VFFVA Documentation

Release v0.1

Marouen Ben Guebila

Aug 10, 2022

Contents

VFFVA is a parallel implementation of Flux Variability Analysis (FVA), which delineates the maximal and minimal bounds for each reaction in a given metabolic model formulated as a Linear Program (LP).

FVA is widely used as it allows to quantify the effect of a perturbation (drug, disease, nutrition) on the system as whole and evaluates the response on the reaction actitivity range.

FVA has been originally implemented in MATLAB and Fast FVA (FFVA), the C implementation of FVA, had a great speedup on the original implementation through the use of innovative features of the C API of the LP solvers IBM CPLEX and GLPK.

As models grow in size to cover thousands of reactions and metabolites to account for communities of bacteria and connected systems of organs, FVA is used mostly in parallel. The standard implementation assumes that each computer cores take an equal chunk of reactions (static load balancing) to process and find their minimal and maximal bounds.

In most cases, the chunks of reactions that each core has, are not equal in processing times, despite being equal in number. The reasons can be manifold as they can related to the core processing speef, the numerical feasibility of the LP to solve etc.

VFFVA considers an implementation that assign each worker chunks of reactions that are not equal in size but are equal in their processing time. This process is achieved through dynamic load balancing, where if a worker finishes an assigned chunk and stays idle, it is assigned another chunk from an overloaded worker, such that the load on all workers is equally balanced.

CHAPTER 1

Contents

1.1 Installation guide

VFFVA is supported in Linux systms in both C and MATLAB.

1.1.1 C

The C implementation is a hybrid MPI/OpenMP code that has two levels of parallelism in both shared memory and non-shared memory systems.

Requirements

- Linux-based system (tested on Ubuntu 16.04, 18.04, and 22.04).
- IBM CPLEX 12.6.3. (tested on 12.6.3, 12.10, and 22.1.0) and above Free academic version.
- OpenMP comes be default in the latest gcc version (For macOs, OpenMp has to be installed separately)
- MPI through the OpenMPI 1.10.3 implementation.

Installation

You need to download and install 1) OpenMPI and 2) IBM CPLEX for 64-bit machines.

- MPI: You can use the following code snippet to install MPI

```
VERSION=3.1.2
wget --no-check-certificate https://www.open-mpi.org/software/ompi/v3.1/downloads/
↪openmpi-$VERSION.tar.gz
tar -xzf openmpi-$VERSION.tar.gz
cd openmpi-$VERSION
mkdir build && cd build
```

(continues on next page)

(continued from previous page)

```
../configure CFLAGS="-w" --prefix=$HOME/open-mpi \
            --without-verbs --without-fca --without-mxm --without-ucx \
            --without-portals4 --without-psm --without-psm2 \
            --without-libfabric --without-usnic \
            --without-udreg --without-ugni --without-xpmem \
            --without-alps --without-munge \
            --without-sge --without-loadleveler --without-tm \
            --without-lsf --without-slurm \
            --without-pvfs2 --without-plfs \
            --without-cuda --disable-oshmem \
            --disable-mpi-fortran --disable-oshmem-fortran \
            --disable-libompitrace \
            --disable-mpi-io  --disable-io-romio \
            --disable-static #--enable-mpi-thread-multiple
make -j2
make install
```

You might also need to add MPI path

```
export PATH=$HOME/open-mpi/bin:$PATH
```

- IBM CPLEX: The recommended approach is to download IBM CPLEX and register for the free academic version.

Here is a code snippet for installing CPLEX for Ubunutu, this is only an example and you need to get your installation file after creating an IBMid. In this example, CPLEX will be installed in ~/CPLEX_Studio1210.

```
VERSION_CPLEX=1210
wget "https://ak-dsw-mul.dhe.ibm.com/sdfdl/v2/fulfill/CC439ML/Xa.2/Xb.XwdHXGdhWvrm/Xc.
↪CC439ML/cplex_studio1210.linux-x86-64.bin/Xd./Xf.lPr.D1VC/Xg.10736638/Xi./XY.
↪scholars/XZ.UBs6UV1K_zA_5uS6T9I81YuWNmI/cplex_studio1210.linux-x86-64.bin#anchor"
chmod +x cplex_studio$VERSION_CPLEX.linux-x86-64.bin

#specify install options
echo "INSTALLER_UI=silent\n
INSTALLER_LOCALE=en\n
LICENSE_ACCEPTED=true\n
USER_INSTALL_DIR=$HOME/CPLEX_Studio$VERSION_CPLEX"> myresponse.properties

#work around installation bug
export PS1=">"

#install
./cplex_studio$VERSION_CPLEX.linux-x86-64.bin -f "./myresponse.properties"
```

Then, make sure that the CPLEXDIR path in `lib/Makefile` corresponds to the installation folder of CPLEX (~/CPLEX_Studio1210 in the previous example).

- Once the required dependencies installed, `cd VFFVA/lib` then `make` at the root of `lib`.
- Alternatively, you can open an issue here.

1.1.2 MATLAB

VFFVA.m is the MATLAB implementation that consists of a wrapper around the C version.

Requirements

- Linux-based system.
- MATLAB
- IBM CPLEX 12.6.3. and above Free academic version.
- OpenMP comes be default in the latest gcc version.
- MPI through the OpenMPI 1.10.3 implementation.

Installation

First, istall the C version, then add the path of the installed C version to your MATLAB path.

```
addpath(genpath('VFFVA'))
```

1.1.3 Python

VFFVA.py is the Python3 implementation that consists of a wrapper around the C version.

Requirements

- Linux-based system.
- Python 3
- IBM CPLEX 12.6.3. and above Free academic version.
- OpenMP comes be default in the latest gcc version.
- MPI through the OpenMPI 1.10.3 implementation.

Installation

First, istall the C version, then add the path of the installed C version to your Python path.

1.1.4 FAQ

- In MacOS, I get the error "Clang: Error: Unsupported Option '-Fopenmp'" Error -> In MacOS, OpenMP is not provided by default, therefore you need to install it by updating to the latest version of llvm.
- Too many output arguments with function BuildMPS -> The version of BuildMPS function provided with VFFVA gives 2 outputs, if you have the COBRAtoolbox in your path, you might be using another version that gives 1 output. Therefore, make sure that VFFVA path supersedes COBRAtoolbox path in MATLAB path.

1.2 Usage guide

The following provides the usage guide for both the C and MATLAB versions of VFFVA.

1.2.1 C version

After installing the dependencies of VFFVA, you can build the binaries at the root of lib using make.

Then call VFFVA as follows:

```
mpirun -np nCores --bind-to none -x OMP_NUM_THREADS=nThreads veryfastFVA
model.mps OPTPERC SCAIND ex
```

Replace the following variables with your own parameters:

- nCores: the number of non-shared memory cores you wish to use for the analysis
- nThreads: the number of shared memory threads within one core
- model.mps: the metabolic model in .mps format. To convert a model in .mat format to .mps, you can use the provided converter convertProblem.m
- OPTPERC: Optimization percentage of the objective value (0-100). The default is 90, where VFFVA will be computed with the objective value set to 90% of the optimal objective.
- SCAIND: (optional) corresponds to the scaling CPLEX parameter SCAIND and can take the values 0 (equilibration scaling: default), 1 (aggressive scaling), -1 (no scaling). scaling is usually desactivated with tightly constrained metabolic model such as coupled models to avoid numerical instabilities and large solution times.
- ex: .csv file containing 0-based indices of reactions to optimize e.g., 0,1,2,3,4,5 or check rxns.csv in the repository.

Example: `mpirun -np 2 --bind-to none -x OMP_NUM_THREADS=4 veryfastFVA ecoli_core.mps`

VFFVA will perform 2n Linear Programs (LP), where n is the number of reactions in a metabolic model, corresponding to a minimization and a maximization in each dimension.

The ouput file is saved as modeloutput.csv, with model is the name of the metabolic model.

1.2.2 MATLAB version

The MATLAB version VFFVA.m is a wrapper around the C version, which means that the previous installation steps of the C version have to be performed.

Then VFFVA.m can be called from MATLAB using the following function description:

```
  USAGE:

     [minFlux,maxFlux]=VFFVA(nCores, nThreads, model, scaling, memAff, schedule,␣
↪nChunk, optPerc)

  INPUT:
     nCores:           Number of non-shared memory cores/machines.
     nThreads:         Number of shared memory threads in each core/machine.
     model:            .mps format: path to metabolic model in .mps format.
                       COBRA format: will be automatically formatted to .mps format.␣
↪Make sure to add VFFVA folder to
                       your MATLAB path to access the conversion script.

  OPTIONAL INPUTS:
     scaling:          CPLEX parameter. It corresponds to SCAIND parameter (Default␣
↪= 0).
```

(continues on next page)

(continued from previous page)

```
                        -1: no scaling; 0: equilibration scaling; 1: more aggressive␣
↪scaling.
                        more information here: https://www.ibm.com/support/
↪knowledgecenter/SSSA5P_12.7.0/ilog.odms.cplex.help/CPLEX/Parameters/topics/ScaInd.
↪html.
     optPerc:           Percentage of the optimal objective used in FVA. Integer␣
↪between 0 and 100. For example, when set to 90
                        FVA will be computed on 90% of the optimal objective.
      memAff:           'none', 'core', or 'socket'. (Default = 'none'). This an␣
↪OpenMPI parameter, more
                        information here: https://www.open-mpi.org/faq/?
↪category=tuning#using-paffinity-v1.4.
      schedule:         'dynamic', 'static', or 'guided'. (Default = 'dynamic'). This␣
↪is an OpenMP parameter, more
                        information here: https://software.intel.com/en-us/articles/
↪openmp-loop-scheduling
      nChunk:           Number of reactions in each chunk (Default = 50). This is an␣
↪OpenMP parameter, more
                        information here: https://software.intel.com/en-us/articles/
↪openmp-loop-scheduling
      ex:               0-based indices of reactions to optimize. (Default, all␣
↪reactions)

  OUTPUTS:
     minFlux:           (n,1) vector of minimal flux values for each reaction.
     maxFlux:           (n,1) vector of maximal flux values for each reaction.
```

1.2.3 Python version

The Python version VFFVA.py is a wrapper around the C version, which means that the previous installation steps of the C version have to be performed.

Then VFFVA.py can be imported into a Python 3 script using the following function description:

```
   USAGE:
   minFlux,maxFlux=VFFVA(nCores, nThreads, model, scaling, memAff, schedule, nChunk,␣
↪optPerc, ex)

   :param nCores:   Number of non-shared memory cores/machines.
   :param nThreads: Number of shared memory threads in each core/machine.
   :param model:    .mps format: path to metabolic model in .mps format.
   :param scaling:  CPLEX parameter. It corresponds to SCAIND parameter (Default =␣
↪0).
                    -1: no scaling; 0: equilibration scaling; 1: more aggressive␣
↪scaling.
                    more information here: https://www.ibm.com/support/
↪knowledgecenter/SSSA5P_12.7.0/ilog.odms.cplex.help/CPLEX/Parameters/topics/ScaInd.
↪html.
   :param memAff:   'none', 'core', or 'socket'. (Default = 'none'). This an OpenMPI␣
↪parameter, more
                    information here: https://www.open-mpi.org/faq/?category=tuning
↪#using-paffinity-v1.4.
   :param schedule: 'dynamic', 'static', or 'guided'. (Default = 'dynamic'). This is␣
↪an OpenMP parameter, more
```

(continues on next page)

(continued from previous page)

```
                    information here: https://software.intel.com/en-us/articles/
↪openmp-loop-scheduling
    :param nChunk:   Number of reactions in each chunk (Default = 50). This is an␣
↪OpenMP parameter, more
                    information here: https://software.intel.com/en-us/articles/
↪openmp-loop-scheduling
    :param optPerc:  Percentage of the optimal objective used in FVA. Integer between␣
↪0 and 100. For example, when set to 90
                    FVA will be computed on 90% of the optimal objective.
    :param ex:       Indices of reactions to optimize as a numpy array.  (Default,␣
↪all reactions)
    :return:
           minFlux:          (n,1) vector of minimal flux values for each reaction.
           maxFlux:          (n,1) vector of maximal flux values for each reaction.
```

1.3 Tutorials

First, make sure that VFFVA.m in MATLAB is correctly installed.

1.3.1 Comparison of the results of FVA and VFFVA

In this tutorial, we would like to compare the consistency of the results between the COBRA Toolbox FVA function and VFFVA.

- Install the COBRA Toolbox through entering in your command prompt

```
git clone https://github.com/opencobra/cobratoolbox.git
```

- Then launch MATLAB and add COBRA Toolbox to the path

```
addpath(genpath(\path\to\cobratoolbox))
```

- Initiate the COBRA Toolbox

```
initCobraToolbox
```

- Change the solver to IBM CPLEX

```
changeCobraSolver('ibm_cplex')
```

- Run FVA on Ecoli core model

```
load ecoli_core_model.mat
ecoli=model;
optPercentage=90;
[minFluxFVA,maxFluxFVA]=fluxVariability(ecoli, optPercentage);
```

- Run VFFVA on Ecoli core model

```
nCores=1;
nThreads=4;
load ecoli_core_model.mat
```

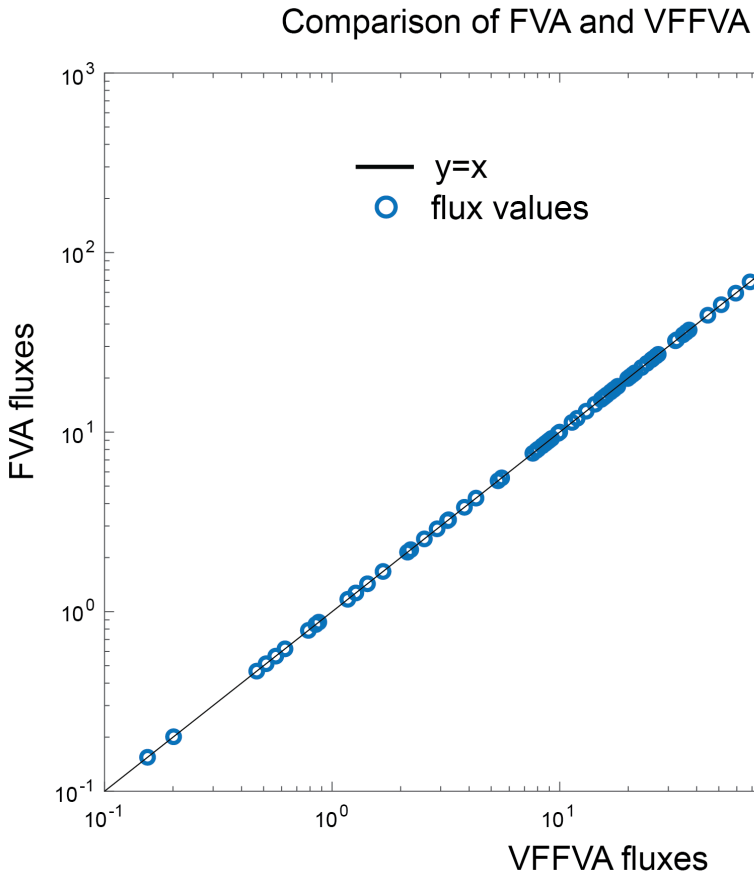
(continues on next page)

(continued from previous page)

```
ecoli=model;
[minFluxVFFVA,maxFluxVFFVA]=VFFVA(nCores, nThreads, ecoli);
```

- Compare the results

```
%Using a log-log scale
figure;
loglog(abs([minFluxVFFVA;maxFluxVFFVA]),abs([minFluxFVA;maxFluxFVA]),'o')
hold on;
loglog([0.1 1000],[0.1 1000])
```



As we can see the results lie perfectly on the identity line.

We can further check the largest difference in precision between the two results. Since we are using the same solver, the results are nearly identical.

```
max([minFluxVFFVA;maxFluxVFFVA]-[minFluxFVA;maxFluxFVA])

ans =

   4.9314e-07
```

1.4 Changelog

- : Support for Cv <= d constraints
- : Improve the docs
- : Changelog added to the doc

1.5 License

The software is provided under MIT License.

```
MIT License

Copyright (c) 2018 Ben Guebila Marouen

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

CHAPTER 2

Indices and tables

- genindex
- modindex
- search