pygace Documentation

Release 2018.12.13

Yingxing Cheng

Dec 14, 2018

CONTENTS:

CHAPTER

ONE

PYGACE

1.1 pygace package

1.1.1 Subpackages

pygace.examples package

Subpackages

pygace.examples.hfo2 package

Submodules

pygace.examples.hfo2.hfo2_gace module

A GA-to-CE example of oxygen-vacancy-containing HfO2 system given in this module.

class pygace.examples.hfo2.hfo2_gace.**HFO2App**(*ce_site=8*, *ce_dirname='./data/iter1'*, *ele_1st='O'*, *ele_2nd='Vac'*, *params_config_dict=None*)

Bases: *pygace.gace.AbstractApp*

An app of HfO(2-x) system which is implemented from AbstractApp object

This object is used to execute a GACE simulation, user only need to implement several interfaces to custom their application.

Parameters

ce_site: int the concept of site used in ATAT program.

ce_dirname: str The name of a directory which contain information of MMAPS or MAPS running

ele_1st: str The first type of element in the `site` in `ATAT`.

ele_2nd: str The second type of element in the `site` in `ATAT`.

params_config_dict: dirt Parameter dict used to custom GACE AbstractApp.

Attributes

app [AbstractApp] A subclass object of AbstractApp.

iter_idx [int] Index of GA-to-CE iteration.

DEFAULT_SETUP = {'MU_OXYGEN': -4.91223, 'STEP': 1, 'TMP_DIR': '/home/yxcheng/PycharmPr

evalEnergy(*individual*)
Evaluation function for the ground-state searching problem.

The problem is to determine a configuration of n vacancies on a crystalline structures such that the energy of crystalline structures can obtain minimum value.

Parameters

individual

Returns

float Fittness value

get_epoch(*nb_vac*)
Obtain the epoch state of the previous running

Parameters

nb_vac [int] The number of vacancy

Returns

int The epoch state of previous running.

ind_to_elis(*individual*)
Convert individual (number list) to element list

Parameters

individual

Returns

list A list of element symbol string.

multiple_run(*mission_name*, *repeat_iters*)
For multiple tasks

Parameters

mission_name [str] A string used to represent the name of current running.

repeat_iters [int] How many times should a simulation repeat for statistic error(different grouond-state in different running iterations with identical parameter setting).

Returns

list A list contain the results of running

run(*iter_idx=1*, *target_epoch=50*)
Main function to run a GACE simulation which will be called by *AbstractRunner*.

Parameters

iter_idx [int] Determine which iteration the ECI is used in.

target_epoch [int] Iteration in GA simulation.

Returns

None

single_run(*mission_name*, *repeat_iter*)
A single running task.

Parameters

mission_name [str] A string used to represent the name of current running.

repeat_iters [int] How many times should a simulation repeat for statistic error(different grouond-state in different running iterations with identical parameter setting).

Returns

tuple A tuple of pupulation, random states and hall of fame.

update_ce(*site=8*, *dirname='./data/iter1'*)

Parameters

site [int, optional] The site used in cluster expansion.

dirname [str, optional] The name of directory which contains file required in cluster expansion.

Returns

None

class pygace.examples.hfo2.hfo2_gace.**HfO2EleIndv**(*ele_lis*, *app=None*)

Bases: *pygace.utility.EleIndv*

A class that use list chemistry element to represent individual.

Parameters

ele_lis [list] A list of chemistry element.

app [AbstractApp] An application of GACE which is used to obtain ground-state structures based generic algorithm and cluster expansion method.

Attributes

app: AbstractApp An application handling GACE running process.

ele_lis: list A list of chemistry element string.

ce_energy

Return CE energy

Returns

float Energy predicted by CE method.

ce_energy_corrdump

Return relative energy defined in ATAT and computed by corrdump program.

Returns

float Relative energy generated by corrdump program.

dft_energy(*iters=None*)

Return DFT energy

Parameters

iters [int] index of iteration of GA-to-CE

Returns

None or float

class pygace.examples.hfo2.hfo2_gace.**Runner**(*app=None*, *iter_idx=None*)

Bases: *pygace.gace.AbstractRunner*

A runner for running a GACE simulation.

This object is used to execute a GACE simulation in HfO2 system.

Parameters

app [subclass of HFO2App] A subclass object of HFO2App, default is *None*.

iter_idx [int] Index of GA-to-CE iteration, default is *None*.

Attributes

app [HFO2App] A subclass object of HFO2App.

iter_idx [int] Index of GA-to-CE iteration.

compare_gs(*new_gs*, *old_gs*)

Determine whether current and previous ground-state are identical.

Parameters

new_gs [str] Ground-state configuration predicted by current iteration.

old_gs : Ground-state configuration predicted by previous iteration.

Returns

bool

Raises:

RuntimeError when the number of point defect (oxygen vacancy here) is not equal in two iteration.

print_gs()

Function used to extract ground-state information from pickle file saved during GACE running.

Returns

None

run()

Main function to run.

Returns

None

str2energy(*string*)

Obtain energy from string consists of numbers joined by '_', e.g., `'1_2_3_19_'`, in which the number is the position index in lattice structure template file.

Parameters

string [str] The string consists by index of point defect.

Returns

——-

float CE energy.

Module contents

A GA-to-CE example of oxygen-vacancy containing HfO2 given in this module.

pygace.examples.sto package

Submodules

pygace.examples.sto.sto_gace module

A GA-to-CE example given in this module.

class pygace.examples.sto.sto_gace.**Runner**(*app=None*, *iter_idx=None*)
Bases: *pygace.gace.AbstractRunner*

A runner for running a GACE simulation.

This object is used to execute a GACE simulation in STO system.

Parameters

app [subclass of STOApp] A subclass object of STOApp, default is *None*.

iter_idx [int] Index of GA-to-CE iteration, default is *None*.

Attributes

app [STOApp] A subclass object of HFO2App.

iter_idx [int] Index of GA-to-CE iteration.

create_dir_for_DFT(*task_fname='./DFT_task.dat'*)
Function to create directories for DFT calculation for ground-state configurations candidates in each GA iteration. This function is used to God_view function. The identical functional method in a standard GACE iteration is included *print_gs* member function in STOApp.

Parameters

task_fname [str] The name of file contain the information of directory in which DFT task should be performed.

Returns

None

god_view()
In some cases, the number of all candidate configurations in sample space is limited, and we can enumerate these configurations one by one to calculate CE energis, which is a fast and efficient way to obtain potential ground-state structures than standard genetic algorithms selection.

Returns

None

print_gs()
Function used to extract ground-state information from pickle file saved during GACE running.

Returns

None

run()
Main function to run GA-to-CE iterations.

Returns

None

class pygace.examples.sto.sto_gace.**STOApp**(*ce_site=1*, *ce_dirname='./data/iter0'*, *ele_1st='Ti_sv'*, *ele_2nd='Nb_sv'*, *params_config_dict=None*)

Bases: *pygace.gace.AbstractApp*

An app of SrTi(1-x)Nb(x)O3 system which is implemented from AbstractApp object

This object is used to execute a GACE simulation, user only need to implement several interfaces to custom their application.

Parameters

ce_site: int the concept of site used in ATAT program.

ce_dirname: str The name of a directory which contain information of MMAPS or MAPS running

ele_1st: str The first type of element in the `site` in `ATAT`.

ele_2nd: str The second type of element in the `site` in `ATAT`.

params_config_dict: dirt Parameter dict used to custom GACE AbstractApp.

Attributes

app [AbstractApp] A subclass object of AbstractApp.

iter_idx [int] Index of GA-to-CE iteration.

DEFAULT_SETUP = {'MU_OXYGEN': -4.91223, 'TMP_DIR': '/home/yxcheng/PycharmProjects/pyga

create_dir_for_DFT(*task_fname='./DFT_task.dat'*)

Function to create directories for DFT calculation for ground-state configurations candidates in each GA iteration. This function is used to God_view function. The identical functional method in a standard GACE iteration is included *print_gs* member function in STOApp.

Parameters

task_fname [str] Name of file restoring the directory in which DFT task should be performed.

Returns

None

evalEnergy(*individual*)

Evaluation function for the ground-state searching problem.

The problem is to determine a configuration of n vacancies on a crystalline structures such that the energy of crystalline structures can obtain minimum value.

Parameters

individual

Returns

tuple A tuple contains energy in the first position, which is compatible with `DEAP`.

ind_to_elis(*individual*)

Convert individual (number list) to element list.

Parameters

individual

Returns

list A list of element symbol string.

multiple_run(*ce_iter*, *ga_iters*)
For multiple tasks

Parameters

ce_iter [int] How many times should a simulation repeat for statistic error(different grouond-state in different running iterations with identical parameter setting).

ga_iters [int] How many times should a GA simulation repeat in each GA-to-CE iteration.

Returns

list A list contain the results of running

run(*iter_idx=0*, *target_epoch=50*)
Main function to run a GACE simulation which will be called by *AbstractRunner*.

Parameters

iter_idx [int] Determine which iteration the ECI is used in.

target_epoch [int] Iteration in GA simulation.

Returns

None

single_run(*ce_iter*, *ga_iter*)
A single running task.

Parameters

ce_iter [str] How many times should a simulation repeat for statistic error(different grouond-state in different running iterations with identical parameter setting).

ga_iter [int] How many times should a GA simulation repeat in each GA-to-CE iteration.

Returns

tuple A tuple of pupulation, random states and hall of fame.

update_ce(*site=1*, *dirname='./data/iter0'*)
Function to update inner CE object

Parameters

site [int, optional] The site defined in `lat.in` which is one of input files in ATAT.

dirname [str] The name of directory contain running results of `MMAPS`.

Returns

None

Module contents

A GA-to-CE example of Nb-doped SrTiO3 given in this module.

Module contents

The module contains an example of Nb-doped SrTiO3 with 75 atoms supercell.

In this module, we will introduce two methods for obtaining ground-state structures. One is used in general mode to obtain ground-state with GA-to-CE iteration, the other one can be used in the case with samples of small order of magnitude.

1.1.2 Submodules

1.1.3 pygace.ce module

The module wrapper cluster expansion method (CE) implemented in `MMAPS` in `ATAT`.

class `pygace.ce.`**CE**(*lat_in=None*, *site=16*, *corrdump_cmd=None*, *compare_crystal_cmd=None*)
Bases: `object`

An wrapper for commends in `ATAT`.

This class provides several commands that are commonly used in `ATAT`.

Parameters

lat_in [str] File name of `lat.in` in `ATAT`.

site [int] The site in which different can occupy.

corrdump_cmd [str] Command of `corrdump` in `ATAT`, default is 'corrdump'

compare_crystal_cmd [str] Command of program which is used to determine whether two crystal structures are identical using symmetry analysis.

Attributes

COMPARE_CRYSTAL [str] This string restore a command used to determine whether two configurations are identical in symmetry.

CORRDUMP [str] This string restore the command of *corrdump* in `ATAT`.

clster_info [str] Filename of cluster information, default is `clusters.out` in `ATAT`.

eci_out [str] File name of `eci.out` in `ATAT`.

lat_in [str] File name of *lat.in* in `ATAT`.

per_atom_energy [dict] A dict restore element type and their responding energy defined in `atoms.out` file in `ATAT`

site [int] The site in which different can occupy.

work_path [str] The directory of `MMAPS` or `MAPS` running.

COMPARE_CRYSTAL = None

CORRDUMP = None

static compare_crystal(*str1*, *str2*, *compare_crystal_cmd=None*, ***kwargs*)
To determine whether structures are identical based crystal symmetry analysis. The program used in this package is based on `XXX` library which developed by XXX.

Parameters

str1 [str] The first string used to represent elements .

str2 [str] The second string used to represent elements.

compare_crystal_cmd [str] The program developed to determine whether two crystal structures are identical, default *None*.

kwargs [dict arguments] Other arguments used in *compare_crystal_cmd*.

Returns

bool True for yes and False for no.

corrdump(*cmd*)

Obtain energy predicted by `corrdump` command in `ATAT`.

Parameters

cmd [str] Shell command which call system `corrdump` command of `ATAT`.

Returns

float Energy predicted by `corrdump` command.

fit(*dirname='./.tmp_atat_ce_dir'*)

Obtain all information of a directory in which a correct calculation of CE fitting has been performed.

Parameters

dirname [str] Which directory of the CE running.

Returns

None

get_total_energy(*x*, *is_corrdump=False*, *is_ref=False*, *site_repeat=-1*, *sum_corr=0.0*, *delete_file=True*)

Calculate absolute energy of a crystal structure like first-principles calculation software package computed.

Parameters

x [str] String for filename of lattice crystal, default `str.out`.

is_corrdump [bool] Determine whether function use energy computed by `corrdump` command to replace absolute energy, default *False*.

is_ref [bool] Determine whether function use relative energy provided by users.

site_repeat [int] This variable should be used seriously when a lattice structure cannot map parent lattice.

sum_corr [float] If *is_ref* is *True* this value will be input as energy predicted by `corrdump` command.

delete_file : Whether to delete tmp file generated by program.

Returns

float : Total energy or corrdump energy.

mmaps(*dirname*, *cal=False*, **args*, ***kwargs*)

Call `MMAPS` command in system.

Parameters

dirname [str] Directory name of `MMAPS` command running. Usually, it contains a `lat.in` file, `vasp.wrap` or other wrap file for different first-principles calculation.

cal [bool] Determine whether to run a CE fitting. If *False*, the function will return when clusters information is obtained, and vice versa CE fitting is running until users stop it.

args [position arg] Position arguments for `MMAPS` command.

kwargs [dict arg] Dict arguments for `MMAPS` command.

Returns

None

predict(*x*)

Predict energy by given file name of structure.

Parameters

x [str] 'x' is a name of lattice structure, such as `str.out` in `ATAT`.

Returns

str Energy predicted by corrdump command in `ATAT`

1.1.4 pygace.config module

The module contains config file for pygace running.

1.1.5 pygace.ga module

The module contains genetic algorithms and relevant operator used in GA, e.g., crossover operator, mutation operator.

class pygace.ga.**Individual**(*gene_length=64*, *fitness=0.0*)

Bases: `object`

A wrapper class for individual in GA.

Parameters

gene_length [int] The length of gene.

fitness [float] The fitness value of gene

generate_individual()

Greate a random individual

Returns

None

get_gene(*index*)

Obtain gene in the position of *index*.

Parameters

index [int] The index of position.

Returns

int The gene

set_gene(*index*, *value*)

Set gene in position of *index*.

Parameters

index [int] The index of position.

value [float] The value of gene.

Returns

None

size()
Return the length of individual.

Returns

int The length of individual

valid_type = ['Vac', 'Replace']

pygace.ga.**cycle_crossover**(*ind1*, *ind2*, *cross_number*)
Cycle crossover (CX).

CX algorithm:

- parent1: `[|1 |2 3 |4 |5 6 7 8 |9]`
- parent2: `[|5 |4 6 |9 |2 3 7 8 |1]`
- child1: `[|1 |2 6 |4 |5 3 7 8 |9]`
- child2: `[|5 |4 3 |9 |2 6 7 8 |1]`

Parameters

ind1 [iteration object] The first individual participating in the crossover.

ind2 [iteration object] The second individual participating in the crossover.

cross_number [int] The number of crossover is not used in this algorithm.

Returns

tuple A tuple of two individuals

References

More details can bee seen Ref.[1].

pygace.ga.**gaceCrossover**(*indiv1*, *indiv2*, *crossover_type=1*, *cross_num=8*)
Executes a crossover specified by crossover type *crossover_type'and the number of crossover 'cross_num* on the input *sequence* individuals. The two individuals are modified in place and both keep their original length.

Parameters

indiv1 [Individual object] The first individual participating in the crossover.

indiv2 [Individual object] The second individual participating in the crossover.

crossover_type [int] The type of crossover method:

- `1`: Partially-mapped crossover (PMX)
- `2`: Order Crossover (OX1)
- `3`: Position based crossover (POS)
- `4`: Order based Crossover (OX2)
- `5`: Cycle crossover (CX)

[1] Oliver, I.; Smith, D.; Holland, J. A study of permutation crossover operators on the traveling salesman problem. Proceedings of the 2nd International Conference on Genetic Algorithms, J.J. Grefenstette (ed.). Hillsdale, New Jersey, 1987; pp 224-230.

- 6: Subtour exchange crossover (SXX)

cross_num [int] The number of crossover which determine the number exchange in each crossover operation.

Returns

tuple A tuple of two individuals.

pygace.ga.**gaceGA**(*population*, *toolbox*, *cxpb*, *ngen*, *stats=None*, *halloffame=None*, *verbose=True*, *checkpoint=None*, *freq=10*)

Genetic algorithm (GA) used in pygace. Users can define their algorithms based on DEAP package or other GA framework.

Parameters

population [list] A list represent population consists of all individual.

toolbox [toolbox object] DEAP toolbox object

cxpb [float] The probability of crossover happens.

ngen [int] The number of generations.

stats : The random state of simulation.

halloffame [list] Restored individual.

verbose [bool] Whether to show more message of running.

checkpoint [str] The filename of checkpoint file.

freq [int] The number that determine how many step to write a checkpoint.

Returns

tuple A tuple of population and log file.

pygace.ga.**gaceMutShuffleIndexes**(*individual*, *indpb*)

Shuffle the attributes of the input individual and return the mutant. The *individual* is expected to be a *sequence*. The *indpb* argument is the probability of each attribute to be moved. Usually this mutation is applied on vector of indices.

Parameters

individual [Individual object] Individual to be mutated.

indpb [float] Independent probability for each attribute to be exchanged to another position.

Returns

tuple A tuple of one individual.

pygace.ga.**gaceVarAnd**(*population*, *toolbox*, *cxpb*)

Execute crossover and mutation operation in genetic algorithm running process.

Parameters

population [list] The population of all individual.

toolbox [Toolbox object] The *Toolbox* object defined in DEAP.

cxpb [float] The probability or crossover.

Returns

list The new generation.

pygace.ga.**order_based_crossover**(*ind1*, *ind2*, *cross_number*)

Order based Crossover (OX2) operator selects at random several positions in a parent tour, and the order of the cities in the selected positions of this parent is imposed on the other parent.

OX2 algorithm example:

- parent1 `[1 |2 3 4 |5 |6 7 8 |9]`
- parent2 `[5 |4 6 3 |1 |9 2 7 |8]`
- child1 `[2 |4 5 |3 |1 6 9 |7 |8]`
- child2 `[4 |2 |3 1 |5 |6 |7 9 8]`

Parameters

ind1 [iteration object] The first individual participating in the crossover.

ind2 [iteration object] The second individual participating in the crossover.

cross_number [int] The number of crossover which determine the number exchange in each crossover operation.

Returns

tuple A tuple of two individuals

References

More details about OX2 can be seen Ref.[2].

pygace.ga.**order_crossover**(*ind1*, *ind2*, *cross_number*)

Order crossover (OX1) operator was proposed by Davis (1985). The OX1 exploits a property of the path representation, that the order of cities (not their positions) are important. It constructs an offspring by choosing a subtour of one parent and preserving the relative order of cities of the other parent.

order crossover algorithm example:

- parent1: `[1 2 |3 4 5 6| 7 8 9]`
- parent2: `[5 7 |4 9 1 3| 6 2 8]`
- child1: `[7 9 |3 4 5 6| 1 2 8]`
- child2: `[2 5 |4 9 1 3| 6 7 8]`

Parameters

ind1 [iteration object] The first individual participating in the crossover.

ind2 [iteration object] The second individual participating in the crossover.

cross_number [int] The number of crossover which determine the number exchange in each crossover operation.

Returns

tuple A tuple of two individuals

[2] Syswerda, G. Handbook of Genetic Algorithms 1991, 332-349.

References

More details about OX1 can be seen Ref.[3].

pygace.ga.**partial_mapped_crossover**(*ind1*, *ind2*, *cross_number*)

Partially-mapped crossover (PMX) operator was suggested by Goldberg and Lingle (1985). It passes on ordering and value information from the parent tours to the offspring tours. A portion of one parents's string is mapped onto a portion of the other parent's string and the remaining informatin is exchanged..

The algorithm example:

- parent1: `[1,2,|3,4,5,6|,7,8,9]`
- parent2: `[5,4,|6,9,2,1|,7,8,3]`
- child1: `[3,5,|6,9,2,1|,7,8,4]`
- child2: `[2,9,|3,4,5,6|,7,8,1]`

Parameters

ind1 [iteration object] The first individual participating in the crossover.

ind2 [iteration object] The second individual participating in the crossover.

cross_number [int] The number of crossover which determine the number exchange in each crossover operation.

Returns

tuple A tuple of two individuals

References

More details about PMX can be seen in Ref.[4].

pygace.ga.**position_based_crossover**(*ind1*, *ind2*, *cross_number*)

Position-based crossover (PBC)

PBC algorithm:

- parent1: `[1 |2 3 4 |5 |6 7 8 |9]`
- parent2: `[5 |4 6 4 |1 |9 2 7 |8]`
- child1: `[4 |2 3 1 |5 |6 7 8 |9]`
- child2: `[2 |4 3 5 |1 |9 6 7 |8]`

Parameters

ind1 [iteration object] The first individual participating in the crossover.

ind2 [iteration object] The second individual participating in the crossover.

cross_number [int] The number of crossover which determine the number exchange in each crossover operation.

Returns

[3] Davis, L. Applying Adaptive Algorithms to Epistatic Domains. Proceedings of the 9th International Joint Conference on Artificial Intelligence - Volume 1. San Francisco, CA, USA, 1985; pp 162-164.

[4] Goldberg, D.; Lingle, R.; Alleles, L. the Travelling Salesman Problem. Proceedings of the 1st International Conference on Genetic Algorithms and their Applications, J.J. Grefenstette (ed.). Carneige-Mellon University, Pittsburgh, 1985.

tuple A tuple of two individuals

References

More details can be seen Ref.[5].

`pygace.ga.`**`subtour_exchange_crossover`**(*ind1*, *ind2*, *cross_number*)

Subtour exchange crossover (SXX).

SXX algorithm:

- parent1: `[1 2 3 |4 5 6 7| 8 9]`
- parent2: `[3 |4 9 |7 8 |5 2 1 |6]`
- child1: `[1 2 3 |4 7 5 6| 8 9]`
- child2: `[3 |4 9 |5 8 |6 2 1 |7]`

Parameters

ind1 [iteration object] The first individual participating in the crossover.

ind2 [iteration object] The second individual participating in the crossover.

cross_number [int] The number of crossover is not used in this algorithm.

Returns

tuple A tuple of two individuals

References

More details about SXX can be seen Ref.[6].

`pygace.ga.`**`transfer_from`**(*ind*)

1.1.6 pygace.gace module

GACE framework module

This module provide abstract GACE object used to be implemented by users in their application, and it defines several interface which are called in concreate application.

`class` `pygace.gace.`**`AbstractApp`**(*ce_site=8*, *ce_dirname='./data/iter1'*, *params_config_dict=None*)

Bases: `object`

Abstract application object for `GACE` framework.

AbstractApp initial process needs input parameters of CE simulation and informatin of output directory. Also, the parameters for DFT calculation should also be included in `params_config_dict` for user custom.

Parameters

ce_site [int] The concept of site used in `MAPS` or `MMAPS` in `ATAT` program.

ce_dirname [:obj: str, optional]

[5] Syswerda, G. Handbook of Genetic Algorithms 1991, 332-349.
[6] Yamamura, M.; Ono, T.; Kobayashi, S. Japanese Society for Artificial Intelligence.

A path of directory which contains information after running `MMAPS` or `MAPS`.

params_config_dict [dict, optional] A dict used to update DEFAULT_DICT of AbstractApp object.

Attributes

ce [CE] CE object defined in *ce.CE*.

params_config_dict [dict] Parameters used in to construct CE object and other parameters used in GACE simulation. User can custom this dict for their own needs.

energy_database_fname [str] Filename of file that restore energies for different configurations to accelerate energy-calculation of a energy-unknown configuration.

toolbox [ToolBox] The ToolBox object defined in *deap.tools*.

DEFAULT_SETUP [dict] Class attribute which restores revelant parameters used in GA and CE simulation process. See also *params_config_dict* for custom.

ENERGY_DICAT [dict] A dict in which key is list of num representing a configuration and value is the fitness value of the configuration, e.g., total energy or formation energy of point defects.

PREVIOUS_COUNT [int] A parameter used to restore the execution step of previous simulation in order to run from previous stop step.

TYPES_ENERGY_DICT [dict] A dict restores different elements and their responding number index in order to convert a element to a number in GA simulation, e.g., {'Hf':1, 'O':2, 'Vac':3}.

TEMPLATE_FILE_STR [str] A string to restore the template of *lat.in* which is a main input file in `ATAT`.

DEFAULT_SETUP = {'TMP_DIR': '/home/yxcheng/PycharmProjects/pygace/doc/tmp_dir', 'NB_SI'

evalEnergy(*individual*)

get_ce()
obtain inner ce object

Returns

CE object

get_energy_info_from_database()
Initial energy database

Returns

None

ind_to_elis(*individual*)
Convert a object used in GA to a object used in `ATAT`.

This method is used to convert a list which contains number to a list containing chemistry element, e.g., [2,2,1,3] to ['Hf', 'Hf', 'O', 'Vac']

Parameters

individual: list Convert a list of `int` to a list of chemistry element.

Returns

None

Raises

NotImplementedError This method must be implemented in subclass.

initial()
Initialization for GA simulation.

Returns

toolbox [Toolbox] A Toolbox object contains responding parameters used in GA.

run(*iter_idx=1*, *target_epoch=0*)

Parameters

iter_idx [int] The index of GA-to-CE iteration, in which a DFT calculation is usually executed for update `eci.out` file in `ATAT`.

target_epoch [int] The repeat times of identical simulation of GA, for which the results of GA simulation is relevant with random number, thus a different GA simulation maybe select a different ground-state configuration. This is useful especially in complex system with substantial `sites` to substitute for different configurations.

Returns

None

Raises

NotImplementedError If this method is not implemented, this type error would be raised.

set_dir()
Initial directory.

Returns

None

transver_to_struct(*element_lis*)
Convert element list to *ATAT str.out* file

The chemistry symbol in `element_lis` would be substituted in `str.out` file in `ATAT`.

Parameters

element_lis [list] a list of chemistry symbol, e.g. ['Hf', 'Hf', 'O']

test_param1 [int] the first test parameter

Returns

str filename of `ATAT` structure file, default *str.out*

update_ce(*site=1*, *dirname=None*)
Update inner CE object.

The parameters should contained the `site` information in `MMAPS` and a path of directory containing output file after a CE fitting.

Parameters

site: :obj: 'int', optional The number of `site` in a crystal structure, which does not contain a specific element instead of a site used to restore different type of atoms to simulate alloy configurations in `ATAT`, more detail see `lat.in` file in `ATAT`.

dirname: :obj: 'str', optional

Returns

None

class pygace.gace.**AbstractRunner**(*app=None*, *iter_idx=None*)
Bases: object

Abstract Runner for running a GACE simulation.

This object is used to execute a GACE simulation, user only need to implement several interfaces to custom their application.

Parameters

app [subclass of AbstractApp] A subclass object of AbstractApp, default is *None*.

iter_idx [int] Index of GA-to-CE iteration, default is *None*.

Raises

NotImplementedError If *run()* or *print_gs()* method is not implemented by subclass of *AbstractRunner*, this type of error would be raised.

Attributes

app [AbstractApp] A subclass object of AbstractApp.

iter_idx [int] Index of GA-to-CE iteration.

app

iter_idx

print_gs()
Function used to check ground-state configurations, to obtain their formation energy predicted by CE, and to determine whether a DFT calculation is needed to executed for next GA-to-CE iteration.

Returns

None

Raises

NotImplementedError if this function is not implemented in their subclass, this type error would be raised.

run()
Main runction for running GACE simulation.

Returns

None

Raises

NotImplementedError if this function is not implemented in their subclass, this type error would be raised.

1.1.7 pygace.utility module

There are some general helper function defined in this module.

class pygace.utility.**EleIndv**(*ele_lis*, *app=None*)
Bases: object

A class that use list chemistry element to represent individual.

Parameters

ele_lis [list] A list of chemistry element.

app [AbstractApp] An application of GACE which is used to obtain ground-state structures based generic algorithm and cluster expansion method.

Attributes

app: AbstractApp An application handling GACE running process.

ele_lis: list A list of chemistry element string.

ce_energy
The absolute energy predicted by CE.

Returns

float CE absolute energy.

ce_energy_ref
The relative energy predicted by CE.

Returns

float CE relative energy

ce_object

dft_energy(*iters=None*)
The DFT energy of individual represented by element list.

Parameters

iters [int] Specific which iteration DFT energy are computed.

Returns

float or None If the directory of DFT calculated exists and the calculation has been finished the DFT energy will be return, or a new DFT calculation directory will be created and first-principles calculation should be performed in this directory.

is_correct()
Determine whether the dft energy and the ce energy of indv equivalent are identical within error.

Returns

bool

set_app(*app*)

pygace.utility.**compare_crystal**(*str1*, *str2*, *compare_crystal_cmd='CompareCrystal '*, *str_template=None*, ***kwargs*)
To determine whether structures are identical based crystal symmetry analysis. The program used in this package is based on `XtalComp` library which developed by David C. Lonie.

Parameters

str1 [str] The first string used to represent elements .

str2 [str] The second string used to represent elements.

compare_crystal_cmd [str] The program developed to determine whether two crystal structures are identical, default *CompareCrystal*.

str_template [str] String template for the definition of lattice site.

kwargs [dict arguments] Other arguments used in *compare_crystal_cmd*.

Returns

bool

References

https://github.com/allisonvacanti/XtalComp

pygace.utility.**get_num_lis**(*nb_Nb*, *nb_site*)
Get number list by given the number point defect and site defined in lattice file

Parameters

nb_Nb [the number of point defect]

nb_site [int] The number of site defined in lattice file

Yields

All combinations.

pygace.utility.**reverse_dict**(*d*)
Exchange *key* and *value* of given dict

Parameters

d [dict] A dict needed to be converted.

Returns

Dict The new dict in which *key* and *value* are exchanged with respect to original dict.

pygace.utility.**save_to_pickle**(*f*, *python_obj*)
Save python object in pickle file.

Parameters

f [fileobj] File object to restore python object

python_obj [obj] Object need to be saved.

Returns

None

1.1.8 Module contents

Searching the most stable atomic-structure of a solid with point defects (including the extrinsic alloying/doping elements), is one of the central issues in materials science. Both adequate sampling of the configuration space and the accurate energy evaluation at relatively low cost are demanding for the structure prediction. In this work, we have developed a framework combining genetic algorithm, cluster expansion (CE) method and first-principles calculations, which can effectively locate the ground-state or meta-stable states of the relatively large/complex systems. We employ this framework to search the stable structures of two distinct systems, i.e., oxygen-vacancy-containing HfO(2-x) and the Nb-doped SrTi(1-x)NbxO3 , and more stable structures are found compared with the structures available in the literature. The present framework can be applied to the ground-state search of extensive alloyed/doped materials, which is particularly significant for the design of advanced engineering alloys and semiconductors.

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

p

INDEX