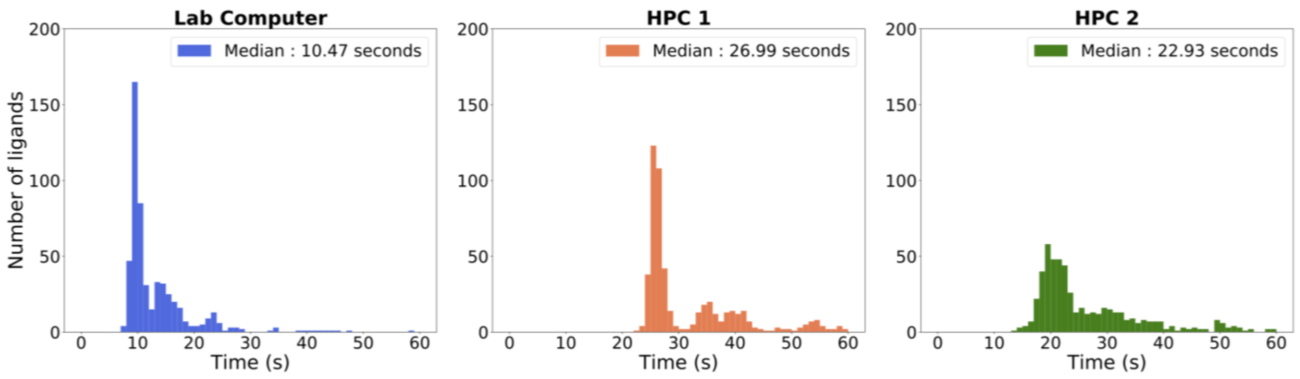

Figure S1. Statistical distribution of the preparation time per ligand on different computing architectures using 1 CPU core.

To establish the optimal settings for using PrepFlow, the preparation of the same benchmark library (600 ligands) was redone using an increasing number of CPU cores on the three architectures (Figure S2). The results show that the total preparation time decreases with the number of CPU cores. However, it appears that the scaling factor decreases rapidly, such that *PrepFlow* performances plateau at 2 CPU cores on *Lab computer* and *HPC2*, and drop considerably at 8 CPU cores on *HPC1*, possibly due to communication bottlenecks; see Figure S2A. Analysis of the median preparation time per ligand provides means to quantify the gain in performances when running on multiple cores per CPU. Consistent with results in Figure S1, ligand preparation is always faster on the *Lab computer*, which is explained by the higher frequency of the CPU in these latest-generation processors. And, when running on 2 cores per CPU, performances increase by 30-40% on both the *Lab computer* and *HPC2*, which introduces an extra gain in performances (Figure S2B). Altogether and based on the results above, we strongly recommend using *PrepFlow* on distributing computing and if possible running on 2 cores per CPU to optimize its performances.

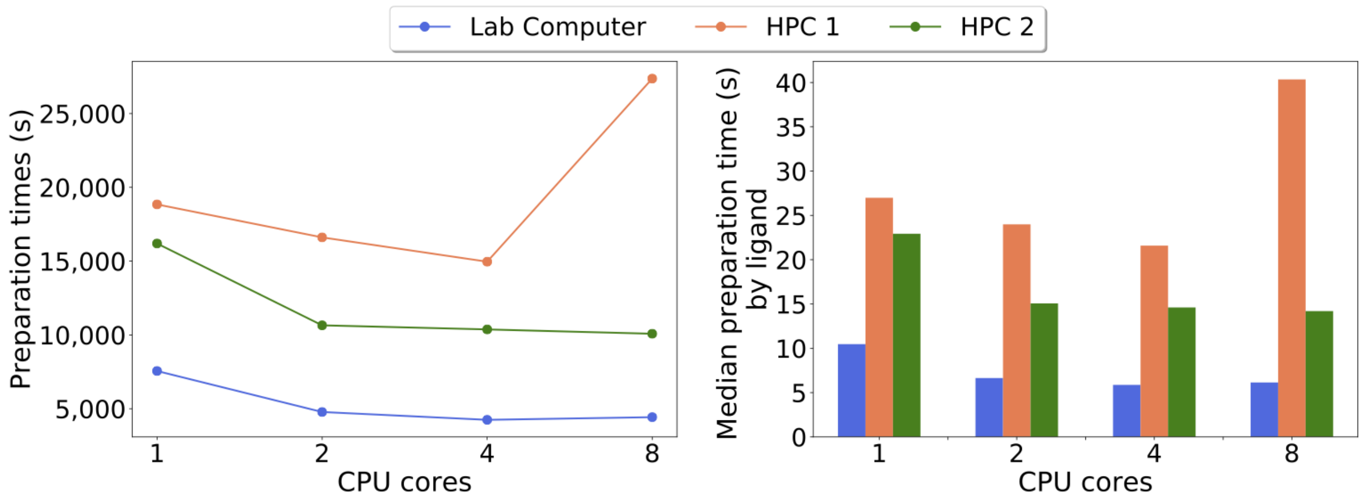

Figure S2. Benchmark of the preparation time of PrepFlow on three computing architectures. (A) The total preparation time of the DONE ligands (548 over 600 ligands) on various machines is given as a function of the number of CPU cores used. (B) The median time preparation time per ligand is displayed per architecture depending on the number of CPU cores used.

Computational time for ChEMBL (1 941 406 ligands) with 81 parallel preparations using 2 CPU cores on HPC1

$$Theorical\ preparation\ \ time = \left. \left(\frac{(AverageTimeByLig * SizeLib)}{3600\ (s) * 24\ (h)}\right) \middle/ \begin{matrix} Number\ of\ jobs \\ using\ 2\ CPU\ cores \\ running\ in\ parallel \end{matrix} \right. \quad (1)$$

$$= \left. \left(\frac{(23.99 * 1941406)}{3600\ (s) * 24\ (h)}\right) \middle/ 81 \right.$$

$$= \text{ 6.65 days}$$

Ratio undone ligands over the total number of ligands in each libraries:

The ratio (Formula 2) quantifies the numbers of ligands disregarded by PrepFlow.

$$RATIO = \frac{UNDONE}{TOTAL} \quad (2)$$

$$= \frac{ERROR\ +\ PASS}{DONE\ +\ SKIP\ +\ ERROR\ +\ PASS}$$

With DONE for the prepared ligands, SKIP for the ligands already present in the DATABASE, ERROR for ligand presenting an error during the preparation and PASS for ligands not passing the filtering step.

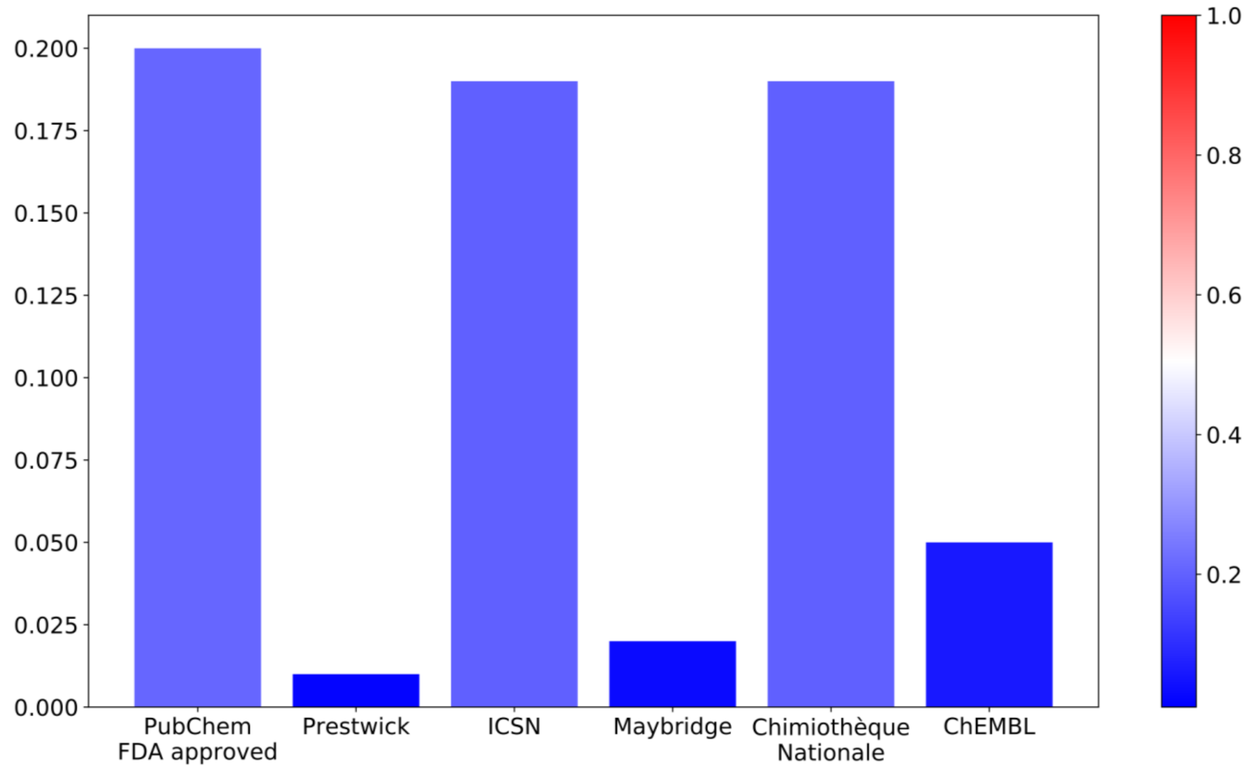

Figure S3. Ratio of disregarded ligands in the different library

	RATIO	DONE	SKIP	ERROR	PASS
PubChem FDA approved	0,20	1014	16	11	244
Prestwick	0,01	1500	6	9	5
ICSN	0,19	4322	96	20	1028
Maybridge	0,02	52045	103	863	341
Chimiothèque Nationale	0,19	51317	11299	5664	9027
ChEMBL	0,05	1415978	107713	7222	72847

Table S1. Number of ligands by status in each library.

PrepFlow User Manual

In this section, we present the user manual of PrepFlow. First, the pre-requisites in terms of existing software and libraries for the installation and use of PrepFlow are indicated. The command lines to download and install them in the Linux operating environment are also provided. Second, all available options and defaults in a typical PrepFlow execution are described. Third, a series of tutorials illustrating the PrepFlow functionalities and running on a desktop computer or an HPC environment are given.

I. Pre-requisites:

a) Python 3:

PrepFlow is coded in python3, so it needs to be present on the machine. NumPy and pandas, two libraries used by PrepFlow can be downloaded using:

- `sudo apt-get install python3-numpy`
- `sudo apt-get install python3-pandas`

b) ChemAxon:

After creating an account on ChemAxon website, the `marvin.deb` and `structure_representation_toolkit.deb` are available for download. To install ChemAxon:

- *Cxcalc* and *molconvert*:
  - `sudo dpkg -i marvin.deb`
- *Standardize*
  - `sudo dpkg -i structure_representation_toolkit.deb`
- The license (license.cxl) is mandatory to use these tools, it can be asked on the website of ChemAxon. Once the license is given, the license.cxl need to be put in the .chemaxon folder present in the user home.

c) RDKit:

A virtual environment is required to use RDKit. Using the following command line a virtual environment, called `virtual_env`, is created using Conda with RDKit ready to be used.

- `conda create -c conda-forge -n virtual_env rdkit`

Once the virtual environment is created, it can be activated and deactivated as required. When it will be activated RDKit will be available.

- `conda activate virtual_env`
- `conda deactivate virtual_env`

d) Export the path of PrepFlow to use it with the PrepFlow tag name

Add PrepFlow to your .bashrc

- `export PREPFLOW_HOME=/`*`PATH`*`/PREPFLOW/`

- `export PATH=${PATH}:${PREPFLOW_HOME}/bin/`

II. Options:

PrepFlow presents two types of options, mandatories and optional. The mandatories must be provided to launch the preparation unlike the optional that have default values.

To discover all the options available on PrepFlow:

-h, --help show this help message and exit

a) Mandatory Options:

Parameter	Description
-i	Input file in SDF or SMILES format
-proto	Name of the preparation folder
-database	Name of the library prepared in the database folder

b) Optional:

- Input file:

Parameter	Default Value	Description
--header_smile_file	[**No**/Yes]	To specify if the SMILES input file contains a header
--separator_smile_file	[‘,’]	To specify the separator field in the SMILES input file

- Filtering

Parameter	Default Value	Description
--prepare_all_atom	[**no**/yes]	By default, ligands presenting an atom not in the list ['H', 'B', 'C', 'N', 'O', 'P', 'S', 'F', 'Cl', 'Br', ‘I’] are discarded. To prepare all the atoms option yes
--number_maximum_chiral_center_undefined	3	Ligands with a number of undefined chiral center larger than the threshold are discarded, by default 3
--number_size_ring_max	7	Ligands with rings larger than the threshold are discarded, by default 7-membered rings.

--number_rotatable_bonds_max	20	Ligands presenting a number of rotatable bonds upper the default value are passed

- Database

Parameter	Default Value	Description
--database_path	[**no**/path to database folder]	Path of the database to increment
--database_name	[**no**/name of the database folder]	By default PrepFlow name the database folder DATABASE, with this flag it can be change

- Preparation

Parameter	Default Value	Description
--ph	7.0	pH value used to compute the dominant tautomers distribution, by default 7.0
--lim_tauto	10	Discard tautomers whose occurrence probability is lower than the threshold, by default 10%

- HPC Launching

Parameter	Default Value	Description
—hpc	[**no**/yes]	To launch on HPC
--header_hpc	[**no**/header file]	Header of the launching system used on HPC (i.e. slurm)
--number_ligands_by_files	1500	When running in parallel, PrepFlow launches multiple preparations on different CPU. This parameter defines the maximum number of ligands per job, by default 1500.
--archive_summary	[**no**/ARCHIVE file]	Archive file

- Merging after HPC preparation

Parameter	Default Value	Description

Parameter	Default Value	Description
—merge	[**no**/yes]	To concatenate all the output of PrepFlow on HPC in one protocol folder and one database

- Implementation Database after HPC preparation

Parameter	Default Value	Description
—database_update	[**no**/yes]	To concatenate all the output of PrepFlow on HPC
—ligands_to_extract	[**no**/yes]	File containing the ligands to extract from the database

- Statistics

Parameter	Default Value	Description
--statistics	[**No**/Yes]	To calculate the features distributions of the input file, no preparation is done

III. Tutorials

The tutorial provides examples to launch PrepFlow with different input files, i.e. SMILES or SDF. The SMILES file, Trypsine_10L.smi, contains 10 ligands, each row is composed by the name of the ligand and its smile. The file, Trypsine_3L.smi, contains only the 3 first ligands of Trypsine_10L.smi. The SDF file, ChEMBL_10L.sdf, also presents 10 ligands.

The use of PrepFlow is shown on several machines, i.e. a lab computer and an HPC, with or without the archive strategy.

a) Launching PrepFlow interactively

- *Tuto 1: Preparation of a SMILES file input*

```
PrepFlow -i Trypsine_10L.smi -proto Trypsine -database Trypsine
--header_smile_file yes --separator_smile_file ','
```

- *Tuto 2: Preparation of a SDF file input*

```
PrepFlow -i ChEMBL_10L.sdf -proto ChEMBL  -database ChEMBL
```

- *Tuto 3: Launching two preparations using the same database*

First library:

```
PrepFlow -i Trypsine_10L.smi -proto Trypsine -database Trypsine
--header_smile_file yes --separator_smile_file ','
```

Second library using the same archiving system:

```
PrepFlow -i ChEMBL_10L.sdf -proto ChEMBL  -database ChEMBL
--database_path /[PATH]/Tuto_3/DATABASE/
```

- *Tuto 4: Launching twice the same ligands to use the database extraction*

First library:

```
PrepFlow -i Trypsine_3L.smi -proto Trypsine_3L -database Trypsine
--header_smile_file yes --separator_smile_file ','
```

To prepare Trypsine.smi again but with new ligands:

```
PrepFlow -i Trypsine_10L.smi -proto Trypsine_10L -database
Trypsine   --header_smile_file yes --separator_smile_file ',' --
database_path /[PATH]/Tuto_4/DATABASE/
```

b) Launching PrepFlow on HPC

To launch calculations on HPC, a header for the job scheduler operating on the master node is required, an example of a slurm header is presented in Figure S4. The SBATCH lines can be modified to specify to the HPC the partition to use (-p), the name of the job (--job-name=), the number of nodes (-N), the number of cores (-n) and the maximum execution time requested (-t). According to the HPC used, it may be necessary to load modules for the software to function properly.

```
#! /bin/bash
# CPU 1 node 2 cores
#SBATCH -p public
#SBATCH --job-name="Preparation"
#SBATCH -N 1
#SBATCH -n 2
```

```
#SBATCH -t 24:00:00

module load python/python3
```
Figure S4: Example of a slurm header

i) Without the archive system

- *Tuto 5: Launching on HPC*

```
PrepFlow -i ChEMBL_10L.sdf -proto ChEMBL -database ChEMBL --hpc
yes --header_hpc header_HPC_example.txt --number_ligands_by_files
2
```

- *Tuto 6: Merging after Launching on HPC*

```
PrepFlow -i ChEMBL_10L.sdf -proto ChEMBL -database ChEMBL --hpc
yes --header_hpc header_HPC_example.txt --number_ligands_by_files
2 --merge yes
```

- *Tuto 7: Incrementation DATABASE after HPC*

```
PrepFlow -i ChEMBL_10L.sdf -proto ChEMBL_all -database ChEMBL --
hpc yes --header_hpc header_HPC_example.txt --
number_ligands_by_files 2 --database_update yes --database_path
/[PATH]/Tuto_7/DATABASE
```

ii) With the archive system

- *Tuto 8: Launching on HPC with ARCHIVE file*

```
PrepFlow -i Trypsine_10L.smi -proto Trypsine -database Trypsine
--header_smile_file yes --separator_smile_file ',' --hpc yes
--header_hpc header_HPC_example.txt  --number_ligands_by_files 2
--archive_summary ARCHIVE.txt
```

- *Tuto 9: Merging after Launching on HPC with ARCHIVE file*

```
PrepFlow -i Trypsine_10L.smi -proto Trypsine -database Trypsine
--header_smile_file yes --separator_smile_file ',' --hpc yes
--header_hpc header_HPC_example.txt  --number_ligands_by_files 2
--archive_summary ARCHIVE.txt --merge yes
```

- *Tuto10: Incrementation DATABASE after HPC with ARCHIVE file*

```
cd Trypsine

PrepFlow -i Trypsine_10L.smi -proto Trypsine_all -database
Trypsine   --header_smile_file yes --separator_smile_file ','
--hpc yes  --header_hpc header_HPC_example.txt  --
number_ligands_by_files 2   --database_update  yes --database_path
/[PATH]/Tuto_10/DATABASE  --ligands_to_extract
Ligands_to_extract_in_archive.txt
```