StackGen User Guide

Arunima Ghosh, Rahul Sahu and Sandeep K. Reddy

code version: 16-09-2023

Contents

1 Introduction

StackGen is intended for researchers, practitioners, and students who want to obtain a stable supramolecular stack at a very low computational cost. It is an open-source framework, capable of generating stable one-dimensional supramolecular stacks within a low computational cost. It streamlines the process of identifying stable structures from a vast array of stack configurations that arise from translating and rotating adjacent molecules in different directions. This stack formation is guided by four order parameters: translation along three axes (tx, ty, tz) and rotation around the z-axis (twist). The software is distributed with the GPL-3.0 license, and the source code can be accessed and downloaded from the GitHub repository: https://github.com/sandeepgroup/build_supramolecular_polymers. Particle Swarm Optimization (PSO) algorithm is employed in StackGen workflow to find out the optimum value of these order parameters, and as an ultimate outcome of StackGen, a stable one-dimensional supramolecular structure is generated based on the parameter values returned by PSO. Currently, StackGen has the following main features:

- The framework offers users three distinct options concerning the rotation center and the arrangement of neighbouring molecules within the stack. These options enable users to generate one-dimensional stacks:
  - with the geometric center of the monomer as the pivotal rotation point
  - in a zigzag configuration
  - with a rotation point defined by the user
- The dimension of the search space can be flexibly defined, by selecting one (twist), two (tz and twist), three (tx, tz, and twist), or four (tx, ty, tz, and twist) order parameters.
- Users can set lower and upper bounds for order parameters, thereby confining the potential candidate solutions of the PSO within defined limits.
- Users have the choice to use either fixed hyperparameter values in StackGen or the dynamic optimization of PSO hyperparameters during execution.

2 Installation

2.1 Prerequisites Requirements

Before installing StackGen, ensure that you have the following prerequisites:

- Python 3 or above
- natsort modules for Python

You can install the required Python module using the following command:

```
pip install natsort
```

Note: The rest of the libraries are preinstalled in Python 3 or above.

You will also need the Quantum mechanical simulation software DFTB+. You can download it from the following link: https://dftbplus.org/download

2.2 Downloading StackGen

The source files of StackGen can be downloaded from GitHub. You can clone the public repository using the command provided below:

```
git clone https://github.com/sandeepgroup/build_supramolecular_polymers.git
```

2.3 Environment Setup and Compilation

After downloading StackGen, follow these steps to initialize the environment:

1. In the StackGen folder, edit the `STACKGEN_SRC` variable in the `set_initenv.sh` file to include the path of the StackGen folder as follows:

   ```
   STACKGEN_SRC = '.../path to the folder containing main.py file/'
   ```

   To prevent the need for setting the variable each time, include the aforementioned variable in the `.bashrc` file.

2. Download the DFTB+ software package and either update the `.bashrc` file by adding the path of DFTB+ as an environment variable or add it to the current shell temporarily.

2.4 Running StackGen

StackGen is pre-compiled and designed to run on Linux environments. To run the StackGen code, follow these steps:

1. Update the path for DFTB parameters in the `dftb_in.hsd` file. Use the file in `examples` as a starting point for creating the dftb input file.

2. Enter the parameter values in the `input.user` file located in the `generate_configuration` folder. Example format:

```
tx_lower=0.0
tx_upper=2.0
ty_lower=0.0
ty_upper=2.0
tz_lower=3.5
tz_upper=3.8
twist_lower=120
twist_upper=180
input_struct=input.xyz
atoms=7,45,16
stack_size=3
label=pdi
```

3. Source the `set_initenv.sh` file using the following command:

```
source set_initenv.sh
```

4. Execute the code using the following command:

```
stackgen_run > out
```

5. To clean the temporary files, use the following command:

```
stackgen_clean
```

The optimized stacked configuration will be saved in a file named `generated_`*label_stack_size*`.xyz`, which can be visualized using software such as VMD.

3 Input Script Structure

This section describes how an input script can be formatted and the essential commands for the StackGen workflow. The input script typically consists of five parts:

3.1 Specifying Input Molecular Structure

It's essential to provide a monomer structure and specify three non-linear atom indexes capable of forming a molecular plane. These atom indexes will be employed for geometric transformations during the process of stack generation. The subsequent input parameters establish both the input molecular structure and the structure of the resulting stacked configuration:

- `input_struct`: Input monomer structure (in .xyz format)
- `atoms`: Three non-linear atom indexes of the input molecule
- `stack_size`: Size of the supramolecular stack
- `rot_type`: The center of rotation and arrangement of consecutive molecules

3.2 Setting Lower and Upper Bounds for Order Parameters

Selecting the lower and upper bounds of order parameters is crucial for constraining PSO candidate solutions. The following parameters specify these bounds and the dimension of the search space:

- `tx_lower`, `tx_upper`: Translation along x-axis bounds
- `ty_lower`, `ty_upper`: Translation along y-axis bounds
- `tz_lower`, `tz_upper`: Translation along z-axis bounds
- `twist_lower`, `twist_upper`: Rotation along z-direction bounds
- `dimension`: Dimension of the search space (e.g., one, two, three, four parameters)

The default values of tx, ty, tz, and twist are [0.0, 3.0] Å, [0.0, 3.0] Å, [3.4, 3.6] Å and [10°, 50°] respectively.

3.3 Setting the PSO Parameters

Specify parameters essential for defining PSO particle trajectories:

- `pso_type`: Type of PSO algorithm (Globalbest or Localbest)
- `nparticle_pso`: Swarm size
- `c1`: Cognitive coefficient
- `c2`: Social coefficient
- `w`: Inertia weight
- `k`: Neighbor size for LocalbestPSO
- `p`: Distance computation (L1 or L2 distance)
- `maxiterations`: Maximum iterations for swarm search
- `ftol`: Acceptable relative error for convergence
- `ftol_iter`: Iterations over which relative error is acceptable

3.4 Optimizing Hyperparameters

In StackGen, you can optimize PSO hyperparameters (ω, c1, c2) on the fly:

- `oh_strategy`: Boolean parameter for hyperparameter optimization (True or False)
- `c1_start`, `c1_end`: Bounds of cognitive coefficient
- `c2_start`, `c2_end`: Bounds of social coefficient
- `w_start`, `w_end`: Bounds of inertia weight

3.5 Additional Parameters

- `nproc_dftb`: Number of processors for DFTB+ calculation
- `label`: Label for the file containing the optimized stacked structure

4 Understanding the StackGen Framework

The main objective of the StackGen framework is to develop a workflow capable of obtaining or modeling stable one-dimensional supramolecular stacks with low computational cost. To achieve this, several individual modules have been used to compose the complete environment for stack generation, maintaining readable and well-structured code. The StackGen framework consists of mainly three layers:

1. The first layer collects inputs from the user, checks for the environment for DFTB+, and sets up the PSO parameters.
2. The second layer implements PSO using the Pyswarms module with a revised convergence criterion.
3. The final layer produces an optimized supramolecular stack structure based on the parameter values returned by PSO.

4.1 Code Overview

The central component of the package is the main module. It assigns default values to parameters, collects user inputs and parameters from the input file, and builds the framework by calling methods and other modules.

4.1.1 env_check()

This method checks the environment setup for a DFTB+ simulation by running a shell script named `../utils/check_env.sh`. It also performs DFTB+ calculations on a test system to verify the correctness of the DFTB+ environment.

Returns: A value of 1 to indicate the environment setup is correct.

4.1.2 optimize(objective_func, maxiters, oh_strategy, start_opts, end_opts)

Implements the optimization with time-varying options and custom ending points for each option (c1, c2, ω).

Parameters:

- `objective_func`: Objective function to be optimized
- `maxiters`: Maximum number of iterations for the PSO algorithm
- `oh_strategy`: Dictionary containing strategies for each option (c1, c2, ω)
- `start_opts`: Dictionary containing starting values for each option
- `end_opts`: Dictionary containing ending values for each option

Returns: Best cost and position after optimization

4.1.3 configuration_generate(tx, ty, tz, twist)

Generates a stacked configuration of the given size by translating and twisting the molecule based on the position of each particle present in the swarm. Translation and rotation are carried out using an external Python script named `orient.py`.

Parameters:

- `tx`: Translation along x-axis
- `ty`: Translation along y-axis
- `tz`: Translation along z-axis
- `twist`: Rotation along the z-axis

This function uses a global variable `rotation_type` to define the center
of rotation and arrangement of consecutive molecules in the stack.
It saves the stacked configuration in a file named as `generated_label_size.xyz`.

5 Sample Input File Examples

This section demonstrates how to define the input script with basic and advanced parameters.

5.1 Basic Input with Essential Parameters

```
tx_lower=0.0
tx_upper=3.0
ty_lower=0.0
ty_upper=3.0
tz_lower=3.3
tz_upper=3.7
twist_lower=0.0
twist_upper=180.0
dimension=4
atoms=2,4,6
input_struct=BTA.xyz
stack_size=2
pso_type=Globalbest
nparticle_pso=15
```

In this scenario, the default values (predefined in the code) of other parameters will be used to run the code.

5.2 Input with Custom PSO Parameters and rotation with respect to geometric center of molecule

```
tx_lower=0.0
tx_upper=0.0
ty_lower=0.0
ty_upper=0.0
tz_lower=3.3
tz_upper=3.7
twist_lower=0.0
twist_upper=180.0
dimension=2
rot_type=0
atoms=2,4,6
input_struct=BTA.xyz
stack_size=2
pso_type=Globalbest
```

```
nparticle_pso=15
c1=0.5
c2=0.3
w=0.9
```

5.3 Input for Zigzag Structure with Custom PSO Parameters

```
tx_lower=0.0
tx_upper=0.0
ty_lower=0.0
ty_upper=0.0
tz_lower=3.3
tz_upper=3.7
twist_lower=0.0
twist_upper=180.0
dimension=2
rot_type=1
atoms=2,4,6
input_struct=BTA.xyz
stack_size=2
pso_type=Globalbest
nparticle_pso=15
c1=0.5
c2=0.3
w=0.9
```

5.4 Input with user-defined center of rotation and Custom PSO parameters

```
tx_lower=0.0
tx_upper=0.0
ty_lower=0.0
ty_upper=0.0
tz_lower=3.3
tz_upper=3.7
twist_lower=0.0
twist_upper=180.0
dimension=2
```

```
rot_type=2
atoms=2,4,6
input_struct=BTA.xyz
stack_size=2
pso_type=Globalbest
nparticle_pso=4
c1=0.5
c2=0.3
w=0.9
```

5.5 Input with Advanced Parameters and Hyperparameter Optimization

```
tx_lower=0.0
tx_upper=3.0
ty_lower=0.0
ty_upper=3.0
tz_lower=3.3
tz_upper=3.7
twist_lower=120.0
twist_upper=180.0
dimension=4
rot_type=1
atoms=2,4,6
input_struct=input.xyz
stack_size=2
pso_type=Globalbest
nparticle_pso=4
c1=0.5
c2=0.3
w=0.9
c1_start=2.5
c1_end=0.5
c2_start=0.5
c2_end=2.5
w_start=0.9
w_end=0.4
```

```
oh_strategy=True
```

5.6 Full Input File

```
tx_lower=0.0
tx_upper=3.0
ty_lower=0.0
ty_upper=3.0
tz_lower=3.3
tz_upper=3.7
twist_lower=10.0
twist_upper=180.0
dimension=4
rot_type=1
atoms = 2,4,6
input_struct=input.xyz
stack_size=2
pso_type=Globalbest
nparticle_pso=15
c1=0.5
c2=0.3
w=0.9
c1_start=2.5
c1_end=0.5
c2_start=0.5
c2_end=2.5
w_start=0.9
w_end=0.4
k=6
p=2
oh_strategy=False
maxiterations=200
nproc_dftb=1
label=supramo
f_tol=1
ftol_iter=10
```

6 Output Files

This section demonstrates the details of the output files which the user receives as an outcome of StackGen. To illustrate, consider a case where a user aims to generate a tetramer supramolecular stack using the BTA molecule, employing fixed PSO hyperparameter values. The corresponding input file appears as follows:

```
tx_lower=0.0
tx_upper=0.0
ty_lower=0.0
ty_upper=0.0
tz_lower=3.1
tz_upper=4.0
twist_lower=240.0
twist_upper=360.0
dimension=2
rot_type=0
atoms=2,4,6
input_struct=BTA.xyz
stack_size=4
pso_type=Globalbest
nparticle_pso=15
c1=0.7
c2=1.5
w=0.6
c1_start=0.7
c1_end=0.1
c2_start=1.5
c2_end=2.0
w_start=0.7
w_end=0.4
k=6
p=2
oh_strategy=False
maxiterations=200
label=supramo
```

```
ftol=1
ftol_iter=30
nproc_dftb=2
```

Once the simulation is done, the user will see the following files as output:

- `Generated_supramo_4.xyz`: This file contains the optimized tetramer stacked structure of the BTA monomer. Visualization software such as VMD can be employed to visualize this optimized structure.
- `traj.out`: This file captures the progress of the best fitness value found by the entire swarm at each iteration. Each line in `traj.out` corresponds to an iteration, detailing the best fitness value and the order parameter values (tx, ty, tz, twist) at that iteration.
- `swarm_traj.out`: It records the complete trajectory of the PSO process. It documents the fitness value and the position of swarm particles for each iteration.