GARFfield: User's Manual
v1.0

Andres Jaramillo-Botero

California Institute of Technology
http://www.wag.caltech.edu/garffield

March 13, 2014

Contents

Abstract

First-principles based force fields developed from fitting to large quantum mechanical data sets are now the norm in predictive molecular dynamics simulations, as opposed to force fields fitted solely from phenomenological data. In principle, the former allow improved accuracy and transferability over a wider range of molecular compositions, interactions and environmental conditions unexplored by experiments. The trade-off has been force field engines that are functionally complex, with a large number of non-bonded and bonded analytical forms that give rise to enormous parameter search spaces. To address this problem, we have developed GARFfield, a parallel hybrid multi-objective Pareto-optimal parameter development scheme based on genetic algorithms, hill-climbing routines and conjugate-gradient minimization. Currently, GARFfield supports development of ReaxFF reactive force fields, effective core pseudo-potentials for the non-adiabatic electron force field (eFF-ECP), Morse potentials for atomistic and coarse-grain force fields, and COMB force fields. Furthermore, the flexible and open architecture of GARFfield enables cost-efficient optimization of parameters from first-principles data sets for any force field.

1 Introduction to GARFfield

GARFfield's computational core is written in ANSI C with a Message Passing Interface (MPI) for parallel computation support. GARFfield uses a modified PGAPack[1] library to produce a random population of the sub-set of force field parameters chosen by the user for optimization (see Parameter selection block in Fig. 1). This population is evaluated through library calls to the LAMMPS parallel molecular dynamics simulation code[2] based on a training set defined by the user (see Training set block in Fig. 1). The training set may contain different objective functions, and multiple entries per objective function, both of which can be weighted by the user. The referenced labels in the training set entries are used as index keys to the atomic or coarse-grain structure definition files, which are provided by the user or automatically generated by GARFfield in LAMMPS native format according to the force field being trained. Fitness and ranking values are produced, in parallel if desired, for every force field parameter set in the random population evaluated over the entire training set (see Fitness function, GA, and force field engines in Fig. 1). Based on a force field's rank it is either used to evolve new siblings via mutation or crossover operations or discarded. The iterated population improves towards a higher-quality set of force fields, and upon convergence, the best performer is written to $ffield.best$ along with the corresponding training set evaluation results in $trainset.err.best$. A hill-climbing routine may be used periodically to search for lower minima and a conjugate gradient method can be applied at the basin of a solution well to improve convergence speed. The LAMMPS library used by GARFfield is compiled to include the Reax/c[3] and eFF[4] user packages. It can be extended to support other force fields as required. Further details about GARFfield follow.

2 Running GARFfield

The general command line to run GARFfield is:

```
garffield <GeoFile> <ForceFieldFile> <TrainSetFile> <ParamsFile> [Options]
```

Where *GeoFile*, *ForceFieldFile*, *TrainSetFile* and *ParamsFile* correspond to the geometry file, force field parameter file, training set file, and the parameters selection file, respectively. These files can take any name as specified by the user, nevertheless they have a specific format (described in subsection 2).

The optional arguments are:

Where GRGA corresponds to Generational (i.e. all strings) and SSGA to a steady-state replacement strategy, MPE is Mean Percent Error, SIWE is Sum Inverse Weighted Error, RMSE is Root Mean Square Error, NRMSE is the Normalized RMSE, and MSE is Mean Squared Error.

To run GARFfield in single processor mode with the default settings, issue the following command line:

```
garffield <GeoFile> <ForceFieldFile> <TrainSetFile> <ParamsFile>
```

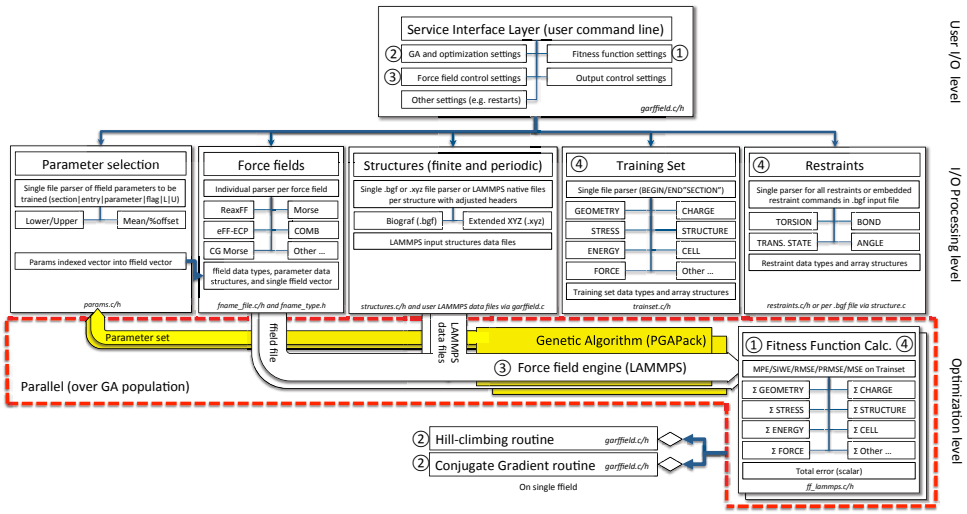

Figure 1: GARFfield's software architecture. Circled numbers indicate connections 1-1, 2-2, 3-3, and 4-4 between functional blocks. Dotted block indicates all calculations performed in parallel, over the population size as *pop*/*procs*.

To run over multiple processors, issue:

```
mpirun -n p garffield <GeoFile> <ForceFieldFile> ...
<TrainSetFile> <ParamsFile> [Options]
```

Where p is the number of processes spawned within the Message Passing Interface (MPI) environment. GARFfield uses a modified version of pgapack [1] that creates an initial random population of force field parameter sets of size *pop* and parallelizes its evaluation using the selected force field type over the number of desired MPI processes, hence, $p \leq pop$. The parallel implementation uses a *master-slave* setup that varies with the number of processes. When two processors are used, both the *master* and the *slave* compute function evaluations. When more than two processors are used, the *master* is responsible for bookkeeping only, and the *slaves* take care of the function evaluations.

1 Setting GARFfield's evolutionary algorithm optimizer

GARFfield uses real-valued strings to represent the force fields, or GA chromosomes. The GA selection process by which it allocates reproductive trials to force fields on the basis of their fitness, is done using the tournament selection method. The crossover operators that take bits from parent force fields and combines them to create sibling force fields is set to two-point crossover at a rate of 0.85. The option *-o* may be used to change the crossover rate. It is set to undergo mutation with a uniform probability (between 0-1) that defaults to the

Option	Arguments	Default	Description
GA			
-i	'range' or 'percent'	'range'	Parameter interval
-s	'maxiter' or 'nochange'	'maxiter'	Convergence criteria
-t	[int]	400	Maximum number of GA iterations
-p	[int]	100	Population size
-g	'GRGA' or or 'eGRGA' or 'SSGA'	'SSGA'	Population replacement strategy
-z	[1-100]	10%	Percent replacement with SSGA
-x	*none*	Mutation only	Perform mutation and crossover operations
-m	[0:1.0]	1/numParams	Mutation rate
-o	[0:1.0]	0.85 (2-point)	Crossover rate
Other			
-C	*threshold tol*	off	Activate CG optimizer on best acc. ffield
-F	reax	Auto-detect	Use Fortran reaxFF engine
-f	*none*	Minimization	Use RMS force instead of minimization
-M	[int]	500	Number of structure minimization steps
-c	*i*	No hillclimb	Hillclimb every i iterations
-e	'MPE' or 'SIWE' or 'RMSE' or 'NRMSE' or 'MSE'	'MPE'	Error function
-d	'full_debug' or 'lammps_debug'	None	Debug verbosity
-r	*j*	10	Report total error every j iterations
-R	*k*	50	Restart every k GA iterations
-u	'real'	electron units (eFF-ECP)	Force units to be real
-v	*none*	off	Use low-gradient vdW correction in reax/c
-W	*none*	off	Randomize objective function weights
-w	*BestForceFieldFile* name	off	Write *BestForceFieldFile* every j iterations
-l	*LogFile* name	Console	Write log to *LogFile*

Table 1: Command line options in GARFfield

reciprocal of the string length, i.e. the number of force field parameters being optimized. If the allele values generated by mutation fall outside of the initial range of a particular gene, then it is reset to the lower value of the initialization range defined in the *ParamsFile*.

2 GARFfield's input files

The following input files are required:

- *GeoFile*. A single ASCII file containing all the molecular structure definition files in sequence, separated by a blank line. Supported formats are biograf (.bgf[**?**]) or extended xyz[5]. Alternatively, the user can provide a local LAMMPS format molecular data structure file for every geometry called within the training file, in which case no *geofile* needs to be provided. The latter is required for the *eff* force field because the explicit electron properties (i.e. radius and spin) are only described in the LAMMPS definition[4]. In this particular case, the user must also specify in each LAMMPS molecular structure file header the atom *number:name* pairs, where *number=1..n*, and any boundary conditions as follows:

  `ECP opt: 1` $\texttt{AtomName}_1$ `2` $\texttt{AtomName}_2$ `...` n $\texttt{AtomName}_n$

  for finite systems, and

  `ECP opt: 1` $\texttt{AtomName}_1$ `2` $\texttt{AtomName}_2$ `...` n $\texttt{AtomName}_n$`: periodic`

  for periodic systems.

- *ForceFieldFile*. Force field parameter file parsers are included for: ReaxFF[6], eFF-ECP[7, 4], COMB[8, 9], and Morse for atomistic and coarse-grain systems. Adding support for other force fields is straightforward, requiring a

corresponding parser (source and header files) and force field header file to define any new data types. The force field engine used to calculate the fitness of each parameter set is automatically detected from the force field file header definition (see below), except in the case of ReaxFF. Currently, LAMMPS[2] is used to compute the different energy/force evaluations in GARFfield. There are two versions of ReaxFF in LAMMPS, a Fortran version and a C version[3], and GARFfield supports both. By default, the C version is used for all reaxFF force field optimizations, but the user can switch to the Fortran version using the option "-F reax" in the command line. For GARFfield to detect either of the ReaxFF implementations, the force field header file needs to start with the word "Reactive".

The eFF-ECP force field format for the s, p and hybrid $s-p$ potential functional forms is specified as:

```
eFF-ECP
n
AtomName_1 ECPtype_1 ECPradius_1 a_1 b_1 c_1 d_1 e_1
AtomName_2 ECPtype_2 ECPradius_2 a_2 b_2 c_2 d_2 e_2
:
AtomName_n ECPtype_n ECPradius_n a_n b_n c_n d_n e_n
```

Where n is an integer corresponding to the number of ECP entries in the force field, AtomName$_i$ corresponds to the name of atom i, ECPtype$_i$ to the type of ECP used (i.e. s, p, h [hybrid] or x [pseudopotential] as described in [7]) for atom i, ECPradius$_i$ to the core radius of atom i, and $a-e$ to the parameters in the corresponding type of ECP. There should be one AtomName definition line per ECP entry in the force field, i.e. $i = 1..n$ in total.

The eFF-ECP force field format for the angular momentum projection operator, s, p, d, or up to f, is specified as:

```
eFF-ECP
n
AtomName_1 s p_1 p_2 p_3 p_4
AtomName_2 p p_1 p_2 p_3 p_4 p_5 p_6
AtomName_3 d p_1 p_2 p_3 p_4 p_5 p_6 p_7 p_8
AtomName_n f p_1 p_2 p_3 p_4 p_5 p_6 p_7 p_8 p_9 p_10
```

Where n is an integer corresponding to the number of ECP entries in the force field, AtomName$_i$ corresponds to the name of atom i, ECPtype$_i$ to the type of ECP used (i.e. with s, p, d, or f angular momentum projections[7]) for atom i, and p_i to the parameters in the corresponding type of ECP; first two are global angular projections and two additional parameters per each level of projection (i.e. 2 for s, 2 for p, 2 for d, and 2 for f). There should be one AtomName definition line per ECP entry in the force field, i.e. $i = 1..n$ in total.

The Morse format as:

```
Morse

n atom types
AtomName_1 AtomMass_1
AtomName_2 AtomMass_2
:
AtomName_n AtomMass_n

m pair types
AtomName_1 AtomName_1 D_0 α r_0
AtomName_1 AtomName_2 D_0 α r_0
:
AtomName_n AtomName_n 0 α r_0
```

Where n and m are a integers ($m \leq 2^n$), AtomName$_i$ corresponds to the name of atom i, and AtomMass$_i$ to the mass of atom i.

- *TrainSetFile*. A diverse set of cases computed from quantum mechanics and used by GARFfield to calculate the fitness of a particular parameter set. This file may contain a single or a combination of objective function sections, in the following format:

```
CHARGE <weight>
#Key weight Atom Charge (a,f,i)
ENDCHARGE
CELL PARAMETERS <weight>
#Key weight a[b,c] Value (a,f,i)
ENDCELL PARAMETERS
FREQUENCIES <weight>
#Key WeighF  WeighM  QM file
ENDFREQUENCIES
HEATFO <weight>
#Key weight Value (a,f,f)
ENDHEATFO
GEOMETRY <weight>
#Key weight Atom_1 Atom_2 Atom_3 Atom_4 Value (a,f,i,i,(i,i,)f)
ENDGEOMETRY
STRUCTURE <weight>
#Key weight Atom_1 Atom_2 Value (a,f,i,i,f)
ENDSTRUCTURE
ENERGY <weight>
#weight op_1 Key_1 / n_1 op_2 Key_2 / n_2 op_3 Key_3 / n_3 op_4 Key_4 / n_4 ...
op_n Key_n / n_n Value (f,(a,a,i),f)
ENDENERGY
```

Where, *Key* corresponds to the structure file name specified in *GeoFile* or LAMMPS data.*Key* files, *weight* is the scalar multiplier used for each section or each entry when computing the weighted sum in the total error, *op* are mathematical + or − operators and *Value* is the literal reference value from quantum mechanical calculations or experiments. The section weight will default to 1 when not entered directly by the user or randomly generated by GARFfield (with the "-W" option).

With the exception of eFF-ECP training, which uses 'electron' units, all others use units of length in Angstroms, units of energy in kcal/mol, units of force in kcal/mol-Angstrom, and units of charge in multiples of electron charge. Electron units are length in Bohr, energies in eV, forces in Hartrees/Bohr, and charges in multiples of electron charge. The user can force eFF-ECP optimization using real units by the command line argument "-u real". This requires all molecular structure files and training set to be defined in the chosen units (i.e. GARFfield will not perform automatic conversion of units for user-specified data).

The letters in parenthesis are meant to indicate the data type for each entry, i.e. i=integer, f=real, a=ASCII, and should not be part of the actual entry. Those within () denote possible repeating entries.

- *ParamsFile.* Determines what parameters will be optimized by GARFfield. The file has a columnar format, with every row entry as follows:

```
Section Entry Parameter 0.0 Low High (i,i,i,f,f,f)
```

Where the Parameter from Entry in Section has an optimization range between Low and High, when the command line option "-i range" is used. Alternatively, when a starting force field with acceptable values is available to the user, such values may be used as a mean from which GARFfield can explore with a ± percent offset, using the command line option "-i percent." The file format must then be given as:

```
Section Entry Parameter 2.0 Mean %Offset (i,i,i,f,f,f)
```

Where *Mean* is taken directly from the *ForceFieldFile* and the % offset is taken from the *ParamsFile*. The first number in column 4 should be $\leq$ 2.0.

- *RestraintFile.* For training reactive force fields it is useful to include information about the reactants, the products, and the transition states. Since GARFfield performs geometry minimization with the selected force field engine, restraints must be applied to avoid structural deviations from the desired reaction pathway/coordinate. These restraints may be applied on relative distances between 2 atoms, angles between 3 atoms and torsions between 4 atoms. The format for defining such restraints in the *RestraintFile* is:

RESTRAINT Key Keyword args

Where,
Key = name of structure file in *GeoFile*
keyword = *BOND* or *ANGLE* or *TORSION* or *TS*
BOND args = $atom_1$ $atom_2$ r_0 K_{start} K_{stop}

$atom_1, atom_2$ = IDs of 2 atoms in bond
K_{start}, K_{stop} = start/end restraint coefficients (energy units)
r_0 = equilibrium bond distance (distance units)

ANGLE args = $atom_1$ $atom_2$ $atom_3$ θ_0 K_{start} K_{stop}

$atom_1, atom_2, atom_3$ = IDs of 3 atoms in angle, $atom_2$ = middle atom
K_{start}, K_{stop} = start/end restraint coefficients (energy units)
θ_0 = equilibrium angle (degrees)

TORSION args = $atom_1$ $atom_2$ $atom_3$ $atom_4$ ϕ_0 K_{start} K_{stop}

$atom_1, atom_2, atom_3, atom_4$ = IDs of 4 atoms in dihedral (linear order)
K_{start}, K_{stop} = start/end restraint coefficients (energy units)
ϕ_0 = equilibrium dihedral angle phi (degrees)

TS args = $body_1$ $atom_2$ $atom_3$ R_{12} K_{start} K_{stop}

$atom_1, atom_2, atom_3$ = IDs of 3 atoms in a transition state
K_{start}, K_{stop} = start/end restraint coefficients (energy units)
R_{12} = ratio of bond lengths, i.e. r_1/r_2 (no units)

The same start/stop force constants in a TS restraint will be applied to the each bond in the transition state. Atom 2 is the center atom, between atoms 1 and 3, r_1 corresponds to the bond length between atoms 1 and 2, and r_2 to the bond length between atoms 2 and 3.

GARFfield will automatically detect and use a *RestraintFile* as its 5^{th} argument in the command line. When using a biograf (.bgf) format *GeoFile* the user can describe the restraints directly in each structure definition via the keyword RESTRAIN, thereby avoiding the need for a separate *RestraintFile*.

An atom in the nomenclature used above for GARFfield can be any of a number of particles, including an actual atom, an electron, or a bead representing a group of particles (e.g. for coarse-grain force fields).

3 GARFfield's output files

When GARFfield starts, it outputs to the console: 1) the default settings and any changes made to these by the user, and the general details of 3) the *Force-*

FieldFile read, 4) the *GeoFile*, 5) the *TrainSetFile*, and 5) the *ParamsFile* before starting the optimization process and outputting the total error per iteration. In addition to this, there are other types of output from GARFfield, including:

- *Force field parameters.* A *ffield.best* file containing the best set of parameters found during the optimization is produced at the end of every run. This file has the same format as the input *ForceFieldFile*. A file *ForceFieldFile.original* corresponding to the starting force field is written at the start of the program, and an intermediate *ffield.new* file corresponding to the current iteration force field file is written every GA iteration. If the option *-w* is used, a *ffield.restart* file corresponding to the best string evaluated up to the last GA reported iteration is also written (the restart frequency is controlled by the *-r* option value).

- *Error information.* A *trainset.err.best* file is written with the error values obtained from the evaluation of *TrainSetFile* with *ffield.best*. A *trainset.err.initial* file is written with the error values obtained from the evaluation of *TrainSetFile* with the *ForceFieldFile.original* file. The files follow the format of the original *TrainSetFile*, except that additional columns are included for the force field values, the calculated weighted error figure for every entry, and the total accumulated weighted error.

- *Debug information.* There are two levels of debug verbosity written by GARFfield to the console (or a *logfile* when using the *-l* command line option), as controlled by the *-d* option in the command line (see Table 2). These are written only for process 0.

- *Restart.* Two restart options are provided in GARFfield: 1) a periodic restart using the *-R* option produces a new population reseeded from the best accumulated force field (i.e. mutated variants of the seed, with mutation probability at 0.5), and 2) the *-w BestForceField* option to periodically write out the best accumulated force field, which can then be used as input to restart an optimization run. Option 2 is useful for large runs that require restarts (e.g. queued systems), and option 1 is recommended when the initial force field corresponds to a reasonably good set of parameters (setting the restart frequency to a large number).

Additional intermediate files are generated for parallel runs; all of which are cleaned on exit.

3 GARFfield's parallel performance

The speedup from using multiple processors will vary with the amount of computation associated with the function evaluation (i.e. the size and complexity of the training set), the ratio between the GA population size (controlled using the *p* option in the command line) and the number of processors available

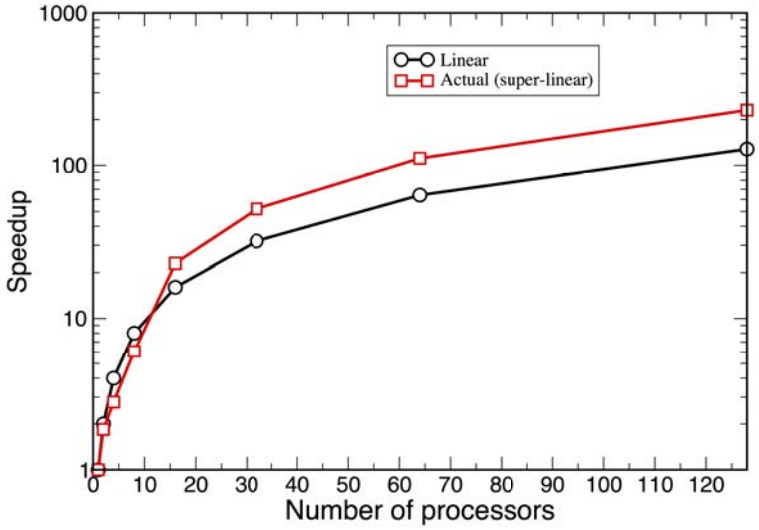

Figure 2: Speedup for ReaxFF SiC parameter set optimization under Los Alamos' National Laboratory supercomputer Mapache.

for the computation, the number of population strings created each GA generation (i.e. the replacement policy), and the communication/synchronization overheads associated with distributing and collecting information to and from the *slaves*. For example, when using two processors, if the population size is 100 and 100 new strings are created each GA generation, then 101 processors can be used effectively to run the (1) master and (100) slaves processes. However, the default setting (SSGA) only replaces 10% of the population each GA iteration, hence only 10 processors can be effectively used to maximize the degree of parallelism. The percent replacement can be modified using the option (-*z*). Depending on the computer architecture used to run GARFfield, significant non-linear speedup can be achieved from the use of cache hierarchies (see **??**).

Bibliography

[1] D. Levine, "Users guide to the pgapack parallel genetic algorithm library." 1996.

[2] S. Plimpton, "Fast parallel algorithms for short-range molecular-dynamics," *Journal of Computational Physics*, vol. 117, no. 1, pp. 1–19, 1995.

[3] H. M. Aktulga, J. C. Fogarty, S. A. Pandit, and A. Y. Grama, "Parallel reactive molecular dynamics: Numerical methods and algorithmic techniques," *Parallel Computing*, vol. 38, no. 4-5, pp. 245–259, 2012.

[4] A. Jaramillo-Botero, J. Su, A. Qi, and W. A. Goddard, "Large-scale, long-term nonadiabatic electron molecular dynamics for describing material properties and phenomena in extreme environments," *Journal of Computational Chemistry*, vol. 32, no. 3, pp. 497–512, 2011.

[5] G. Csanyi, "Extended xya format," 2013.

[6] A. C. T. van Duin, S. Dasgupta, F. Lorant, and W. A. Goddard, "Reaxff: A reactive force field for hydrocarbons," *Journal of Physical Chemistry A*, vol. 105, no. 41, pp. 9396–9409, 2001.

[7] H. Xiao, A. Jaramillo-Botero, P. Theofanis, and W. Goddard, "Non-adiabatic molecular dynamics for 2nd and 3rd row elements of the periodic table," *In preparation for submission to Journal of Phys. Chem.*, 2013.

[8] S. R. Phillpot and S. B. Sinnott, "Simulating multifunctional structures," *Science*, vol. 325, no. 5948, pp. 1634–1635, 2009.

[9] T. Liang, T. R. Shan, Y. T. Cheng, B. D. Devine, M. Noordhoek, Y. Z. Li, Z. Z. Lu, S. R. Phillpot, and S. B. Sinnott, "Classical atomistic simulations of surfaces and heterogeneous interfaces with the charge-optimized many body (comb) potentials," *Materials Science & Engineering R-Reports*, vol. 74, no. 9, pp. 255–279, 2013.