webng Documentation

Release 0.0.2

Ali Sinan Saglam

Sep 29, 2022

CONTENTS:

BioNetgen (BNG) is a modelling language for rule-based modelling of complex biological systems. WESTPA is a python package that implements the weighted ensemble sampling scheme which focuses compuational power to sample rare events in stochastic simulations. Models written with BioNetGen langauge (BNGL) can be simulated as ODEs or can be simulated stochastically using stochastic simulation algorightm (SSA). This can lead to rare events which are hard to sample and WESTPA can help sample these events.

WEBNG is a command line tool designed to simplify the installation of BNG and WESTPA while also providing a simplepipeline to get a WESTPA simulation setup of a model written in BNGL. The tool also includes some sample analyses that are specifically tailored for BNGL models.

Please see *Quick Start* page to learn how to use webng.

CHAPTER

ONE

QUICK START

1.1 Installation

The suggested python distribution to use is the Anaconda python distribution. After that is installed, you can install webng directly from PyPI using pip

```
pip install webng
```

which will also install WESTPA and BioNetGen python libraries which means you only need to run this command and have your model to run a WESTPA simulation. Please note that Windows is not currently supported.

1.1.1 Common installation issues

If you are using WsL, you will need libncurses5, you can install it with

```
sudo apt-get install libncurses5
```

If the webng dependencies don't automatically install, you can clone the webng repository with

```
git clone https://github.com/ASinanSaglam/webng.git
```

and install the repo locally with

```
cd webng
pip install -r requirements.txt
pip install -e .
```

which should install everything you need.

1.2 Usage

After installation complete you can test to see if it's properly installed with

```
webng -h
```

if this command prints out help, the command line tool is installed.

In order to use the tool, you will need a YAML configuration file. This tool comes with a subcommand that can generate a template YAML configuration file for you with the command

```
webng template -i mymodel.bngl -o mysim.yaml
```

this should write a sample configuration file to `mysim.yaml`. See *Configuration Options* page to learn more about what is contained in this file. Next let's actually make the WESTPA simulation folder with

```
webng setup --opts mysim.yaml
```

this will create a folder that corresponds to `sim_name` in `mysim.yaml` file. You can now initialize the simulation with

```
cd mymodel
./init.sh
```

if this command completes successfuly you are ready to run your WESTPA simulation. You can run the simulation using a single core with

```
w_run --serial
```

or you can use multiple cores with the command

```
w_run --n-workers X
```

where `X` is the number of cores you want to use. In order to extend the simulation further you will have to edit `west.cfg` file, please read WESTPA tutorials to learn how to run and manage these simulations.

See *Analysis* page to learn more about the available analyses in webng.

CHAPTER

TWO

CONFIGURATION OPTIONS

2.1 Simulation setup options

The section that's relevant for the simulation setup should look something like this:

```
binning_options:
   block_size: 10 # Number of trajectories to be processed in blocks
   center_freq: 1 # How frequently do we add new Voronoi centers?
   max_centers: 300 # Maximum number of Voronoi centers to be added
   traj_per_bin: 100 # Number of trajectories per Voronoi center
path_options: # this entire section should be automatically set by the tool
   WESTPA_path: /home/USER/westpa
   bng_path: /home/USER/apps/anaconda3/lib/python3.7/site-packages/bionetgen/bng-linux
   bngl_file: /home/USER/webng/testing/test.bngl
   sim_name: /home/USER/webng/testing/test # you can adjust sim folder here
propagator_options:
   pcoords: # These should match observables in your model
   - Atot
   - Btot
   propagator_type: libRoadRunner # this is the suggested propagator
sampling_options:
   dimensions: 2 # Dimensionality of the WESTPA progress coordinates
   max_iter: 10 # Maximum number of WE iterations
   pcoord_length: 10 # Number of data points per WE iteration
   tau: 100 # Resampling frequency
```

you can change various aspects of the simulation setup in this file. Let's look at each block separately.

2.1.1 Binning

```
binning_options:
   block_size: 10 # Number of trajectories to be processed in blocks
   center_freq: 1 # How frequently do we add new Voronoi centers?
   max_centers: 300 # Maximum number of Voronoi centers to be added
   traj_per_bin: 100 # Number of trajectories per Voronoi center
```

`block_size` refers to how many trajectories will be ran at a time. This is important for multicore runs, try to keep the blocksize an integer multiple of the number of cores you have. `center_freq` refers to how frequently voronoi bins will be placed, in units of WE iterations. `max_centers` is the maximum number of voronoi centers that will be placed. Finally, `traj_per_bin` is the number of trajectories in each voronoi center.

2.1.2 Path Options

```yaml
path_options: # this entire section should be automatically set by the tool
   WESTPA_path: /home/USER/westpa
   bng_path: /home/USER/apps/anaconda3/lib/python3.7/site-packages/bionetgen/bng-linux
   bngl_file: /home/USER/webng/testing/test.bngl
   sim_name: /home/USER/webng/testing/test # you can adjust sim folder here
```

Most of these option should be set automatically if WESTPA and BNG are both python importable. `WESTPA_path` is the path to WESTPA to be used, `bng_path` is the path where BNG2.pl lives. `bngl_file` is the bngl model and `sim_name` is the folder that will be used for the WESTPA setup.

2.1.3 Propagator Options

```yaml
propagator_options:
   pcoords: # These should match observables in your model
   - Atot
   - Btot
   propagator_type: libRoadRunner # this is the suggested propagator
```

`pcoords` is the list progress coordinates to be used for WESTPA and should match the observables in your BNGL model. `propagator_type` is the type of propagator to be used. If available, use libRoadRunner since it's currently significantly more efficient for WESTPA runs. If not, you can select "executable" propagator which uses BNG2.pl in combination with bash scripts for each walker.

2.1.4 Sampling Options

```yaml
sampling_options:
   dimensions: 2 # Dimensionality of the WESTPA progress coordinates
   max_iter: 10 # Maximum number of WE iterations
   pcoord_length: 10 # Number of data points per WE iteration
   tau: 100 # Resampling frequency
```

`dimensions` is the number of dimensions to be used for WESTPA progress coordinates and should match the number of BNGL observables you are using. `max_iter` is the maximum number of WE iterations to be ran (this can be changed later from within the setup). `pcoord_length` is the number of data points each walker will return. `tau` is the length of each BNGL simulation/walker.

2.2 Analysis options

When you first create a setup configuration file like `mysim.yaml`, you will see an analysis section like this

```yaml
analyses:
   enabled: false
   work-path: /home/USER/webng/testing/test/analysis # the folder to run the analysis␣
↪under
   average:
      dimensions: null # you can limit the tool to the first N dimensions
      enabled: false # this needs to be set to true to run the analysis
```

(continues on next page)

(continued from previous page)

```
      first-iter: null # first iteration to start the averaging
      last-iter: null # first iteration to end the averaging
      mapper-iter: null # the iteration to pull the voronoi bin mapper from, last␣
↪iteration by default
      normalize: false # normalizes the distributions
      output: average.png # output file name
      plot-energy: false # plots -ln of probabilies
      plot-opts: # various plotting options like font sizes and line width
         name-font-size: 12
         voronoi-col: 0.75
         voronoi-lw: 1
      plot-voronoi: false # true if you want to plot voronoi centers
      smoothing: 0.5 # the amount of smoothing to apply
   evolution:
      avg_window: null # number of iterations to average for each point in the plot
      dimensions: null # you can limit the tool to the first N dimensions
      enabled: false # this needs to be set to true to run the analysis
      normalize: false # normalizes the distributions
      output: evolution.png # output file name
      plot-energy: false # plots -ln of probabilies
      plot-opts: # various plotting options like font sizes and line width
         name-font-size: 12
```

Let's take a look at individual sections.

```
analyses:
   enabled: false
   work-path: /home/USER/webng/testing/test/analysis # the folder to run the analysis␣
↪under
```

This is upper level analysis block and has a single option called `enabled`. If set to false, none of the analyses will run. Each analysis subsection will have the same `enabled` option to set if that particular analysis will be ran or not. `work-path` is the folder where all analysis will be ran.

2.2.1 Average

```
average:
   dimensions: null # you can limit the tool to the first N dimensions
   enabled: false # this needs to be set to true to run the analysis
   first-iter: null # first iteration to start the averaging
   last-iter: null # first iteration to end the averaging
   mapper-iter: null # the iteration to pull the voronoi bin mapper from, last iteration␣
↪by default
   normalize: false # normalizes the distributions
   output: average.png # output file name
   plot-energy: false # plots -ln of probabilies
   plot-opts: # various plotting options like font sizes and line width
      name-font-size: 12
      voronoi-col: 0.75
      voronoi-lw: 1
   plot-voronoi: false # true if you want to plot voronoi centers
```

(continues on next page)

(continued from previous page)

```
   smoothing: 0.5 # the amount of smoothing to apply
```

This is the block for Average analysis. `dimensions` is normally set to null which makes the tool plot all dimensions. If this is set to `N` the tool will plot the first `N` dimensions. `first-iter` and `last-iter` are the iterations to start and stop the averaging. `mapper-iter` is the iteration to pull the voronoi mapper from, if you don't want the mapper from the final WE iteration. `normalize` can be used to enable normalization of probability distributions before plotting. `output` is the file name for the output and this can be set to a `png` or `pdf` file. `plot-energy` takes the `-ln` of the probabilities before plotting. `plot-voronoi` controls if the voronoi centers are plotted on top of the probability distributions. `smoothing` can be changed to reduce or increase the gaussian smoothing used for probability distributions. `plot-opts` contain some options for plotting. `name-front-size` is the font-size used in plotting. `voronoi-col` is the color to be used for voronoi bins and `voronoi-lw` is the line width for the same lines.

2.2.2 Evolution

```
evolution:
   avg_window: 1 # number of iterations to average for each point in the plot
   dimensions: null # you can limit the tool to the first N dimensions
   enabled: false # this needs to be set to true to run the analysis
   normalize: false # normalizes the distributions
   output: evolution.png # output file name
   plot-energy: false # plots -ln of probabilies
   plot-opts: # various plotting options like font sizes and line width
      name-font-size: 12
```

This is the block for Evolution analysis. `avg_window` the number of iterations to average over for every data point. `dimensions` is normally set to null which makes the tool plot all dimensions. If this is set to `N` the tool will plot the first `N` dimensions. `normalize` can be used to enable normalization of probability distributions before plotting. `output` is the file name for the output and this can be set to a `png` or `pdf` file. `plot-opts` contain some options for plotting. `name-front-size` is the font-size used in plotting.

2.2.3 Cluster

```
cluster:
   assignments: null
   cluster-count: 4
   enabled: true
   first-iter: null
   last-iter: null
   metastable-states-file: null
   normalize: null
   states:
   - coords:
      - - 20.0
      - 4.0
      label: a
   - coords:
      - - 4.0
      - 20.0
      label: b
```

(continues on next page)

(continued from previous page)

```
    symmetrize: null
    transition-matrix: null
```

This is the block for Cluster analysis. `assignments` is the assignment file to be used for clustering. This can be pointed to a assignment file you generated using `w_assign` or, if left null, the tool will attempt to generate an assignment file itself. `states` is where you can define states for `w_assign` if you want the tool to run it for you. `cluster-count` is the number PCCA+ will try to cluster the data into. `first-iter` and `last-iter` are WE iterations to pull the data for clustering. `metastable-states-file` is a python pickle file that contains a dictionary which defined which bin is assigned to which metastable state. `normalize` makes it so that the output text is normalized to percentages. `symmetrize` controls if the transition matrix is made symmetrical or not. `transition-matrix` can point to a binary numpy file where you give the tool a custom transition matrix or, if left null, the tool will generate one for you using the assignment file.

2.2.4 Network

```
network:
    enabled: true
    metastable-states-file: null
    pcca-pickle: null
    state-labels: null
```

This is the block for Network generation. `metastable-states-file` is a python pickle file that contains a dictionary which defined which bin is assigned to which metastable state. `pcca-pickle` is the python pickle object that the cluster analysis generates (or you can use pyGPCCA to generate one yourself). `state-labels` is the labels you want to use for each cluster generated by *Cluster*

CHAPTER

THREE

ANALYSIS

In the `mysim.yaml` file you created you should see an analysis section. To learn more about the available options, please see *Configuration Options* page. The analyses can be ran using the command

```
webng analysis --opts mysim.yaml
```

and webng will run the analyses using the configuration options given by the file.

3.1 Probability distribution analyses

3.1.1 Average analysis

`average` will create a N by N set of plots where N is the number of observables you have in the BNG model after running the appropriate WESTPA tools. The results will be saved in the folder written under `analyses/average/output` (by default it is set to `average.png`). The plot will look like a matrix of plots where the diagonal contains 1D probability distributions of each observable and every off-diagonal will be a 2D probability heatmap of each pair of observables. For various options available, see *Average* page.

3.1.2 Evolution analysis

The analysis `evolution` will make a probability distribution evolution plot for each observable so you can track the progress of your simulation. For various options available, see *Evolution* page.

3.2 Clustering and network analyses

3.2.1 Clustering analysis

`cluster` analysis will allow you to use the transition matrix estimated from your simulation and do PCCA+ clustering. This type of clustering is useful to maximize transition within stable states and minimize transitions between unstable states and in this way it takes kinetics of the system into account. For various options available, see *Cluster* page.

3.2.2 Network generation

`network` analysis will use the results of your clustering and give you `gml` files that can be used to visualize the clusters. This analysis will give you two files, one for the full transition matrix and one for the clustered result. For various options available, see *Network* page.

Warning: This documentation is still a work in progress and is incomplete.

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex
- search