JOURNAL OF CHEMICAL INFORMATION AND MODELING

Application Note

pubs.acs.org/jcim

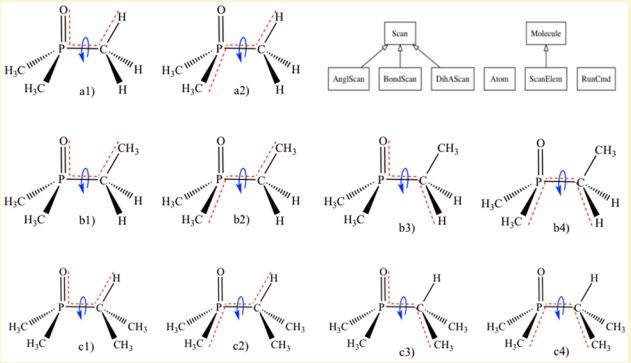
ParFit: A Python-Based Object-Oriented Program for Fitting Molecular Mechanics Parameters to ab Initio Data

Federico Zahariev,* Nuwan De Silva, Mark S. Gordon, Theresa L. Windus, and Marilu Dick-Perez

Department of Chemistry and Ames Laboratory, Iowa State University, Ames, Iowa 50011, United States

Supporting Information

ABSTRACT: A newly created object-oriented program for automating the process of fitting molecular-mechanics parameters to ab initio data, termed ParFit, is presented. ParFit uses a hybrid of deterministic and stochastic genetic algorithms. ParFit can simultaneously handle several molecular-mechanics parameters in multiple molecules and can also apply symmetric and antisymmetric constraints on the optimized parameters. The simultaneous handling of several molecules enhances the transferability of the fitted parameters. ParFit is written in Python, uses a rich set of standard and nonstandard Python libraries, and can be run in parallel on multicore computer systems. As an example, a series of phosphine oxides, important for metal extraction chemistry, are parametrized using ParFit. ParFit is in an open source program available for free on GitHub (https://github.com/fzahari/ParFit).



INTRODUCTION

Molecular mechanics (MM) is a method for approximating the energies and forces of molecular systems by using the laws of classical mechanics.[1−17] Although the laws of quantum mechanics govern the structure and motion of molecular systems, a skillful choice of an MM classical Hamiltonian can reproduce the structural and relative-energy experimental data with reasonable precision. The MM calculations are several orders of magnitude faster than ab initio calculations and are thus able to handle much larger molecular systems or many more molecular systems of the same size. MM also has better computational resource scaling with respect to molecular system size. A straightforward implementation of MM scales as $O(N^2)$, N being the number of atoms in the system, due to the pairwise Coulomb interaction between atoms. However, the use of particle-mesh Ewald methods can reduce the scaling to between $O(N)$ and $O(N^2)$.[18,19] In comparison, the simplest post-Hartree−Fock method that can capture electronic correlation, the second-order Møller−Plesset perturbation method (MP2),[20] scales between $O(n^4)$ and $O(n^5)$, where n is the total number of basis functions, depending on the implementation.

The two components of the MM method are a general functional form of the MM energy and a set of parameters corresponding to different types of interatomic energy components. Popular versions for both are UFF,[21] AMBER,[22−24] OPLS,[25] MM3,[26−28] and MMFF[29] and are, except for UFF, predominantly used for calculations of large organic and biological molecules. However, the lack of availability and/or quality of parameters for organometallic compounds limits the types of systems for which MM can be used.

To obtain parameters for a given atom based MM functional form, one either determines them manually on a trial-and-error basis, guided by experience and intuition, or one uses the least-squares method to fit to experimental[30] or ab initio data.[31] Another appealing possibility is to use the coarse-grained force matching method.[32,33] The manual fitting process is not ideal because it can be time-consuming. A wide range of optimization algorithms has been used to automate the fitting process using the least-squares method. These optimization algorithms generally fall into two categories: deterministic and stochastic algorithms. Within the deterministic category, the Powell,[34] Newton−Raphson,[35] and Nelder−Mead simplex (NM)[36] algorithms are used; and within the stochastic category, the Monte Carlo simulated annealing[34,37] and the genetic algorithm (GA)[38−42] are used for the MM parameter fitting process. ForceBalance is an example of a recent efficient program for MM parameter fitting that is written in Python and based on either deterministic or stochastic algorithms.[43,44] ForceBalance can utilize the energy-gradient, in addition to the energy itself. A hybridization of deterministic and stochastic algorithms was also recently shown to be effective in the implementation of the Paramfit program.[45] For a review on MM parametrization methods, see the Norrby and Brandt review.[46]

Each of the above three optimization strategies has certain advantages and disadvantages. ParFit,[47] presented here, takes

Received: October 26, 2016
Published: February 7, 2017

DOI: 10.1021/acs.jcim.6b00654
J. Chem. Inf. Model. 2017, 57, 391−396

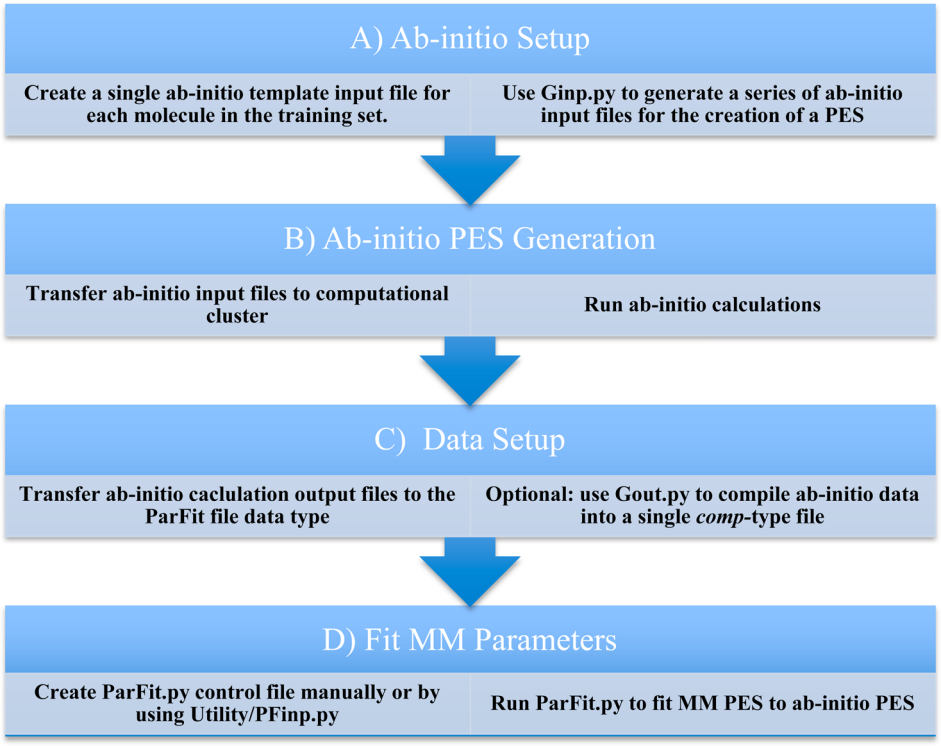


Figure 1. Schematic representation of the ParFit workflow.

advantage of all three strategies: the stochastic GA, the deterministic Powel or NM simplex algorithms, and a sequential hybridization of these two types of algorithms. ParFit is primarily aimed at being a platform for quick prototyping of new MM parameter fitting strategies and additionally having the possibility to be combined with different ab initio and MM programs. To this end, ParFit is written in Python using a flexible object-oriented design and parallelized for an order of magnitude faster performance with respect to serial execution on modern multicore computer systems. In addition, ParFit can apply symmetric and antisymmetric constraints on the optimized MM parameters, i.e. two MM parameters might be constrained to have the same absolute value and either the same (symmetric constraint) or opposite (antisymmetric constraint) sign during the optimization process.

ParFit currently uses GAMESS (General Atomic and Molecular Electronic Structure System)[48,49] or NWChem[50] for ab initio and the SerenaSoft MM Engine for the MM3 and MMFF calculations.[51] ParFit can easily be extended to other ab initio programs, MM force fields, and MM engines. The SerenaSoft MM Engine was chosen as a basis for ParFit, because the MM3 force-field form is among the most accurate ones for small organic molecules.[52]

The applicability and accuracy of the MM method depends, to a large extent, on the transferability of the MM parameters. Transferability of MM parameters means that the parameters obtained by fitting one group of molecules can be successfully used on molecules outside of this initial group. The ability of ParFit to simultaneously handle several molecules during the MM parameter fitting process greatly enhances the transferability of the fitted parameters.

WORKFLOW AND FILE STRUCTURE

The overall MM-parameter fitting process workflow in ParFit can be broken down into four steps outlined in Figure 1. Figure 1A, a fully optimized molecular geometry is stored in a template GAMESS (or NWChem) input file which can be created using a visualization program such as MacMolPl.[53] The *Ginp.py* script uses the geometry from the template input file to generate a series of input files each containing molecular coordinates, where a user specified geometric parameter, such as bond length, bond angle, or dihedral angle, is varied stepwise for a range of values.

Next, consider Figure 1B. The generated input files must be transferred from *~/ParFit/Data/Gamess* (or *~/ParFit/Data/Nwchem*) to a computational cluster and used for a sequence of ab initio calculations. A sample utility file *~/ParFit/Utility/submit.py* illustrates how one may automate the submission process on a computational cluster. Each geometry is then optimized while keeping the specified geometric parameter fixed. After the computations are completed, Figure 1C, the resultant ab initio calculation output files are transferred back to the appropriate directories, *~/ParFit/Data/Gamess* or *~/ParFit/Data/Nwchem*. ParFit uses the energy of the optimized geometries for each fixed geometric parameter to build a potential energy curve. To simplify the fitting process, *Gout.py* may then optionally be used to generate a single file, referred to as a *comp* file in the ParFit program, containing the resulting pertinent information from the series of ab initio calculations.

Lastly, as shown in Figure 1D, a *ParFit.py* control file containing the molecular and parameter information is generated manually or in an interactive session using the standalone utility program *~/ParFit/Utility/PFinp.py*. *ParFit.py*, the fitting control software, is invoked with the generated control file and the fitting process starts. Ginp.py, Gout.py and

DOI: 10.1021/acs.jcim.6b00654
J. Chem. Inf. Model. 2017, 57, 391−396

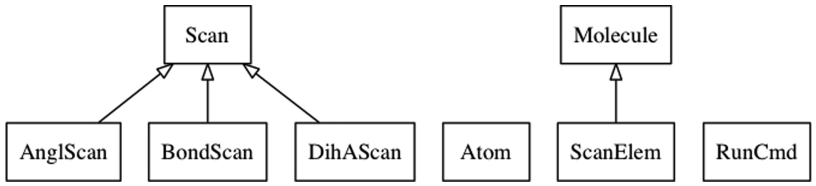


Figure 2. Schematic representation of ParFit class hierarchy.

ParFit.py are the three driver programs that use the ParFit library of classes as described in the next section.

In addition to *PFinp.py*, the *~/ParFit/Utility* subdirectory contains utility scripts, such as *clean.py* for the cleanup of temporary subdirectories and files and *QM_vs_MM_energies.py* for visualization of the fitting by the *matplotlib* Python library.

OBJECT-ORIENTED DESIGN AND CODE STRUCTURE

ParFit is written in an object-oriented fashion, and as a result, extensions can be quickly and easily made. For example, although ParFit currently works only with GAMESS[48,49] or NWChem[50] and the MM Engine,[51] for the ab initio and MM data respectively, the software can easily be extended to work with virtually any other MM and ab initio programs.

The backbone of ParFit consists of a relatively small hierarchy of classes: *Atom*, *Molecule*, *ScanElem*, *Scan*, *BondScan*, *AngleScan*, *DihAScan*, and *RunCmd*, as schematically depicted in Figure 2. (The important class attributes and methods are listed in Figure SI.1 from the Supporting Information section.) The *Atom* and *Molecule* classes are contained in the *GeomStr.py* file. *GeomStr.py* also contains four Python dictionaries: the default_charge, default_mm3_type, default_mmff94_type, and cov_radii, that store the charge, mm3 type, mmff94 type, and covalent radii of atoms, respectively. The *Atom* and *Molecule* classes use geometric functions defined in the helper file *_GeomCalc.py*. The *ScanElem*, *Scan*, *BondScan*, *AngleScan*, and *DihAScan* classes are contained in the *Scan.py* file. The *RunCmd* class is contained in the helper file *_Engine.py*. *_Engine.py* also contains several functions pertinent to running the MM Engine. Another helper file, *_IO.py*, contains helper functions that read and write the MM parameter file and read the ParFit input file. The *Ga.py* file contains a few GA related functions.

The class *Atom* encapsulates the atomic data that is relevant in the present context such as the atomic symbol, charge, Cartesian coordinate, and MM type for a particular atom. The MM type can currently be *mm3* or *mmff94* but the code can be extended to other MM types.

The class *Molecule* is constructed by a tuple of *Atoms* and has several member functions. Here tuple is used in the Python context and is essentially a sequence of immutable Python objects. Given a tuple of indices that correspond to atoms in a molecule, the member functions *calc_dist()*, *calc_angle()*, and *calc_dihedral()* compute a distance, angle, and dihedral angle, respectively. The member functions *bond_chng()*, *angl_chng()*, and *diha_chng()* change the bond length, bond angle, and dihedral angle, given a tuple of indices and a value of the respective change. The member function *set_conn()* determines interatomic distances and forms the molecular connectivity structure based only on atomic coordinates. A molecular bond is declared to exist between *atom1* and *atom2* if the interatomic distance between them is smaller than *cov_radii(atom1)+cov_radii(atom2)+wdv_dist_tol*, where *cov_radii(atomN)* is the covalent radius of an atom type corresponding to "*atomN*" and the constant *wdv_dist_tol* is chosen to be 0.45 Å. The member function *bond_ord()* automatically detects the bond order, i.e. whether a bond is single, double, or triple in character based on data in the Python dictionary *bond_ords* that contains the corresponding cutoff distances. The member function *benz_ring()* automatically detects benzene rings based on the established molecular connectivity.

The class *ScanElem* extends the class *Molecule* mainly by defining new member functions responsible for reading and writing GAMESS (or NWChem) and MM-engine input or output files.

The class *Scan* contains an extendable array that is filled by some of the member functions with instances of *ScanElem*. The read and write member functions of *Scan* delegate the read and write operations on individual *ScanElem*s to read and write member functions of the respective *ScanElem*s. For example, *read_gamess_outputs()* (or *read_nwchem_outputs()*) reads in data from a sequence of GAMESS (or NWChem) output files corresponding to a bond, angle, or dihedral angle scan. For each GAMESS (or NWChem) output file there is an associated instance of *ScanElem*. *read_gouts_data()* reads in data corresponding to a bond, angle, or dihedral angle scan from a single *comp*-type file. On the basis of the scan information stored in an array of *ScanElem*'s, the member function *write_engine_inputs()* creates a sequence of MM engine input files and the member function *run_scan()* initiates MM engine runs with these input files. *run_scan()* distributes the scan element runs across multiple cores on a given machine by the use of *Pool.map()* from the standard Python library *multiprocessing*. Each individual scan element run is initiated by *run_elem()*. *run_elem()* subsequently uses the *Run* member function of the *RunCmd* class from the helper file *_Engine.py*. *run_elem()* also uses the *read_add()* and *write_add()* functions from the helper file *_IO.py*. *read_add()* reads the current MM parameters that are being fitted, while *write_add()* writes these parameters back after an update by the fitting algorithm. The MM engine output files corresponding to one bond, angle, or dihedral angle scan are read back to *Scan* by the member function *read_engine_outputs()* and the root-mean-square error (RMSE) of the MM energies with respect to the relative ab initio energies is calculated by the member function *calc_rmse()*. The zero of the ab initio energy is shifted to the minimum or maximum of the MM energy or any other convenient reference energy.

The classes *BondScan*, *AngleScan*, and *DihAScan*, simple extensions of the base class *Scan*, set a member variable *self._styp* of the parent class *Scan* to *Bond*, *Angle*, or *DihA*, respectively. The member variable *self._styp* tunes the behavior of the *Scan* member functions to the respective scan type, i.e. depending on whether *self._styp* is *Bond*, *Angle*, or *DihA*. The base *Scan* member functions act as member functions of the *BondScan*, *AngleScan*, or *DihAScan* classes.

DOI: 10.1021/acs.jcim.6b00654
J. Chem. Inf. Model. 2017, 57, 391−396

FITTING ALGORITHMS

In the MM parameter fitting process, the RMSE is viewed as a function of the MM parameters; the fitting algorithm searches for a minimum in the MM parameter space. Three types of search algorithms are used in ParFit: the stochastic genetic algorithm (GA), the deterministic Powell or NM simplex algorithms (P/NM), and a hybrid algorithm that sequentially combines GA and P/NM algorithms (HYBR).

A limitation of the traditional deterministic optimization algorithms is that there is a high likelihood of the algorithms being trapped in a local minimum, because these algorithms utilize geometric information pertaining to the basin of the initial conditions only.

The stochastic optimization methods in general, and the GA algorithm in particular, are capable of avoiding the local-minima traps. An entire population of candidate solutions is considered at any given GA step and only one of these solutions happens to be the best. Nevertheless, the other members of the population might sample the configuration space outside of the basin, where the current best solution belongs, and thus a better global solution may be found throughout the next GA steps. To make the next GA step, some of the members randomly crossover and mutate. Ultimately, a selection process, in which the fittest members survive, forms the next population.

The rationale behind the hybridization is that while GA is better than P/NM in avoiding the traps of false local minima, GA might nevertheless face a very slow convergence rate once the search is relatively close to the global minimum. At this stage, P/NM is usually able to find the "global minimum" faster than GA. In practice, it is rarely known if the found minimum is the unique global minimum. The typical RMSE function has numerous shallow local minima and much deeper quasi-degenerate minima.

The Powell and NM simplex algorithms are implemented respectively by the *fmin* and *fmin_powell* functions from the *optimize* part of the *SciPy* library, while the GA algorithm is implemented based on the *Deap* library.[54−56] In the GA algorithm, the roulette wheel selection is used to form a new generation from an old one. The population size is chosen to be 50, the crossover probability is 35%, and the mutation probability is 5% per the recommendations in ref 45; however, the ParFit code can easily be modified for other GA algorithm variations. In the HYBR algorithm, the switchover between GA and P/NM occurs when the best fitness does not change for five generations.[45]

The existence of multiple quasi-degenerate minima is the main reason behind the difficulty of obtaining MM parameters that are suitable for simultaneously describing different molecules. If, on the other hand, the MM parameters are applicable to molecules beyond those used in the fitting process, the MM parameters are transferable. Because ParFit allows multiple molecules to be simultaneously used in the parameter fit, the transferability of the fitted MM parameters is enhanced as compared to manual fitting. In addition, the optimized MM parameters are constrained to lie within reasonable upper and lower limits to avoid unphysical MM parameter minima.

CONCLUSION

This article describes the object-oriented design, file, and code structures, algorithms, and functionality of a new object-oriented Python program called ParFit. The program automates the MM fitting process to recreate the ab initio energy profile. ParFit is aimed at quickly prototyping new parameter fitting algorithms. The fitting process is currently based on the minimization of RMSE, either by the Nelder−Mead simplex method, genetic algorithm, or by a hybrid of the two. ParFit can simultaneously handle MM parameters corresponding to several geometric variables in several molecules, a capability that enables a faster development of transferable MM parameters. ParFit is about an order of magnitude faster when run in parallel on modern multicore computer systems. In addition, ParFit can apply symmetric and antisymmetric constraints on the optimized MM parameters.

ASSOCIATED CONTENT

Supporting Information

The Supporting Information is available free of charge on the ACS Publications website at DOI: 10.1021/acs.jcim.6b00654.

Illustrative examples of MM3 parameter fitting with ParFit. Some of these examples demonstrate the enhanced transferability of the optimized MM3 parameters, when ParFit simultaneously optimizes several molecules. (PDF)

AUTHOR INFORMATION

Corresponding Author

*E-mail: fzahari@iastate.edu.

ORCID

Federico Zahariev: 0000-0002-3223-3576
Mark S. Gordon: 0000-0001-6893-553X
Theresa L. Windus: 0000-0001-6065-3167

Notes

The authors declare no competing financial interest.

ParFit is in an open source program available for free on GitHub (https://github.com/fzahari/ParFit).

ACKNOWLEDGMENTS

The authors thank Benjamin Hay for useful discussions. This work is supported by the Critical Materials Institute, an Energy Innovation Hub funded by the U.S. Department of Energy, Office of Energy Efficiency and Renewable Energy, Advanced Manufacturing Office.

REFERENCES

(1) Burkert, U.; Allinger, N. L. *Molecular Mechanics*; Monograph No. 177; American Chemical Society: Washington, DC, 1982.

(2) Kollman, P. A.; Merz, K. M. Computer Modeling of the Interaction of Complex Molecules. *Acc. Chem. Res.* **1990**, *23*, 246−252.

(3) McCammon, J. A.; Harvey, S. C. *Dynamics of Proteins and Nucleic Acids*; Cambridge University Press, New York, 1987.

(4) Kollman, P. A. Free Energy Calculations: Applications to Chemical and Biochemical Phenomena. *Chem. Rev.* **1993**, *93*, 2395−2417.

(5) Beveridge, D. I.; Dicapua, F. M. Free Energy via Molecular Simulations: Applications to Chemical and Biomolecular Systems. *Annu. Rev. Biophys. Biophys. Chem.* **1989**, *18*, 431−492.

(6) Van Gunsteren, W. F.; Berendsen, H. J. C. Computer Simulation of Molecular Dynamics. *Angew. Chem., Int. Ed. Engl.* **1990**, *29*, 992−1023.

(7) Jorgensen, W. L.; Chandrasekhar, J.; Madura, J. D.; Impey, R. W.; Klein, M. L. Comparison of Simple Potential Functions for Simulating Liquid Water. *J. Chem. Phys.* **1983**, *79*, 926−935.

(8) Miller, K. J.; Hinde, R. J.; Anderson, J. First and Second Derivative Matrix Elements for the Stretching, Bending, and Torsional Energy. *J. Comput. Chem.* **1989**, *10*, 63−76.
(9) Mohamadi, F.; Richards, N. G. J.; Guida, W. C.; Liskamp, R.; Lipton, M.; Caufield, C.; Chang, G.; Hendrickson, T.; Still, W. C. Macromodel—an Integrated Software System for Modeling Organic and Bioorganic Molecules Using Molecular Mechanics. *J. Comput. Chem.* **1990**, *11*, 440−467.
(10) Wang, W.; Wang, L.; Kollman, P. A. What Determines the Van Der Waals Coefficient β in the LIE (Linear Interaction Energy) Method to Estimate Binding Free Energies using Molecular Dynamics Simulations? *Proteins: Struct., Funct., Genet.* **1999**, *34*, 395−402.
(11) Srinivasan, J.; Miller, J.; Kollman, P. A.; Case, D. A. Continuum Solvent Studies of the Stability of RNA Hairpin Loops and Helices. *J. Biomol. Struct. Dyn.* **1998**, *16*, 671−681.
(12) Chong, L. T.; Duan, Y.; Wang, L.; Massova, I.; Kollman, P. A. Molecular Dynamics and Free-Energy Calculations Applied to Affinity Maturation in Antibody 48G7. *Proc. Natl. Acad. Sci. U. S. A.* **1999**, *96*, 14330−14335.
(13) Massova, I.; Kollman, P. A. Combined Molecular Mechanical and Continuum Solvent Approach (MM-PBSA/GBSA) to Predict Ligand Binding. *Perspect. Drug Discovery Des.* **2000**, *18*, 113−135.
(14) Massova, I.; Kollman, P. A. Computational Alanine Scanning to Probe Protein-Protein Interactions: a Novel Approach to Evaluate Binding Free Energies. *J. Am. Chem. Soc.* **1999**, *121*, 8133−8143.
(15) Lee, M. R.; Duan, Y.; Kollman, P. A. Use of MM-PB/SA in Estimating the Free Energies of Proteins: Application to Native, Intermediates, and Unfolded Villin Headpiece. *Proteins: Struct., Funct., Genet.* **2000**, *39*, 309−316.
(16) Reyes, C. M.; Kollman, P. A. Investigating the Binding Specificity of U1A-RNA by Computational Mutagenesis. *J. Mol. Biol.* **2000**, *295*, 1−6.
(17) Radmer, R. J.; Kollman, P. A. The Application of Three Approximate Free Energy Calculations Methods to Structure Based Ligand Design: Trypsin and its Complex with Inhibitors. *J. Comput.-Aided Mol. Des.* **1998**, *12*, 215−227.
(18) Darden, T.; York, D.; Pedersen, L. Particle Mesh Ewald: An N-log(N) Method for Ewald Sums in Large Systems. *J. Chem. Phys.* **1993**, *98*, 10089−10093.
(19) Essmann, U.; Perera, L.; Berkowitz, M. L.; Darden, T.; Lee, H.; Pedersen, L. A Smooth Particle Mesh Ewald Method. *J. Chem. Phys.* **1995**, *103*, 8577−8593.
(20) Moller, C.; Plesset, M. S. Note on an Approximation Treatment for Many-Electron Systems. *Phys. Rev.* **1934**, *46*, 618−622.
(21) Rappe, A. K.; Colwell, K. S.; Casewit, C. J. Application of a Universal Force Field to Metal Complexes. *Inorg. Chem.* **1993**, *32*, 3438−3450.
(22) Cornell, W. D.; Cieplak, P.; Bayly, C. I.; Gould, I. R.; Merz, K. M., Jr.; Ferguson, D. M.; Spellmeyer, D. C.; Fox, T.; Caldwell, J. W.; Kollman, P. A. A Second Generation Force Field for the Simulation of Proteins, Nucleic Acids, and Organic Molecules. *J. Am. Chem. Soc.* **1995**, *117*, 5179−5197.
(23) Wang, J.; Cieplak, P.; Kollman, P. A. How Well Does a Restrained Electrostatic Potential (RESP) Model Perform in Calculating Conformational Energies of Organic and Biological Molecules? *J. Comput. Chem.* **2000**, *21*, 1049−1074.
(24) Dixon, R. W.; Kollman, P. A. Advancing Beyond the Atom-Centered Model in Additive and Nonadditive Molecular Mechanics. *J. Comput. Chem.* **1997**, *18*, 1632−1646.
(25) Jorgensen, W. L.; Tirado-Rives, J. The OPLS [Optimized Potentials for Liquid Simulations] Potential Functions for Proteins, Energy Minimizations for Crystals of Cyclic Peptides and Crambin. *J. Am. Chem. Soc.* **1988**, *110*, 1657−1666.
(26) Allinger, N. L.; Yuh, Y. H.; Lii, J. H. Molecular Mechanics. The MM3 Force Field for Hydrocarbons. 1. *J. Am. Chem. Soc.* **1989**, *111*, 8551−8566.
(27) Lii, J. H.; Allinger, N. L. Molecular Mechanics. The MM3 Force Field for Hydrocarbons. 2. *J. Am. Chem. Soc.* **1989**, *111*, 8566−8575.
(28) Lii, J. H.; Allinger, N. L. Molecular Mechanics. The MM3 Force Field for Hydrocarbons. 3. *J. Am. Chem. Soc.* **1989**, *111*, 8576−8582.
(29) Halgren, T. A. Merck Molecular Force Field. I. Basis, Form, Scope, Parameterization, and Performance of MMFF94. *J. Comput. Chem.* **1996**, *17*, 490−519.
(30) Lemkul, J. A.; Huang, J.; Roux, B.; MacKerell, A. D., Jr. An Empirical Polarizable Force Field Based on the Classical Drude Oscillator Model: Development History and Recent Applications. *Chem. Rev.* **2016**, *116*, 4983−5013.
(31) Palmo, K.; Mannfors, B.; Mirkin, N. G.; Krimm, S. Potential Energy Functions: from Consistent Force Fields to Spectroscopically Determined Polarizable Force Fields. *Biopolymers* **2003**, *68*, 383−394.
(32) Izvekov, S.; Parrinello, M.; Burnham, C. J.; Voth, G. A. Effective Force Fields for Condensed Phase Systems from Ab Initio Molecular Dynamics Simulation: a New Method for Force-Matching. *J. Chem. Phys.* **2004**, *120*, 10896−10913.
(33) Izvekov, S.; Voth, G. A. Multiscale Coarse Graining of Liquid-State Systems. *J. Chem. Phys.* **2005**, *123*, 134105−1−1341015−13.
(34) Brommer, P.; Gahler, F. Potfit: Effective Potentials from Ab-Initio Data. *Modell. Simul. Mater. Sci. Eng.* **2007**, *15*, 295−304.
(35) Norrby, P. O.; Liljefors, T. Automated Molecular Mechanics Parameterization with Simultaneous Utilization of Experimental and Quantum Mechanical Data. *J. Comput. Chem.* **1998**, *19*, 1146−1166.
(36) Faller, R.; Schmitz, H.; Biermann, O.; Muller-Plathe, F. Automatic Parameterization of Force Fields for Liquids by Simplex Optimization. *J. Comput. Chem.* **1999**, *20*, 1009−1017.
(37) Guvench, O.; MacKerell, A. D., Jr. Automated Conformational Energy Fitting for Force-Field Development. *J. Mol. Model.* **2008**, *14*, 667−679.
(38) Wang, J. M.; Kollman, P. A. Automatic Parameterization of Force Field by Systematic Search and Genetic Algorithms. *J. Comput. Chem.* **2001**, *22*, 1219−1228.
(39) Pahari, P.; Chaturvedi, S. Determination of Best-Fit Potential Parameters for a Reactive Force Field using a Genetic Algorithm. *J. Mol. Model.* **2012**, *18*, 1049−1061.
(40) Okur, A.; Strockbine, B.; Hornak, V.; Simmerling, C. Using PC Clusters to Evaluate the Transferability of Molecular Mechanics Force Fields for Proteins. *J. Comput. Chem.* **2003**, *24*, 21−31.
(41) Strassner, T.; Busold, M.; Herrmann, W. A. MM3 Parametrization of Four- and Five-Coordinated Rhenium Complexes by a Genetic Algorithm—Which Factors Influence the Optimization Performance? *J. Comput. Chem.* **2002**, *23*, 282−290.
(42) Tafipolsky, M.; Schmid, R. Systematic First Principles Parameterization of Force Fields for Metal−Organic Frameworks using a Genetic Algorithm Approach, J. *J. Phys. Chem. B* **2009**, *113*, 1341−1352.
(43) Wang, L.-P.; Chen, L.; Van Voorhis, T. Systematic Parametrization of Polarizable Force Fields from Quantum Chemistry Data. *J. Chem. Theory Comput.* **2013**, *9*, 452−460.
(44) Wang, L.-P.; Martinez, T. J.; Pande, V. S. Building Force Fields: An Automatic, Systematic, and Reproducible Approach. *J. Phys. Chem. Lett.* **2014**, *5*, 1885−1891.
(45) Betz, R. M.; Walker, R. C. Paramfit: Automated Optimization of Force Field Parameters for Molecular Dynamics Simulations. *J. Comput. Chem.* **2015**, *36*, 79−87.
(46) Norrby, P. O.; Brandt, P. Deriving Force Field Parameters for Coordination Complexes. *Coord. Chem. Rev.* **2001**, *212*, 79−109.
(47) http://github.com/fzahari/ParFit (accessed Jan 1, 2017).
(48) Schmidt, M. W.; Baldridge, K. K.; Boatz, J. A.; Elbert, S. T.; Gordon, M. S.; Jensen, J. H.; Koseki, S.; Matsunaga, N.; Nguyen, K. A.; Su, S.; Windus, T. L.; Dupuis, M.; Montgomery, J. A. General Atomic and Molecular Electronic Structure System. *J. Comput. Chem.* **1993**, *14*, 1347−1363.
(49) Gordon, M. S.; Schmidt, M. W. Advances in Electronic Structure Theory: GAMESS a Decade Later. In *Theory and Applications of Computational Chemistry: the First Forty Years*. Dykstra, C. E., Frenking, G., Kim, K. S., Scuseria, G. E., Eds.; Elsevier: Amsterdam, 2005; pp 1167−1189.

DOI: 10.1021/acs.jcim.6b00654
J. Chem. Inf. Model. 2017, 57, 391−396

(50) Valiev, M.; Bylaska, E. J.; Govind, N.; Kowalski, K.; Straatsma, T. P.; van Dam, H. J. J.; Wang, D.; Nieplocha, J.; Apra, E.; Windus, T. L.; de Jong, W. A. NWChem: A Comprehensive and Scalable Open-Source Solution for Large Scale Molecular Simulations. *Comput. Phys. Commun.* **2010**, *181*, 1477−1489.
(51) Gilbert, K. MM Engine of PCModel, version 9.2; Serena Software, Bloomington, Indiana. http://www.serenasoft.com/ (accessed Dec 20, 2016).
(52) Gundertofte, K.; Liljefors, T.; Norrby, P. O.; Pettersson, I. A Comparison of Conformational Energies Calculated by Several Molecular Mechanics Methods. *J. Comput. Chem.* **1996**, *17*, 429−449.
(53) Bode, B. M.; Gordon, M. S. MacMolPlt: A Graphical User Interface for GAMESS. *J. Mol. Graphics Modell.* **1998**, *16*, 133.
(54) Distributed Evolutionary Algorithms in Python. http://deap.gel.ulaval.ca (accessed Dec 20, 2016).
(55) Fortin, F.-A.; De Rainville, F.-M.; Gardner, M.-A.; Parizeau, M.; Gagne, C. Deap: Evolutionary Algorithms Made Easy. *J. Mach. Learn. Res.* **2012**, *13*, 2171−2175.
(56) Rainville, F.-M. D.; Fortin, F.-A.; Gardner, M.-A.; Parizeau, M.; Gagne, C. DEAP: a Python Framework for Evolutionary Algorithms. In *Companion Proceedings of the Genetic and Evolutionary Computational Conference*; ACM: New York, 2012; pp 85−92.

DOI: 10.1021/acs.jcim.6b00654
J. Chem. Inf. Model. 2017, 57, 391−396