# Adaptive Genetic Algorithm code

# User's Guide

AGA code version 1.0.0

User's Guide, updated Jul. 10, 2014

# Contents

# 1. Adaptive Genetic Algorithm

## 1.1 Overview

Crystal structure prediction is one of the key components in new materials design and discovery and has been one of the long-standing challenges in the physical sciences. In recent years, many approaches have been proposed to solve this problem.

The adaptive genetic algorithm (AGA) is introduced to combine the speed of structure exploration by classical potentials with the accuracy of density functional theory calculations in an adaptive and iterative way. In this scheme, auxiliary classical potentials are employed to explore structural phase space while the low-energy structures are determined based on first-principles total energy calculations.

Parameters of the auxiliary potentials are adaptively adjusted to reproduce first-principles results during the course of the GA search, which allows the system to hop from one basin to another in the energy landscape, leading to efficient sampling of configuration space. While retain the accuracy of DFT, the adaptive GA is much faster than full DFT GA and offers a useful tool to study the structures of complex materials containing large number of atoms.

## 1.2 Method [1]

Crystal structure prediction starting from the chemical composition alone has been one of the long-standing challenges in theoretical solid state physics, chemistry, and materials science [2,3]. Progress in this area has become a pressing issue in the age of computational materials discovery and design. In the past two decades several computational methods have been proposed to tackle this problem. These methods include simulated annealing [4-6], genetic algorithm (GA) [7-14], basin (or minima) hopping [15,16], particle swarm optimization [17,18], and *ab initio* random structure search [19]. While there has been steady progress in predicting crystal structures of elementary crystals, oxides, and binary alloys [9-14,17-19], exploration of complex binary, ternary, and quaternary systems has required more advanced algorithms for configuration space exploration and faster and also reliable methods for energy evaluation. While first-principles density functional theory (DFT) calculations offer accurate total energies, its computational cost imposes the bottleneck to the structure identification of complex materials with unit cells containing $\sim 10^2$ atoms and/or with variable stoichiometries. By contrast, calculations based on classical potentials are fast and applicable to very large systems but are limited in accuracy. For various systems, reliable classical potentials are not even available. Our AGA combines the speed of classical potential searches and the accuracy of DFT calculations. It allows us to investigate crystal structures previously intractable by such methods with current computer capabilities.

The flowchart of the AGA scheme is illustrated in Fig. 1. The traditional (regular) GA loop, i.e. left-hand side of the flowchart, is embedded in an adaptive loop. Optimization of offspring structures in the GA loop are performed using auxiliary classical potentials whose parameters are adjusted to reproduce DFT results obtained only in the adaptive loop. The GA is an optimization strategy inspired by the Darwinian evolutionary process and has been widely adopted for atomistic structure optimization in the last two decades [7-14]. During the GA optimization process, inheritance, mutation, selection, and crossover operations [7-14] are included to produce new structures and select most fit survivors from generation to generation. The most time-consuming step in the traditional GA-loop is the local optimization of new off-springs by DFT calculations. For complex structures, GA search usually iterates over 200 generations to converge. In the AGA scheme this most time-consuming step is performed using auxiliary classical potentials. In the AGA (see Fig. 1), single point DFT calculations are performed on a small set of candidate structures obtained in the GA loop using the auxiliary classical potentials. Energies, forces, and stresses of these structures from first-principles DFT calculations are used to update the parameters of the auxiliary classical potentials by force-matching method with stochastic simulated annealing algorithm as implemented in the *potfit* code [20,21]. Another cycle of GA search is performed using the newly adjusted potentials, followed by the re-adjustment of the potential parameters, and the process is then repeated – an adaptive-GA (AGA) iteration. All first-principles DFT calculations were performed using VASP [22,23] or Quantum-ESPRESSO [24], which has been interfaced with the adaptive-GA scheme in a fully parallel manner.

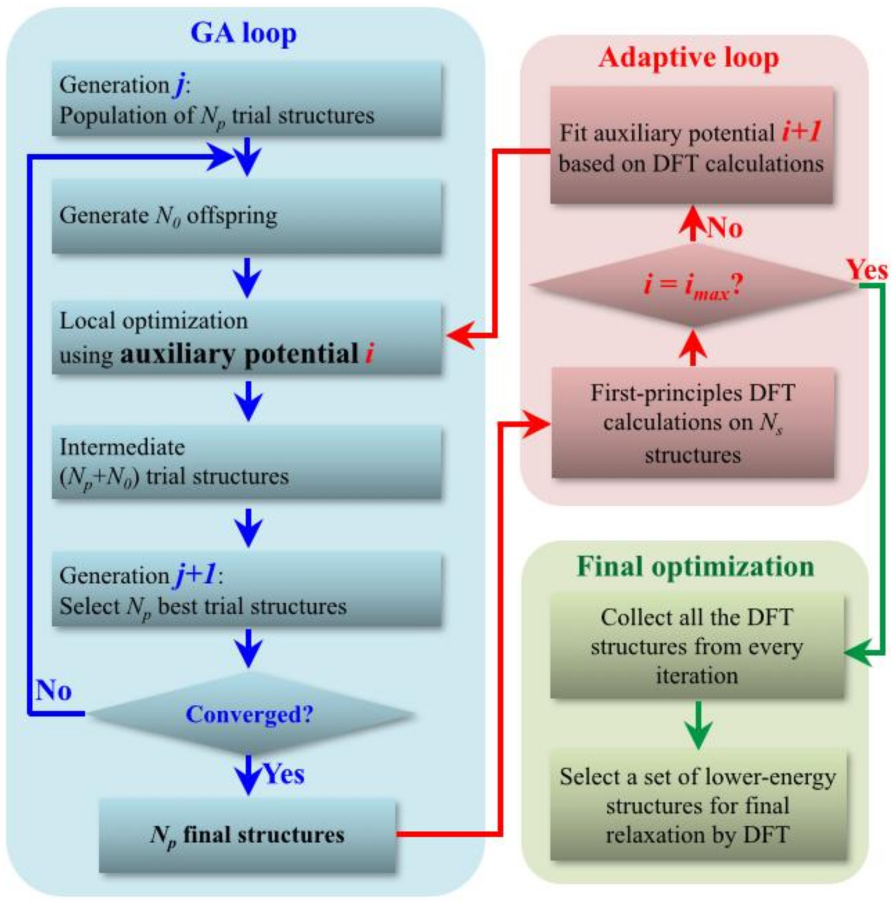

**Fig. 1.** Flowchart of the adaptive genetic algorithm.

The numbers of parent, $N_p$, and off-spring structures, $N_o$, depend on the complexity of the system investigated. For those investigated here, $60 < N_o < 200$, and the total number of structures optimized in each GA-cycle varied between ~12,000 to ~40,000. The use of classical auxiliary potentials for such structure relaxations reduced the computational load by approximately five to six orders of magnitude. It usually takes 30-50 adaptive-GA-iterations to obtain the final structures and the net computational time of the entire adaptive-GA search can be reduced by more than three orders of magnitude. Since the classical potentials are adjusted according to DFT results, the adaptive-GA can explore configuration space more effectively. Structures collected over all adaptive-GA iterations and a set of low-energy metastable structures can be finally screened to locate the ground-state crystal structure. Therefore, the adaptive-GA can essentially search for structures almost with the efficiency of classical potentials but with DFT accuracy.

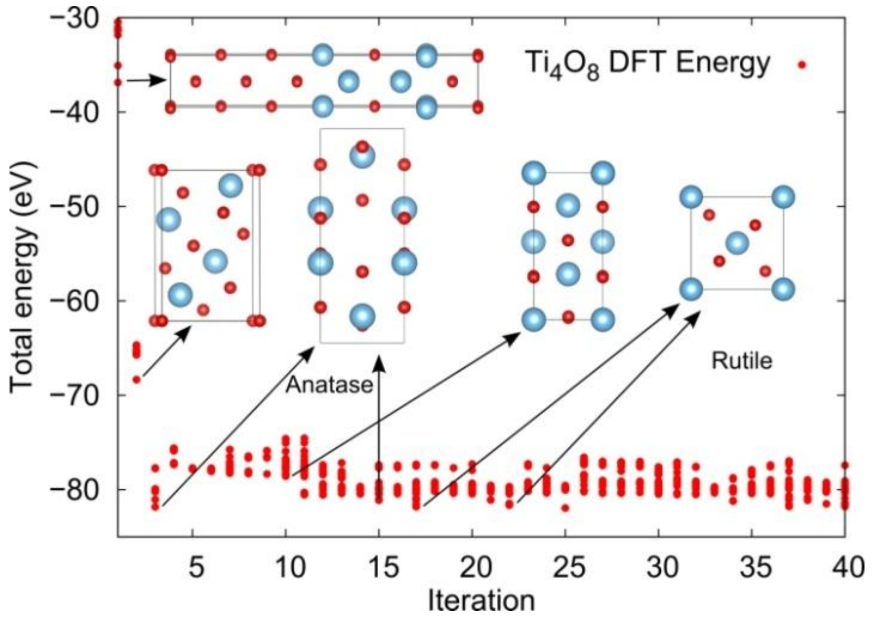

**Fig. 2.** Structural and energetic evolution of $TiO_2$ versus iteration number of the AGA-loop. EAM-type potentials were used in inner GA-loop. Plots show only DFT energies obtained at the end of each AGA-iteration.

We note that the commonly adopted approach of combining classical potentials with DFT calculations for structure optimization involves the use of a *single* set of classical potentials to screen *all* candidate structures followed by a refinement using DFT calculations. This requires accurate and transferable classical potentials able to capture the very-lowest (or few low) energy structures in a complex energy landscape. In contrast, from the energies of the final structures at each iteration as plotted in Fig 2, we can see that the adaptive-GA uses different adjusted potentials to sample structures located in different basins of the energy landscape. Each auxiliary classical potential may not just sample the structures in the same basin, it can sample the structures in a subset of the basins in the energy landscape and some of the basins may overlap with those from other potentials. *Therefore adaptive GA* *is not designed to fit transferable potentials for general atomistic simulations.* It is very difficult or even impossible to fit a classical potential able to accurately describe a system under various bonding environments, especially for binary and ternary systems. However, *it is possible to adjust auxiliary potentials to describe structures located within different subset of basins in the energy landscape with DFT accuracy*. Adapted auxiliary potentials adjusted throughout the adaptive-GA iterations help the system to hop between basins and ensure efficient and accurate sampling of configuration space. An illustration of this point is the search for the crystal structure of $TiO_2$ with Embedded-atom method (EAM) type potentials. We do not expect EAM potentials to describe well the energies of various $TiO_2$ polymorphs. However, as seen in Fig. 2, the adaptive-GA search for structures with 4 formula units (12 atoms per unit cell) was able to find the two low-energy structures of $TiO_2$, *i.e.*, the rutile [25] and the anatase [26] structures – both with 6 atoms per primitive cell only – within 25 adaptive-GA-iterations.

# 2. AGA code

## 2.1 What AGA can do

Our AGA code/method can be used for the crystal structure prediction of complex materials starting from the chemical composition, such as the 3D (Crystal materials), 2D (Surfaces, Grain boundaries, Planar materials), 0D (Clusters) systems/cases, and can also be extended to the related systems/issues.

AGA has been successfully applied to study several systems including the oxides [1, 13], intermetallic compounds [1, 27], grain boundary [28], etc. It is not only good for the perfect crystal structure but also suitable for the material systems with defects.

## 2.2 License and Agreement

### Adaptive genetic algorithm code License

### Agreement

A signed End-user License Agreement from the Ames Lab is required to use these codes, related scripts and subroutines. Redistributions of the Adaptive Genetic Algorithm code in source and/or binary forms, with or without modification, are prohibited without a written permission from the Authors.

Please contact aga@ameslab.gov or Dr. Cai-Zhuang Wang (wangcz@ameslab.gov) if you are interested in our AGA code.

Citations of the following references are recommended for any publications of the work using the Adaptive Genetic Algorithm code/method.

S. Q. Wu, M. Ji, C. Z. Wang, M. C. Nguyen, X. Zhao, K. Umemoto, R. M. Wentzcovitch, and K. M. Ho, An adaptive genetic algorithm for crystal structure prediction. J. Phys.: Cond. Matter 26, 035402 (2014).

X. Zhao, M. C. Nguyen, W. Y. Zhang, C. Z. Wang, M. J. Kramer, D. J. Sellmyer, X. Z. Li, F. Zhang, L. Q. Ke, V. P. Antropov, and K. M. Ho, Exploring the Structural Complexity of Intermetallic Compounds by an Adaptive Genetic Algorithm. Phys. Rev. Lett. 112, 045502 (2014).

X. Zhao, Q. Shu, M.C. Nguyen, Y.G. Wang, M. Ji, H.J. Xiang, K.M. Ho, X.G. Gong, C.Z. Wang, Interface Structure Prediction from First-Principles. J. Phys. Chem. C 2014, 118, 9524-9530 (2014).

## 2.3 News

Our adaptive-GA scheme has be interfaced with different DFT codes, e.g. VASP [22,23] and/or Quantum-ESPRESSO [24] etc, in a fully parallel manner. In the first release version, only the interface with VASP is provided.

## 2.4 Installation

To install AGA, please follow 3 steps:

i.) LAMMPS installation

Follow the LAMMPS instruction (or the following) to build it as a static library (*.a file).

Type:

***Make makelib***

***Make –f Makefile.lib foo***

where foo is the machine name.

Note – For the most recent version of LAMMPS, a few packages will not be installed by default. Please make sure that package MEAM (in order to use the modified EAM potential) and RIGID (for the purpose of interface structure prediction) are installed.

To check the package installation status of LAMMPS, type

***make package-status***

After a successful installation, there ought to be a *.a file.

ii.) GA installation

Go to AGA/src/, and type ***make*** to build the GA code.

Go to AGA/tools/, and type ***./compile.sh*** to build the tools.

Note – Links in the *AGA/src/Makefile* and *AGA/tools/compile.sh* should point to the correct directory in order to compile successfully.

iii.) POTFIT installation

Compile the POTFIT following its instruction.

(http://potfit.sourceforge.net/wiki/doku.php?id=compiling)

Note: The function to write LAMMPS potential in the latest version of POTFIT has been reported to have some problem. The developer has made corrections on it. However, if the potential was found to have trouble, please contact us and we can provide a temporary solution to it.

Meanwhile, more analytical functions are defined. Request can be made by contacting us if needed.

To uninstall the GA code, go to AGA/src/ and type: ***make clean***

To uninstall the tools, go to AGA/tools/ and type: ***./compile.sh clean***

## 2.5 Input Files

### 2.5.1 Input files

Required (always) files in the working directory:

*aga.sh* : a bash script that connects every part of the search: GA, VASP, and POTFIT, which is used to run the jobs.

Note – To run the code in a parallel manner, a machine file with the CPU information, named as *nodes* is needed. If your computer system does not write such information, please contact us for an alter solution.

*ga.in* : contains all the general parameters to set up a search. Format:

parameter1 = value1

…

parameterN = valueN

Note – parameter's order does not matter;

"=" must be separated by space from both the parameter's name and value;

unrecognized parameters will be ignored, therefore note can be added anywhere.

Required files for classical potential GA searches:

*pot.in* & *pot.lammps*: classical potential files used while calling LAMMPS; *pot.in* contains the LAMMPS command and *pot.lammps* is the tabulated potential.

If analytical potential forms are supported by LAMMPS, then all the parameters can be put in *pot.in*, in which case, *pot.lammps* can be omitted.

Required files for first-principles GA searches:

*vaspfiles/* : a folder containing the files needed by VASP (INCAR, POTCAR, KPOINTS) to do first-principles calculations.

Required files for AGA searches:

*vaspfiles/* : a folder containing the files needed by VASP (INCAR, POTCAR, KPOINTS) to do first-principles calculations.

*pot.in* & *pot.lammps*: classical potential files used while calling LAMMPS; *pot.in* contains the LAMMPS command and *pot.lammps* is the tabulated potential.

If analytical potential forms are supported by lammps, then all the parameters can be put in *pot.in*, in which case, *pot.lammps* can be omitted.

*potfit.in* : input file for potential fitting; contains the parameters used by POTFIT.

*mypot_start*: input file for potential fitting; contains the classical potential form used in the AGA search and fitted by POTFIT. It must be supported by POTFIT.

(http://potfit.sourceforge.net/wiki/doku.php?id=potential_files)

While studying different systems, additional information might be required.

Required files for the interface or surface problem:

*fixed_atoms.in* : contains the lattice and coordinates of the atoms whose positions are fixed or treated as rigid body.

Format:

```
N1      ! number of atoms above the interface (top atoms)
a1  a2  a3              ! coordinates of a
b1  b2  b3              ! coordinates of b
c1  c2  c3              ! coordinates of c
x   y   z   type1       ! coordinates of atom #1 and its type
…
x   y   z   typeN1      ! coordinates of atom #N1 and its type
N2                      !number of atoms below the interface (bottom atoms)
a1' a2' a3'             ! coordinates of a'
b1' b2' b3'             ! coordinates of b'
c1' c2' c3'             ! coordinates of c'
x   y   z   type1       ! coordinates of atom #1 and its type
…
x   y   z   typeN2      ! coordinates of atom #N2 and its type
```

All positions are in Cartesian coordinates. If there is no top atom (surface problem), set N1 to 0 and followed by the bottom atoms (see example: TiO2-110surface or SrTiO3-GB).

The relative positions of the fixed/rigid atoms and the interface/surface atoms in the z direction are not important. After reading in the atom positions in *fixed_atoms.in*, they will be re-organized as in the model [28].

For more information on setting the types of the atoms, please see the explanation on **ntype** in *ga.in*.

*latt.in* : contains the lattice parameters for the surface or interface region. Typically, ***a***, ***b*** are the same as the lattice in *fixed_atoms.in* (see example: TiO2-110surface or SrTiO3-GB).

Format:

```
scale                   ! scale of lattice vectors
a1   a2   a3            ! coordinates of a
b1   b2   b3            ! coordinates of b
c1   c2   c3            ! coordinates of c
```

The resulted lattice parameters will be ***a***, ***b***, ***c*** multiplied by the scale.

<u>Required files for cluster structure searches:</u>

*latt.in* : contains the box information which should be large enough to avoid the effect of periodic boundary conditions.

Format is the same as explained above.

<u>Optional input files:</u>

*latt.in* : contains the unit cell information that is used to generate the initial population. If the unit cell is fixed throughout the whole search, please set parameter **cellfree** in *ga.in* to 0.

Format is the same as explained above.

*ilmp.in* : to initialize the LAMMPS object.

If classical potential calculation by LAMMPS is involved, *ilmp.in* will be automatically generated by the GA code while starting the run. If *ilmp.in* already exists in the working directory, the code will read the given file, instead of writing out a new one. Customization can be made in *ilmp.in* to initialize LAMMPS as needed.

*pool.in* : initial population. If **restart** in *ga.in* is set to 1, the search will start from the given population stored in *pool.in*. Format of *pool.in* is the same as the output pool file, e.g. *results.pool*.

*pool_vec.in* : the relative positions between the top rigid bulk and bottom fixed bulk of the initial population. It is needed when searching for interface and both **movetop** and **restart** in *ga.in* are set to 1. Format of *pool_vec.in* is the same as the output vector file, e.g. *vec.pool*.

*site.in* & *pos.in* : to initialize the population with given site information.

Together with **fixsite** in *ga.in* being set to 1, fixed site search will be performed, meaning that the search will be performed with fixed atom positions and only the decorations of the sites will be optimized.

The coordinates are Cartesian in *pos.in* and Direct in *site.in*.

Format:

```
N                   ! # of atoms
x   y   z   type1   ! coordinates of atom #1 and its type
…
x   y   z   typeN   ! coordinates of atom #N and its type
```

Note – when the atom type of certain sites is not fixed, please set it to 0. During the search, those sites can by occupied by any possible type. For more information on setting the types of the atoms, please see the explanation on **ntype** in *ga.in*.

*comp.in* : contains the compositions to be run when performing multiple-composition search. For more information, please see the explanation on parameters **maxnatom** and **minnatom** in *ga.in*.

Format:

```
0    4    5                 ! for compositions A4B5
0    2    7                 ! for compositions A2B7
...
```

### 2.5.2 Parameters in the input files

*aga.sh* :

It allows one argument to represent current generation or iteration, so output from previous iterations will be not overwritten while running AGA.

e.g. to start from iteration #5, run: ***./aga.sh      5***

*ga.in* :

<u>Required parameters</u>

**system** : integer; 0=crystal; 1=cluster; 2=interface/surface

**cstation** : integer; number of stations to perform classical potential calculations

**dstation** : integer; number of stations to perform first-principles calculations

e.g. n(CPU) = 16, station = 8, then 8 structures are calculated simultaneously and each structure is calculated by n(CPU)/station = 2 CPU's.

**cpool** : integer; pool size for the classical potential search

**dpool** : integer; pool size for the first-principles GA search, OR, number of structures selected to do first-principles calculation in order to obtain force, energy and stress information for POTFIT (when running AGA)

**cgen** : integer; number of generations for the classical potential search

**dgen** : integer; number of generations for the first-principles GA search, OR, number of iterations for AGA search

Note – If **cgen** = 0, and **dgen** = non-zero, first-principles search will be performed, in which case, the values of **cstation** and **cpool** do not matter;

If **dgen** = 0, and **cgen** = non-zero, classical potential search will be performed, in which case, the values of **dstation** and **dpool** do not matter;

If **dgen** = no-zero and **cgen** = non-zero, AGA search will be performed.

**natoms** : integer; total number of atoms

**ntype** : integer; number of atom species

Note – The first type is reserved for vacancy. In current version of the code, maximum number of 10 species (including the vacancy) can be considered.

Atoms with type larger than 10 are treated as fixed atoms, with real type = type(atom)%10;

Atoms with type larger than 20 are treated as rigid body (for example in the case of interface), with real type = type(atom)%10.

**atomn** : array of integers; number of atoms of each atom species. e.g. while searching for $A_4B_8$, then,

**atomn** = 0 4 8

Note – First element is reserved for vacancy.

Optional parameters

**e1conv** : double; tolerance for the minimum energy change of the population while checking the convergence of search

**e2conv** : double; tolerance for the average energy change of the population while checking the convergence of search

**n1conv** : integer; criteria used to check the convergence of the minimum energy of the population

**n2conv** : integer; criteria used to check the convergence of the average energy of the population

Default: **e1conv** = **e2conv** = 1e-5; **n1conv** = 1000, **n2conv** = 0

Note – A GA search reaches convergence, if the minimum energy of the population remains unchanged ($\Delta E <$ **e1conv**) for **n1conv** generations and at the same time the average energy of the population remains unchanged ($\Delta E <$ **e2conv**) for **n2conv** generations.

By default, the search converges if the minimum energy of the population converges, because **n2conv** = 0.

**maxnatom** : array of integers; represents the maximum number of atoms for each type

**minnatom** : array of integers; represents the minimum number of atoms for each type

Note – The first element of the array is saved for vacancy. If there is no vacancy, then set the first number to zero.

If **maxnatom** = **minnatom**, only one composition will be searched; otherwise, all possible compositions will be searched at the same time, e.g. for a binary A-B system,

**natoms** = 12

**ntype** = 3

**maxnatom** = 0 7 8

**minnatom** = 0 4 5

The compositions $A_4B_8$, $A_5B_7$, $A_6B_6$, $A_7B_5$ will be searched. The two arrays satisfy **minnatom**[1] = **natoms** – **maxnatom**[2], **minnatom**[2] = **natoms** – **maxnatom**[1].

The pool size should be set to an integer*N(composition), so that the pool can be divided into correct groups.

If *comp.in* is given, only those in *comp.in* will be searched, instead of all allowed compositions from the setting of **maxnatom** and **minnatom**.

**pmut** : double; mutation probability

Default = 0.05

**rcut** : double; a critical distance used to generate new structures in which the minimum bond length > 0.75***rcut**; also used to create bond table of structures while comparing the similarity of the structures (cutoff to create the neighbor list = 4***rcut**).

Default = 1.5 Å

**criteria** : double; tolerance used to check the similarity of the structures

Default = **rcut**/8 Å

Note – it is recommended to set **rcut** and **criteria** according to the system to be studied.

**config2use** : integer; number of configurations to be used in POTFIT. If not given or bigger than **dpool**, it will be set to **dpool**, meaning that all the DFT calculated structures are used.

**aenergy** : array of doubles; atom energy for each type of atoms used to calculate cohesive energy.

Default = {0.0} eV

**avolume** : array of doubles; volume occupied by each type of atoms in the initial structures.

Default = {15.0} Å^3

Note – Volume of the unit cell will be calculated as: $V = \sum_i \mathbf{avolume}[i]$

If *latt.in* is given, and **avolume** is non-zero, the volume of the unit cell in *latt.in* will be rescaled. So if lattice parameters in *latt.in* are expected to be fixed, please set **avolume** = $\{0.0\}$.

**amass** : array of doubles; atom mass of each atom species used for running MD. If no MD simulation is involved, its number does not matter.

Default = {10.0}

**maxangle** : double; maximum angle for the unit cells of initial population

Default = 120˚

**minangle** : double; minimum angle for the unit cells of initial population

Default = 60˚

**press** : double; pressure applied during the search

Default = 0 Kbar

**restart** : integer; if 0, start a new search; if non-zero, start from structures in *pool.in*.

Default = 0

**id** : integer; starting id that the structures are labeled

Default = 100000000

**cellf** : double; factor to determine the lattice parameters of the unit cells of initial population, if *latt.in* is not given.

$$\frac{\sqrt[3]{V}}{cellf} < c < \sqrt[3]{V} * cellf;\ \sqrt{V/c}/cellf < b < \sqrt{V/c} * cellf\ ; a = V/b/c$$

Default = 2.0

**recalce** : integer; if non-zero, energies will be re-calculated after reading the initial population from *pool.in*.

Default = 0

**pot** : string; if coulomb potential is used as the classical potential, please set it to “coul”; for most of other classical potential types, it can be omitted.

**cellfree** : integer; ways to update the positions of atoms in classical calculations

If = 0, no cell relaxation

If = 1, isotropic cell relaxation

If = 2, anisotropic cell relaxation

If = 3, all free cell relaxtion (Default)

Note – To achieve different relaxation schemes in first-principles calculations, please refer to the settings in VASP.

**fixsite** : integer; To fix the atom positions during the search, please set it to non-zero.

Default = 0

**nfixed** : two integers;

**nfixed** = N(top fixed atoms)      N(bottom fixed atoms)

Default =    0    0

**movetop** : integer; if 0, the top fixed atoms will not move; if non-zero, the top atoms will be moved as a rigid body.

Default = 0

**bot2interface** : double; indicates the closest distance along z direction between the bottom fixed atoms and z=0 plane.

Default = 0.5 Å

Note – The atoms in the interface/surface region start from z=0.

**vacuum** : double; height of the vacuum added for running interface/surface

Default = 20 Å

**mdsteps** : integer; number of steps for MD run to update positions of atoms

Default = 0

= 5000, If system = 2 and movetop = 1

Note – To move the top fixed atoms as a rigid body while running interface, MD simulation is used.

If **pot** = coul, and **mdsteps** is set to non-zero, MD will be also used while locally relaxing the structures by LAMMPS instead of ‘minimize’.

For input parameters of VASP, POTFIT, please refer to the corresponding manuals.

## 2.6 Output and Data Collection

### 2.6.1 Output files

*ga.out* : Log file of the GA search

*results.pool* : structure information of current population. Format:

e.g. for a system with N atoms per structure and M species

```
Nstr      N(atoms)    Current_ID
ID: id1    energy1    pressure1    n(vac)    n(type1) …   n(typeM)
a[1]      a[2]      a[3]
b[1]      b[2]      b[3]
c[1]      c[2]      c[3]
pos1[1]      pos1[2]      pos1[3]      type1
…
posN[1]      posN[2]      posN[3]      typeN
ID: id2    energy2    pressure2    n(vac)    n(type1) …   n(typeM)
…
```

*results.pool0* : structure information of updated population. 1/4 of the structures in the pool are newly generated, whose energies have not been calculated and set to 9999.0. Format is the same as *results.pool*.

If first-principles GA search is performed, the progress can be monitored from the *log.ga* file.

If AGA search is performed, after iteration *i* is finished, the output files will be stored as *filename_i*.

filename= *ga.out* : the log file from GA of iteration *i*

= *classical_pool* : copied from *results.pool* of iteration *i*

= *dft_pool* : pool of structures extracted from first-principles calculations; the ID in the *dft_pool* corresponds to the structures in *classical_pool*

= *potential* : the classical potential used in the GA search of iteration *i*

= *my_apot* : the analytical potential form used in the GA search of iteration *i*

= *potout* : the log file from POTFIT

= *config* : the configuration file extracted from the first-principles calculations

= *pot.config* : the configurations used in POTFIT. It contains a subgroup of configurations in config (see the explanation on **config2use** in *ga.in*).

Progress of the AGA search can be monitored through the *log.aga* file.

If running interface and **movetop** in *ga.in* is set to 1,

*vec.pool* : the relative positions between the top rigid bulk and bottom fixed bulk of the structures in *results.pool*.

Format:

```
Nstr
ID: id1     v1[1]      v1[2]      v1[3]
…
ID: id     Nstrv     Nstr[1]      vNstr[2]      vNstr[3]
```

*vec.pool0* : the relative positions between the top rigid bulk and bottom fixed bulk of the structures in *results.pool0*. Format is the same as *vec.pool*.

## 2.6.2 Data collection and analysis

For classical potential or first-principles searches, unless multiple runs were performed, data collection would not be relevant; however, the following explanation on data analysis is still applicable.

**To collect the structure results of AGA search**

Run:

```
cp    -r    tools/collect/    work_directory/
cd    collect/
./collect.sh    i(pool)    j(pool)    criteria
```

It will collect the *dft_pool_** from *dft_pool_i* to *dft_pool_j*. "criteria" has the same meaning as the parameter **criteria** in *ga.in*, which is the tolerance to check the similarity of the structures.

Output:

*pool.all* : contains the collected results after labeling the energy of the same structures to 9999.0 and being sorted by energy. How many structures are deleted can be found in *gaout*.

*e_vs_iter.dat* : data of DFT energies vs. iteration

**To analyze the structures**

First, the collected structures need to be fully relaxed by first-principles method. *prep_vasp.sh* in /tools/analyze/ converts a structure pool to POSCAR format for VASP calculations.

```
prep_vasp.sh    Name(pool)    n(structures to convert)
```

Output: *POSCAR_1*, …, *POSCAR_n*

Note – The source code *reformatpool.cpp* in /tools/ is called.

After the n structures are relaxed, put OUTCAR_1 … OUTCAR_n and POSCAR_1 … POSCAR_n together and run *vasp2pool.sh* in /tools/analyze/:

***vasp2pool.sh     n(POSCAR)     n(species)     Name(output pool)***

Output: a structure pool named as "Name(output pool)" in *results.pool* format

Note – The source code *poscar2pool.cpp* in /tools/ is called.

To check the symmetry of the structures in a pool file, run *check_sym.sh* in */tools/analyze/*:

***check_sym.sh     Name(pool)     n(structures to check)     tolerance***

Output: *str_1.cif*, …, *str_n.cif*

Note – To check the symmetries, the FINDSYM toolkit is used.

(http://stokes.byu.edu/iso/isolinux.php)

**Other tools**

*reformatpool.x* : convert the structures in pool file to other format, such as .xsf (to plot by VESTA), poscar (VASP), and findsym input data.

If *fixed_atom.in* is present, it will output the structures including the atoms of the top/bottom bulks.

*pool2xyz.x* : convert the structures in pool file to .xyz format, mainly used for plotting clusters. In the output, the atoms are gathered around the origin.

*selectconfig.x* : select **config2use** (set in *ga.in*) configurations to do POTFIT based on energy order.

*vasp2force* : script from POTFIT package to extract configuration files from VASP results. *vasp2force5* is modified to work with VASP 5.

# 3. Frequently Asked Questions (FAQ)

To be added.

# 4. Contact and Bug Reports

If you are interested in our AGA code, please contact aga@ameslab.gov. We are also grateful for suggestions, reasonable feature requests, bug reports, or general feedback.

# 5. Bibliography

[1] S.Q. Wu, M. Ji, C.Z. Wang, M.C. Nguyen, X. Zhao, K. Umemoto, R.M. Wentzcovitch, and K.M. Ho, *An adaptive genetic algorithm for crystal structure prediction*. J. Phys.: Cond. Matter 26, 035402 (2014).

[2] J. Maddox, *Crystals from First Principles*. Nature 335**,** 201 (1988).

[3] M.E. Eberhart and D.P. Clougherty, *Looking for Design in Materials Design*. Nature Mater. 3, 659-661 (2004).

[4] S. Kirkpatrick, Jr. C.D. Gelatt, and M.P. Vecchi, *Optimization by Simulated Annealing*. Science 220, 671-680 (1983).

[5] L.T. Wille, *Searching Potential Energy Surfaces by Simulated Annealing*. Nature 324, 46-48 (1986).

[6] K. Doll, J.C. Schön, and M. Jansen, *Global Exploration of the Energy Landscape of Solids on the Ab Initio Level*. Phys. Chem. Chem. Phys. 9, 6128-6133 (2007).

[7] D.M. Deaven, and K.M. Ho, *Molecular Geometry Optimization with a Genetic Algorithm*. Phys. Rev. Lett. 75, 288-291 (1995).

[8] S.M. Woodley, P.D. Battle, J.D. Gale, and C.R.A. Catlow, *The Prediction of Inorganic Crystal Structures Using a Genetic Algorithm and Energy Minimisation*. Phys. Chem. Chem. Phys. 1, 2535-2542 (1999).

[9] K.D.M. Harris, R.L. Johnston, and B.M. Kariuki, *The Genetic Algorithm: Foundations and Apllications in Structure Solution from Powder Diffraction Data*. Acta Crystallogr. A54, 632-645 (1998).

[10] S.M. Woodley, *Prediction of Crystal Structures Using Evolutionary Algorithms and Related Techniques*. Struct. Bonding 110, 95-132 (2004).

[11] A.R. Oganov, and C.W. Glass, *Evolutionary Crystal Structure Prediction as a Tool in Materials Design*. J. Phys.: Condens. Matter 20, 064210 (2008).

[12] A.O. Lyakhov, A.R. Oganov, H. Stokes, and Q. Zhu, *New Developments in Evolutionary Structure Prediction Algorithm USPEX*. Comp. Phys. Comm. 184, 1172-1182 (2013).

[13] M. Ji, K. Umemoto, C.Z. Wang, K.M. Ho, and R.M. Wentzcovitch, *Ultrahigh-Pressure Phases of $H_2O$ Ice Predicted Using an Adaptive Genetic Algorithm*. Phys. Rev. B 84, 220105(R) (2011).

[14] S.Q. Wu, K. Umemoto, M. Ji, C.Z. Wang, K.M. Ho, and R.M. Wentzcovitch, *Identification of Post-Pyrite Phase Transitions in $SiO_2$ by a Genetic Algorithm*. Phys. Rev. B 83, 184102 (2011).

[15] D. Wales, and J. Doye, *Global Optimization by Basin-Hopping and the Lowest Energy Structures of Lennard-Jones Clusters Containing up to 110 Atoms*. J. Phys. Chem. A 101, 5111-5116 (1997).

[16] S. Goedecker S, *Minima Hopping: An Efficient Search Method for the Global Minimum of the Potential Energy Surface of Complex Molecular Systems*. J. Chem. Phys. 120, 9911-9917 (2004).

[17] Y. Wang, J. Lv, L. Zhu, and Y. Ma, *Crystal Structure Prediction via Particle-Swarm Optimization*. Phys. Rev. B 82, 094116 (2010).

[18] Y. Wang, J. Lv, L. Zhu, and Y. Ma, *CALYPSO: A Method for Crystal Structure Prediction*. Comp. Phys. Commun. 183, 2063-2070 (2012).

[19] C.J. Pickard, and R.J. Needs, *Ab Initio Random Structure Searching*. J. Phys. Condens. Matter 23, 053201 (2011).

[20] P. Brommer, and F. Gahler, *Potfit: Effective Potentials from Ab-Initio Data*. Modelling Simul. Mater. Sci. Eng. 15, 295-304 (2007).

[21] P. Brommer, and F. Gahler, *Effective Potentials for Quasicrystals from Ab-Initio Data*. Phil. Mag. 86, 753-758 (2006).

[22] G. Kresse, and J. Furthmuller, *Efficient Iterative Schemes for Ab Initio Total-Energy Calculations Using a Plane-Wave Basis Set*. Phys. Rev. B 54, 11169-11186 (1996).

[23] G. Kresse, and J. Furthmuller, *Efficiency of Ab-Initio Total Energy Calculations for Metals and Semiconductors Using a Plane-Wave Basis Set*. Comput. Mater. Sci. 6, 15-50 (1996).

[24] P. Giannozzi *et al*. *QUANTUM ESPRESSO: a Modular and Open-Source Software Project for Quantum Simulations of Materials*. J. Phys.: Condens. Matter 21, 395502 (2009).

[25] D.T. Cromer, and K. Herrington, *The Structures of Anatase and Rutile*. J. Am. Chem. Soc. 77, 4708-4709 (1955).

[26] L. Vegard, *Results of Crystal Analysis*. Philos. Mag. 32, 505-518 (1916).

[27] X. Zhao, M.C. Nguyen, W.Y. Zhang, C.Z. Wang, M.J. Kramer, D.J. Sellmyer, X.Z. Li, F. Zhang, L.Q. Ke, V.P. Antropov, and K. M. Ho, *Exploring the Structural Complexity of Intermetallic Compounds by an Adaptive Genetic Algorithm*. Phys. Rev. Lett. 112, 045502 (2014).

[28] X. Zhao, Q. Shu, M.C. Nguyen, Y.G. Wang, M. Ji, H.J. Xiang, K.M. Ho, X.G. Gong, C.Z. Wang, *Interface Structure Prediction from First-Principles*. J. Phys. Chem. C 2014, 118, 9524-9530 (2014).

# 6. Appendix

## 6.1 Examples

A few examples are provided to help users understand how the AGA scheme works and the meaning of the parameters.

1. *dft-$Fe_7W_6$/* : example of a first-principles search
2. *$Fe_7W_6$/* : example of an AGA search for binary alloy
3. *$TiO_2$/* : example of an AGA search for binary oxide
4. *$B_2Co_3Zr$/* : example of an AGA search for ternary alloy
5. *$TiO_2$-110surface/* : example of an AGA search for surface
6. *$SrTiO_3$-GB/* : example of an AGA search for grain boundary/interface