GARFfield: User's Manual
v1.1

Andres Jaramillo-Botero, Ph.D.

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

April 16, 2016

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[1], 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[2],
- Effective core pseudo-potentials for electron force fields (eFF-ECP[3, 4]),
- Morse potentials for atomistic and coarse-grain force fields,
- COMB force fields[5],
- Tersoff force fields[6], and
- EAM force fields for alloys[7].

The flexible and open architecture of GARFfield enables cost-efficient optimization of parameters from first-principles data sets for any force field. Adding a new force field involves integrating the force field parser and only minor changes to the GARFfield core.

1 Introduction to GARFfield

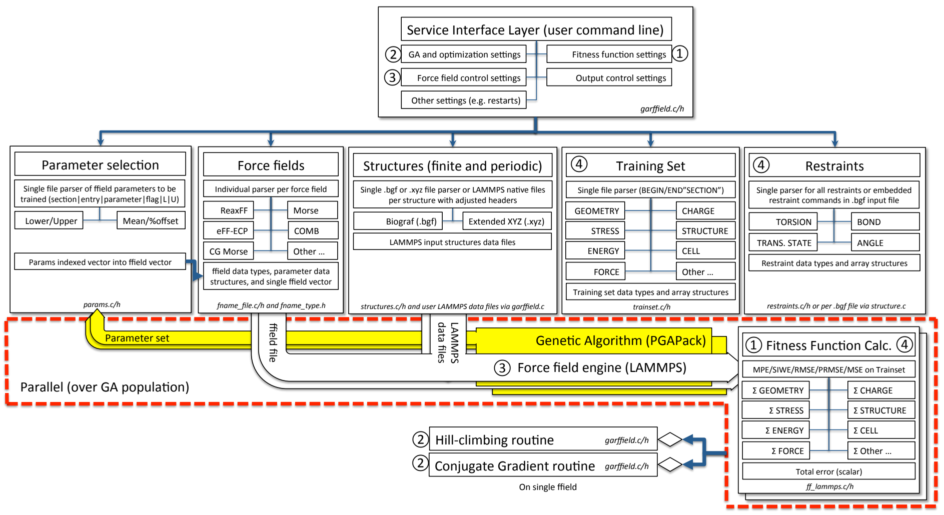
Service Interface Layer (user command line)
2 GA and optimization settings
Fitness function settings 1
3 Force field control settings
Output control settings
Other settings (e.g. restarts)
garffield.c/h
User I/O level
Parameter selection
Single file parser of ffield parameters to be trained (section|entry|parameter|flag|L|U)
Lower/Upper
Mean/%offset
Params indexed vector into ffield vector
params.c/h
Force fields
Individual parser per force field
ReaxFF
Morse
eFF-ECP
COMB
CG Morse
Other ...
ffield data types, parameter data structures, and single ffield vector
fname_file.c/h and fname_type.h
Structures (finite and periodic)
Single .bgf or .xyz file parser or LAMMPS native files per structure with adjusted headers
Biograf (.bgf)
Extended XYZ (.xyz)
LAMMPS input structures data files
structures.c/h and user LAMMPS data files via garffield.c
4 Training Set
Single file parser (BEGIN/END"SECTION")
GEOMETRY
CHARGE
STRESS
STRUCTURE
ENERGY
CELL
FORCE
Other ...
Training set data types and array structures
trainset.c/h
4 Restraints
Single parser for all restraints or embedded restraint commands in .bgf input file
TORSION
BOND
TRANS. STATE
ANGLE
Restraint data types and array structures
restraints.c/h or per .bgf file via structure.c
I/O Processing level
Parameter set
ffield file
LAMMPS data files
Genetic Algorithm (PGAPack)
3 Force field engine (LAMMPS)
Parallel (over GA population)
1 Fitness Function Calc. 4
MPE/SIWE/RMSE/PRMSE/MSE on Trainset
Σ GEOMETRY
Σ CHARGE
Σ STRESS
Σ STRUCTURE
Σ ENERGY
Σ CELL
Σ FORCE
Σ Other ...
Total error (scalar)
ff_lammps.c/h
2 Hill-climbing routine garffield.c/h
2 Conjugate Gradient routine garffield.c/h
On single ffield
Optimization level

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*.

GARFfield's computational core is written in ANSI C with a Message Passing Interface (MPI) for parallel computation support. GARFfield uses a modified PGAPack[8] 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[9] (`http://lammps.sandia.gov`) 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[10] and eFF[11] 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:

Option	**Arguments**	**Default**	**Description**
GA			
-i	'range' or 'percent'	'range'	Parameter interval
-g	'GRGA' or or 'eGRGA' or 'SSGA'	'SSGA'	Population replacement strategy
-m	[0:1.0]	1/numParams	Mutation rate
-o	[0:1.0]	0.85 (2-point)	Crossover rate
-p	[int]	100	Population size
-s	'maxiter' or 'nochange'	'maxiter'	Convergence criteria
-t	[int]	400	Maximum number of GA iterations
-x	*none*	Mutation only	Perform mutation and crossover operations
-z	[1-100]	10%	Percent replacement with SSGA
Other			
-c	i	No hillclimb	Hillclimb every i iterations
-C	[int]	off	Adjust size of mincap array in ReaxFF
-d	'full_debug' or 'lammps_debug'	None	Debug verbosity
-e	'MPE' or 'SIWE' or 'RMSE' or		
-F	reax	Auto-detect	Use Fortran reaxFF engine
-f	*none*	Minimization	Use RMS force instead of minimization
-G	[int]	50	Switch to Conjugate Gradient after in iterations
-l	*LogFile* name	Console	Write log to *LogFile*
-I	[int]	500	Number of structure minimization steps
-M	'fire' or 'sd' or 'cg'	'cg'	Structure minimization method
-e	'NRMSE' or 'MSE'	'MPE'	Error function
-r	j	10	Report total error every j iterations
-R	k	50	Restart every k GA iterations
-S	[double]	1.6	Safezone array allocation factor
-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

Table 1: Command line options in GARFfield

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.

Note that using the low-gradient dispersion correction in ReaxFF force fields (-v option in GARFfield) requires additional parameters (33-34) in the atom section and an additional parameter (7) in the off-diagonal section of ReaxFF force field file.

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

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

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 [8] 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 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) (see `http://www.chem.cmu.edu/courses/09-560/docs/msi/modenv/D_Files.html#944609`) or extended .xyz (see `https://camtools.cam.ac.uk/wiki/site/5b59f819-0806-4a4d-0046-bcad6b9ac70f/extendedxyz.html`.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[11]. 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[2], eFF-ECP[3, 11], COMB[5], Morse for atomistic and coarse-grain systems, Tersoff[6], and EAM[7][1]. 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[9] 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[10], 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
  $\texttt{AtomName}_1$ $\texttt{ECPtype}_1$ $\texttt{ECPradius}_1$ $\texttt{a}_1$ $\texttt{b}_1$ $\texttt{c}_1$ $\texttt{d}_1$ $\texttt{e}_1$
  $\texttt{AtomName}_2$ $\texttt{ECPtype}_2$ $\texttt{ECPradius}_2$ $\texttt{a}_2$ $\texttt{b}_2$ $\texttt{c}_2$ $\texttt{d}_2$ $\texttt{e}_2$
  :
  $\texttt{AtomName}_n$ $\texttt{ECPtype}_n$ $\texttt{ECPradius}_n$ $\texttt{a}_n$ $\texttt{b}_n$ $\texttt{c}_n$ $\texttt{d}_n$ $\texttt{e}_n$

  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 [12]) 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
  $\texttt{AtomName}_1$ s $\texttt{p}_1$ $\texttt{p}_2$ $\texttt{p}_3$ $\texttt{p}_4$

[1]Tersoff and EAM parsers where added by Dr. Anurag Chaudhry

```
AtomName₂ p p₁ p₂ p₃ p₄ p₅ p₆
AtomName₃ d p₁ p₂ p₃ p₄ p₅ p₆ p₇ p₈
AtomNameₙ f p₁ p₂ p₃ p₄ p₅ p₆ p₇ p₈ p₉ p₁₀
```

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[12]) 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
AtomName₁ AtomMass₁
AtomName₂ AtomMass₂
:
AtomNameₙ AtomMassₙ

m pair types
AtomName₁ AtomName₁ D₀ α r₀
AtomName₁ AtomName₂ D₀ α r₀
:
AtomNameₙ AtomNameₙ ₀ α r₀
```

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.

The Tersoff format as:

```
Tersoff

Element₁ Element₂ Element₃ m gamma lambda₃ c d costheta0 n
beta lambda₂ B R D lambda₁ A
```

Where Element$_1$ corresponds to the center atom in a 3-body interaction, Element$_2$ to the the atom bonded to the center atom, and Element$_3$ to the atom influencing the 1-2 bond in a bond-order sense. n, beta, lambda$_3$ (1/distance units), B (energy units), lambda$_2$ (1/distance units), and A (energy units) parameters are only used for two-body interactions. The m, gamma, lambda$_3$, c, d, and costheta0 (can be a value ¡-1 or ¿1) parameters are only used for 3-body interactions. The R (distance units) and D (distance units) parameters are used for both 2-body and 3-body interactions. The non-annotated parameters are unitless. The value of m must be 3 or 1.

For a single-element simulation, only a single entry is required (e.g. Si Si Si). For a two-element simulation, the file must contain 8 entries (for Si Si Si, Si Si C, Si C Si, Si C C, C Si Si, C Si C, C C Si, C C C), that specify Tersoff parameters for all permutations of the two elements interacting in 3-body configurations. Thus for 3 elements, 27 entries would be required, etc. The Tersoff force field parsing code in GARFfield supports multiple elements, however it does not check completeness (this is left to the user). See `http://lammps.sandia.gov/doc/pair_tersoff.html` for more information.

The COMB potential file format corresponds to the one used in LAMMPS (`http://lammps.sandia.gov/doc/pair_comb.html`).

The support for EAM potentials is a very specific implementation developed by Zhou et al[7]. In this flavor of EAM, there are 20 different parameters which are specified in the *ffield* file, as follows:

`Zhou_EAM`

Element$_1$ Element$_2$ Element$_3$ r$_{e2}$ f$_e$ ρ_e ρ_s α β A B κ λ F$_{n0}$ F$_{n1}$ ... F$_{n2}$ Fn$_{n3}$ F$_0$ F$_1$ F$_2$ F$_3$ η F$_e$

Please refer to Table III in Ref.[7] for the names of these parameters and equations A1-A8 for the corresponding analytical forms of embedding energy, density and pair potential functions formed using these 20 elements. The referenced table gives a set of EAM parameters for 16 different metals. Thus the 20 values in the *ffield file* used in GARFfield for the Zhou EAM potential should correspond to those above. At present GARFfield supports only single element parameter optimization.

- *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
STRESS <weight>
#Key weight xx[yy zz xy xz yz] (a,f,a)
ENDSTRESS
PRESSURE <weight>
#Key weight press|xx|yy|zz|xy|xz|yz Value (a,f,a,f)
ENDPRESS
```

```
STSTRAIN <weight>
#Key weight xx Value [yy Value zz Value xy Value xz Value yz Value] (a,f,a,f[a,f])
ENDSTSTRAIN
ATOM_FORCE
#Key weight Atom$_1$ f$_x$, f$_y$, f$_z$ (a,f,i,f,f,f)
ENDATOM_FORCE
CELL PARAMETERS <weight>
#Key weight a[b,c] Value (a,f,a,f)
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.

The `CHARGE` section is used to optimize atomic charge equilibration parameters for reaxFF, including electronegativity, hardness and radius.

The `STRESS` section is used to optimize force field parameters against a periodic unit cell with no stress.

The `PRESSURE` section is used to optimize force field parameters against a periodic unit cell scalar or pressure tensor components.

The `STSTRAIN` section is used to optimize force field parameters against a particular set of stress tensor components, i.e. principal components (xx,yy,zz) and products (xy,xz,yz). Only one weight is used for all components of the tensor.

The `ATOM_FORCE` section is used to optimize force field parameters against atomic forces (fx,fy,fz). This usually improves the accuracy and transferability of a force field, i.e. by adding energy gradients.

The `CELL_PARAMETERS` section is used to optimize force field parameters against periodic unit cell lattice parameters.

The `GEOMETRY` section is used to optimize force field parameters against geometrical features such as bond lengths, angles, and torsions.

The `STRUCTURE` section is currently supported (only) for radial distribution functions (RDF) used in training coarse-grain Morse force fields.

The `ENERGY` section is used to optimize force field parameters against energy *differences*.

- *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
  BOND args = particle_1 particle_2 K_start K_stop r_0

    particlem_1,particle_2 = IDs of 2 particles in bond
    K_start,K_stop = start/end restraint coefficients (energy units)
    r_0 = equilibrium bond distance (distance units)

  ANGLE args = particle_1 particle_2 particle_3 K_start K_stop θ_0

    particle_1,particle_2,particle_3 = 3 particle IDs in angle,
    particle_2 = middle particle
    K_start,K_stop = start/end restraint coefficients (energy units)
    θ_0 = equilibrium angle (degrees)

  TORSION args = particle_1 particle_2 particle_3 particle_4 K_start K_stop φ_0

    particle_1,particle_2,particle_3,particle_4 = 4 particle IDs in dihedral (linear)
    K_start,K_stop = start/end restraint coefficients (energy units)
    φ_0 = equilibrium dihedral angle phi (degrees)
```

GARFfield will automatically detect and use a *RestraintFile* as its 5th 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*.

A particle in the nomenclature used above for GARFfield can be 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 *ForceFieldFile* 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 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

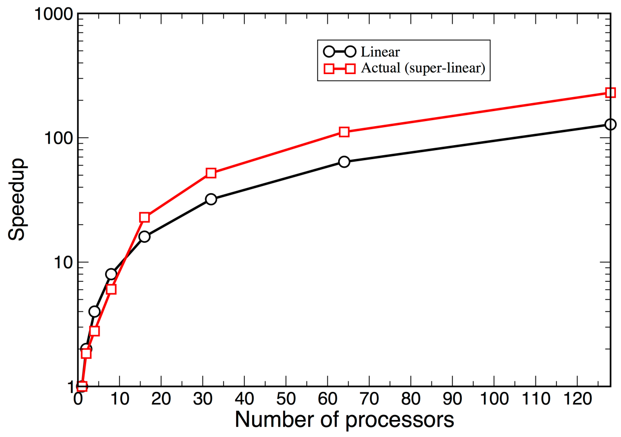

Figure 2: Speedup for ReaxFF SiC parameter set optimization under Los Alamos' National Laboratory supercomputer Mapache.

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 Figure 2).

4 Acknowledgements

Thanks to Dr. Anurag Chaudhry for his force field extensions to support Tersoff and EAM/alloys with GARFfield.

Bibliography

[1] A. Jaramillo-Botero, S. Naserifar, and W. A. Goddard, "General multi-objective force field optimization framework, with application to reactive force fields for silicon carbide," *J. Chem. Theory. and Comput.*, vol. DOI 10.1021/ct5001044, 2014.

[2] A. C. T. van Duin, S. Dasgupta, F. Lorant, and W. A. Goddard, "Reaxff: A reactive force field for hydrocarbons," *J. Phys. Chem. A*, vol. 105, no. 41, pp. 9396–9409, 2001.

[3] H. Xiao, A. Jaramillo-Botero, P. Theofanis, and W. Goddard, "Non-adiabatic dynamics modeling framework for materials in extreme conditions," *Journal Mechanics of Materials*, 2015.

[4] A. Jaramillo-Botero, M. Blanco, and W. Goddard, "The percolation limit near the flory-stockmayer transition in polymer hydrogel networks." 20011.

[5] S. T. C. Y. D. B. D. N. M. L. Y. L. Z. P. S. Liang, T. and S. B. Sinnott, "Classical atomistic simulations of surfaces and heterogeneous interfaces with charge-optimized many body potentials," *Materials Science and Engineering Reports*, 2013.

[6] J. Tersoff, "New empirical approach for the structure and energy of covalent systems," *Phys. Rev. B*, vol. 37, pp. 6991–7000, Apr 1988.

[7] J. R. Zhou, X.W. and H. Wadley, "Misfit-energy-increasing dislocations in vapor-deposited cofe/nife multilayers," *Phys. Rev. B*, 2004.

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

[9] S. Plimpton, "Fast parallel algorithms for short-range molecular-dynamics," *J. Comput. Phys.*, vol. 117, no. 1, pp. 1–19, 1995.

[10] 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.

[11] 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," *J. Comput. Chem.*, vol. 32, no. 3, pp. 497–512, 2011.

[12] 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.