Basic YARP usage

This document contains instructions for running a YARP task.

- First of all, try to get familiar with cluster (bash command) and EMACS.
- For cluster related doubts refer:

  brown

  halstead

  bell
- For understanding the principle of Yet Another Reaction Program (YARP), find here.

Follow the steps as listed below to perform YARP on your interested systems.

- **Set up a YARP directory on cluster**
  - Log in to the cluster (take bell for example)

    ```
    ssh $USEERNAME@bell.rcac.purdue.edu
    ```

    where $USERNAME is your Purdue username.
  - Create one working folder for YARP, i.e.

    ```
    mkdir bin
    cd bin
    mkdir YARP
    cd YARP
    ```
  - Get a copy of YARP from depot

    ```
    git clone /depot/bsavoie/etc/YARP/ .
    ```
  - Check what in YARP package

```
cd YARP
ls
```

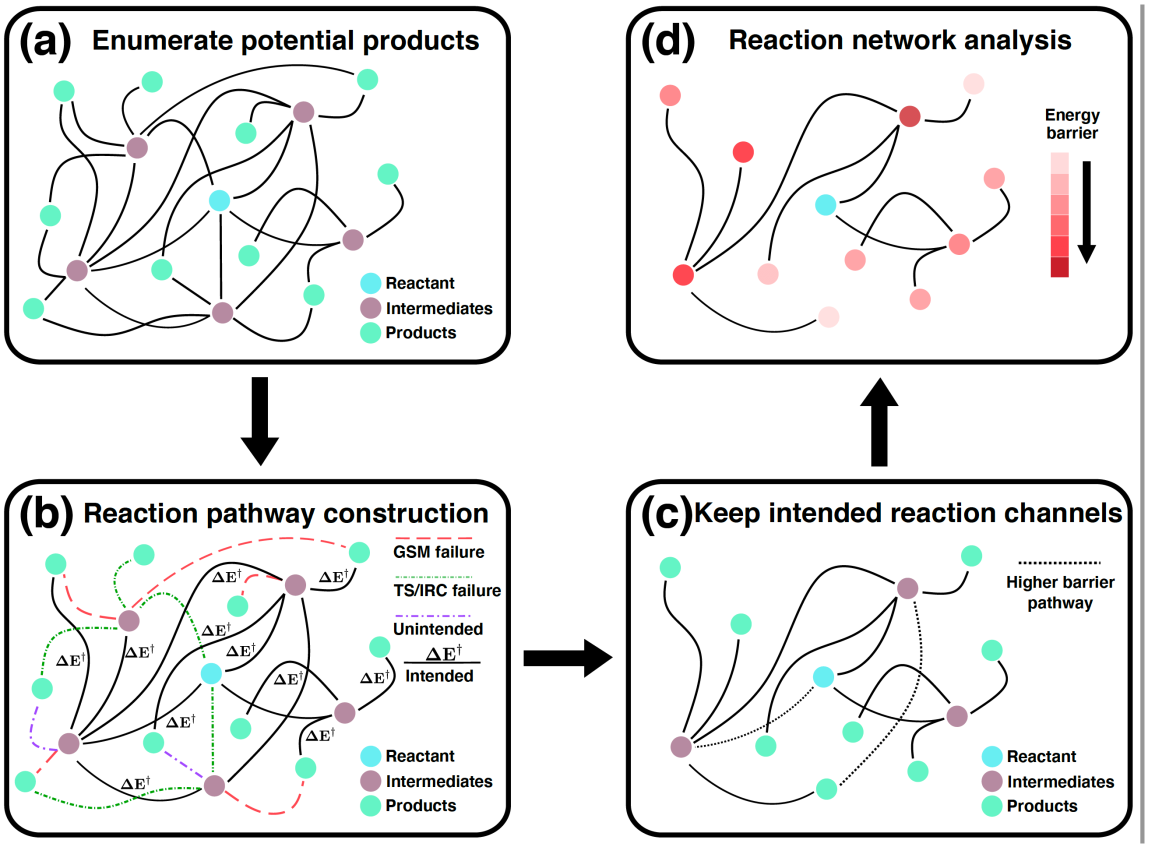
You'll find there are many folders, like version1.0, version2.0 and YARP-catalysis, let's focus on version 2.0.

```
cd version2.0
```

- **Set up needed packages/softwares for YARP2.0**
    - Once you have successfully changed directory to version2.0, check what in this folder again.

    ```
    ls
    ```

    As shown in the flow chart of YARP, there are three main steps in YARP methodology, (a) potential products enumeration following elementary reaction step (ERS) definition; (b) reaction pathway construction using and (d) reaction network analysis. The folder **ERS_enumeration** and **Construct_pathway_Gaussian** (**Construct_pathway_Orca** is also written for applying Orca as quantum chemistry engine) correspond to step (a) and (b), respectively. There is no automated program for step (d) but will add in the future. In the current version, we have to **manually** find important intermediates for the next step. There are some other modules associated with other projects. For instance, **catalysis** is for water-catalysis reaction study; **CONF_GEN** is an external package for performing conformational sampling (incorporated in YARP); **model_reaction** is for generating and utilizing YARP model reaction. In this document, we'll figure out how to use **ERS_enumeration** and **Construct_pathway** modules.


(a) Enumerate potential products
Reactant
Intermediates
Products
(b) Reaction pathway construction
GSM failure
TS/IRC failure
Unintended
ΔE†
Intended
(c) Keep intended reaction channels
Higher barrier pathway
(d) Reaction network analysis
Energy barrier

- Add the anaconda installed in the depot to your local environment:

```
export PATH="/depot/bsavoie/apps/anaconda3/bin/:$PATH"
```

- Load the anaconda environment (named as python3) which is built for YARP

```
source activate python3
```

- Add xTB to your local environment:

```
emacs ~/.bash_profile
export PATH="/depot/bsavoie/apps/xTB/xtb_6.4.0/bin:$PATH"
```

  After saving this file, then source the bash file and test:

```
source ~/.bash_profile
which xtb
```

  if it prints out a pathway to xtb then you've succeeded. (Only need to be done the first time)

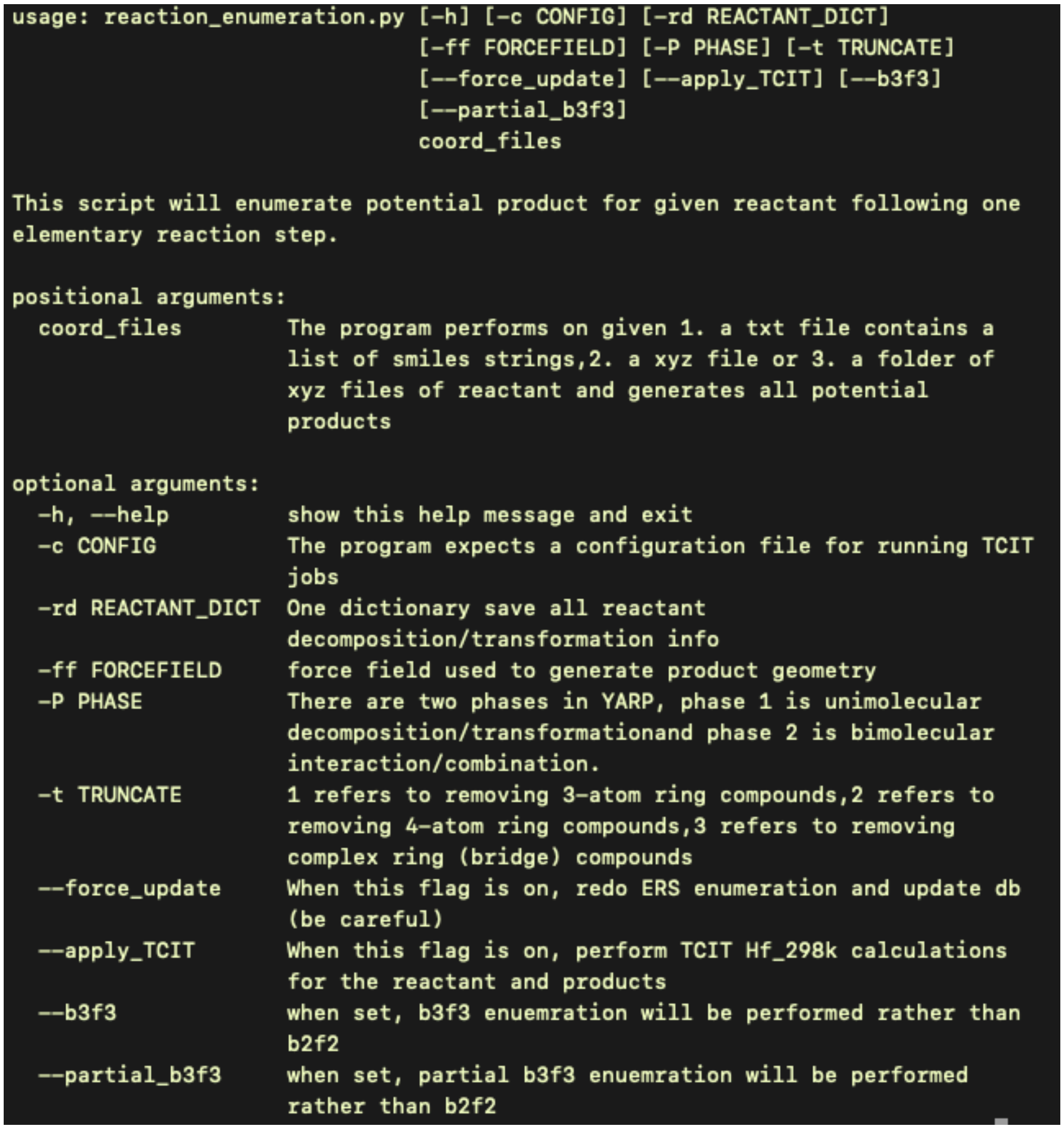
- **Start using YARP2.0**
    - Usage of **ERS_enumeration**

      ```
      cd ERS_enumeration
      ```

        - Put xyz file(s) of your interested reactant(s) in 'Reactant' folder or create a .txt file containing the smiles string of reactants. (*Note: 1. to generate xyz files, use Avogadro to draw chemical species and save as xyz files; then scp it to the cluster; 2. an example of txt file is Reactant/KHP_net.txt; 3. for bimolecular reactions, xyz file is recommended*)
        - Obtain help message by running:

      ```
      python reaction_enumeration.py -h
      ```

      If all python packages are correctly installed, this command will print out following messages:



```
usage: reaction_enumeration.py [-h] [-c CONFIG] [-rd REACTANT_DICT]
                               [-ff FORCEFIELD] [-P PHASE] [-t TRUNCATE]
                               [--force_update] [--apply_TCIT] [--b3f3]
                               [--partial_b3f3]
                               coord_files

This script will enumerate potential product for given reactant following one
elementary reaction step.

positional arguments:
  coord_files          The program performs on given 1. a txt file contains a
                       list of smiles strings,2. a xyz file or 3. a folder of
                       xyz files of reactant and generates all potential
                       products

optional arguments:
  -h, --help           show this help message and exit
  -c CONFIG            The program expects a configuration file for running TCIT
                       jobs
  -rd REACTANT_DICT    One dictionary save all reactant
                       decomposition/transformation info
  -ff FORCEFIELD       force field used to generate product geometry
  -P PHASE             There are two phases in YARP, phase 1 is unimolecular
                       decomposition/transformationand phase 2 is bimolecular
                       interaction/combination.
  -t TRUNCATE          1 refers to removing 3-atom ring compounds,2 refers to
                       removing 4-atom ring compounds,3 refers to removing
                       complex ring (bridge) compounds
  --force_update       When this flag is on, redo ERS enumeration and update db
                       (be careful)
  --apply_TCIT         When this flag is on, perform TCIT Hf_298k calculations
                       for the reactant and products
  --b3f3               when set, b3f3 enuemration will be performed rather than
                       b2f2
  --partial_b3f3       when set, partial b3f3 enuemration will be performed
                       rather than b2f2
```

Here, -rd refers to the reactant dictionary file. (i.e I make one which is ../dict/unireactant.p) If your input reactant is already contained in this dictionary, the program will do nothing but tell you "already in reactant dictionary, directly take it for next step"

-ff will specify a force field to generate product 3D geometry (default: mmff94). UFF can be an alternative option.

-P: if the input xyz file contains one compound, set phase to be 1 to perform unimolecular reaction; else if the input xyz file contains two compounds, set P to be 2 to perform bi-molecular transformation. **The current code can actually automatically identify this value, so there is no need for specifying it**

- Example

```
python reaction_enumeration.py Reactant/ketohydroperoxide.xyz -rd
../dict/unireactant.p -t [3]
```

Once this step successes (no error message appear), we can move on to the second step.

- Usage of **Construct_pathway_Gaussian**

```
cd ../Construct_pathway_Gaussian
```

  - use config.txt file to control the parameters Since "Construct_pathway" has more parameters, I created a config file to manage all of the parameters. Two example config files can be found in /depot/bsavoie/data/YARP-example, namely config_KHP.txt and config_bimole.txt. Make sure replacing all 'zhao922' by your user name in this file and make sure the YARP pathway matches with your setting.

  - Control input reactant by modifying input reactant list. `emacs input_list.txt`

Put first 14 characters of inchikey of your input compound(s). The inchikey will be printed out when running the reaction enumeration, or you can obtain the inchikey from one xyz file, use openbabel as:

```
obabel -ixyz <path_to_xyz_file> -oinchikey
```

e.g, running

```
obabel -ixyz ../ERS_enumeration/Reactant/ketohydroperoxide.xyz -oi
nchikey
```

The inchikey for ketohydroperoxide is 'XSASRUDTFFBDDK-UHFFFAOYSA-N', copy 'XSASRUDTFFBDDK' into input_list.txt.

  - Turn off the TCIT flag if you don't have access to TCIT. (In the example cases apply_TCIT is set to be true since the TCIT calculation has been already done.)

```
apply_TCIT       True --> apply_TCIT       False
```

  - cluster settings

You can use my settings. Many of them are just empirical parameters. When you are familiar with slurm job submission, feel free to modify these settings.
- run locate_TS.py to construct reaction pathways

```
python locate_TS.py
```

- Running two specific example:

    - First copy the example files into **Construct_pathway_Gaussian** folder.

```
cd Construct_pathway_Gaussian
cp /depot/bsavoie/data/YARP-example/*.txt .
```

Then modify the config_KHP.txt (and config_bimole.txt) by changing zhao922 by your username and double checking the YARP pathway matches with your setting. Also make sure creating a sub-folder in the scratch (mkdir /scratch/bell/USERNAME/example).

Then just run the YARP calculation by:

```
python locate_TS.py -c config_KHP.txt
```

Similar for config_bimole.txt. After the jobs are finished, you should have two output folders KHP and bimole in /scratch/bell/USERNAME/example. You can compare those with the outputs in /depot/bsavoie/data/YARP-example. Two important output files are report.txt and IRC-result/IRC-record.txt.