MDSimAid: Automatic Parameter Optimization in Fast Electrostatic Algorithms

Michael S. Crocker Scott S. Hampton

Jesús A. Izaguirre*

Department of Computer Science and Engineering

University of Notre Dame

Notre Dame, IN 46556-0309, USA

September 25, 2004

Abstract

MDSimAid is a recommender system that generates nearly optimal parameters for Particle Mesh Ewald (PME) and multigrid (MG) summation fast electrostatic solvers. MDSimAid optimizes the running time of these methods within a given error tolerance. MDSimAid performs a run-time constrained search on the parameter space of each method starting from semi-empirical performance models. Recommended parameters are presented to the user. MDSimAid's optimization of MG leads to configurations that are up to 14 times faster or 17 times more accurate than published recommendations. Optimization of PME leads to configurations up to 67 times more accurate. MDSimAid and its Python source code are accessible through a web portal located at http://mdsimaid.cse.nd.edu.

*Corresponding author: izaguirr@cse.nd.edu

1 Introduction

At each timestep of a molecular dynamics (MD) simulation under periodic boundary conditions (PBC), it is often necessary to solve the following summation to compute the electrostatic potential energy:

$$U^{\text{electrostatic}} = \sum_{i=1}^{N} \sum_{j \notin \chi(i)} \sum_{\mathbf{m} \in \mathbb{Z}^3}{}' \frac{q_i q_j}{4\pi\varepsilon_0 \left|\mathbf{r}_j - \mathbf{r}_i + \mathbf{m}L\right|}, \tag{1}$$

and to compute the electrostatic force,

$$F^{\text{electrostatic}} = -\nabla U^{\text{electrostatic}}, \tag{2}$$

where ε_0 is the dielectric coefficient; $\mathbf{r}_i$ and q_i are the position and charge of particle i; the set $\chi(i)$ contains atoms to be excluded from calculation with atom i, including those that are covalently bonded to i; the integral triplet $\mathbf{m}$ is a particular periodic image; and L is the length of a periodic cell of volume L^3. Eq. (1) gives a conditionally convergent, infinite summation over all periodically replicated cells. Conditional convergence means that the sum may or may not converge depending on the order in which the summation is performed.

Direct computation of all interacting pairs of N particles leads to an $O(N^2)$ algorithm. Significant effort has been put into reducing the computational cost of calculating these interactions. The Ewald [1] and particle mesh Ewald (PME) [2,3] methods have been optimized to run in $O(N^{3/2})$ and $O(N \log N)$ time, respectively. The fast multipole (FMM) [4] and the multigrid summation (MG) [5–7] methods have improved the asymptotic running time to $O(N)$.

While these fast solvers are asymptotically quicker, in practice there are numerous parameters to configure for optimal performance depending on the system size and desired accuracy. An optimal choice of parameters cannot be fully accomplished analytically due to the complexity of the methods. However, there are many studies that propose schema for determining parameter values for various fast force evaluators [8–16].

MDSimAid automates the optimization of parameters for PME and MG, freeing users from the time consuming process of a manual search. Starting with published recommendations, we performed extensive empirical testing to determine which factors have the greatest impact on accuracy and runtime performance. Using these

results, we have implemented heuristics that efficiently search the parameter space of PME and MG and return near optimal configurations.

We observe large improvements over published recommendations. For a force error tolerance of 1%, MDSimAid's recommendation for MG is up to 14 times faster or 4 times more accurate than the initial prediction. For a force error tolerance of 0.1%, MDSimAid's recommendation is up to 10 times faster or 17 times more accurate. For PME and a force error of 0.0001%, MDSimAid's recommendation is up to 67 time more accurate with only a tenfold increase in time.

MDSimAid operates as a web portal [17]. Source and examples can be obtained from the web page. MDSimAid is written in Python and uses ProtoMol [18] to test PME and MG. ProtoMol is a high-performance MD object-oriented software framework.

1.1 Recommender systems

The purpose of a recommender system is to rank a set of algorithms on particular problems according to some performance criteria. In scientific fields, recommender systems are often used to suggest optimal choices of software (PYTHIA-II [19], MyPYTHIA [20], SANS [21]), algorithms (SPIRAL [22]), and input configurations (PBCAID [23]). PBCAID, for example, optimizes the rotational position of a molecular system to reduce the size of the bounding box used in periodic MD simulations. Smaller bounding boxes provide substantial savings, but are time consuming to calculate by hand.

There are many approaches that recommender systems use for the optimization process: random searches, linear and nonlinear optimization, dynamic programming, tree searches, and machine learning techniques. The methodology typically includes offline (*a priori*) and online (at run-time) profiling of performance data and construction of performance models. We use offline profiling of performance data to produce heuristics that guide the optimization at run-time. Whereas, the initial guess comes from analytical performance models.

2 Characteristics of fast electrostatics solvers

In this section, we briefly describe the fast electrostatics solvers Ewald, PME, and MG.

2.1 Ewald summation

A derivation and analysis of the Ewald summation method can be found in references [1, 24]. Ewald splits Eq. (1) into the sum of two rapidly converging sums:

$$\frac{1}{r} = \underbrace{\left(\frac{1}{r} - f_{\text{smooth}}(r)\right)}_{\text{Real-space}} + \underbrace{f_{\text{smooth}}(r)}_{\text{Reciprocal-space}}, \tag{3}$$

where $f_{\text{smooth}}(r)$ is a softening function. The real-space sum is of the form:

$$\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} q_i q_j \sum_{\mathbf{m}}{}' \left(\frac{1}{r_{ij,\mathbf{m}}} - \frac{\text{erf}(\alpha r_{ij,\mathbf{m}})}{r_{ij,\mathbf{m}}}\right), \tag{4}$$

and the reciprocal-space sum:

$$\frac{1}{2\pi\varepsilon_0 L^3}\sum_{\mathbf{m}\neq 0}\frac{1}{\mathbf{m}^2}\exp\left(\frac{-\pi^2\mathbf{m}^2}{\alpha^2}\right)\hat{\rho}(\mathbf{m})\hat{\rho}(-\mathbf{m}), \tag{5}$$

where

$$\hat{\rho}(\mathbf{m}) = \sum_j q_j \exp(2\pi i \mathbf{m}\cdot\mathbf{r}_j). \tag{6}$$

The parameter α determines the convergence rates of the real- and reciprocal- space sums. Optimal choice of α leads to an $O(N^{3/2})$ algorithm [10, 14].

2.2 Particle Mesh Ewald

PME is similar to Ewald, but PME chooses the splitting parameter α such that the real-space part is $O(N)$, and interpolates the Fourier series, Eq. (5), onto a mesh, thus allowing the use of $O(N\log N)$ FFT for the reciprocal-space part:

$$\hat{\rho}(\mathbf{m}) \approx \sum_n \exp(2\pi i \mathbf{m}\cdot\mathbf{r}_n^h)\sum_j \phi_n(\mathbf{r}_j)q_j, \tag{7}$$

where $\phi_n(F_j)$ does an adjoint interpolation (also called anterpolation) of the charges to the grid, $\mathbf{r}_n^h$ are grid positions, and $\hat{\rho}$ is defined by Eq. (6).

2.3 Multigrid summation

The MG summation splits the contributions to the electrostatic energy of Eq. (1) into a short-range part that is evaluated using a cutoff, and a smooth part. Switching functions are used to bring the energy function smoothly to zero at a cutoff point. The smooth part is approximated on a grid, by first anterpolating the particle charges. This process is repeated recursively, creating a hierarchy of grids, hence the name multigrid summation. In general, 2 to 4 grids are common. Once the grids are formed, the values of the forces can be calculated. The steps of the algorithm are set within Figure 1, starting from the lower left. Algorithm 1 gives an overview of the MG method, while Figure 1 shows schematically what is happening.

Algorithm 1 Pseudo-code of a recursive multi-grid scheme, cf. [25].

main:

1. anterpolate particle charges to the finest charge grid(1) – step (1);
2. call **multiscale**(maxLevel, level := 1);
3. interpolate finest energy grid(1) to particles – step (3);
4. correct particle energies – step (4);
5. compute forces and total energy

multiscale(maxLevel, level := k):

1. if maxLevel = k then
   (a) compute energy values on coarsest grid(maxLevel) – step (2);
2. otherwise
   (a) anterpolate charge grid(k) to coarser charge grid($k+1$) – step (1);
   (b) call **multiscale**(maxLevel, $k+1$);
   (c) interpolate coarser energy grid($k+1$) to finer energy grid(k) – step (3);
   (d) correct energy grid(k) – step (4)

3 Optimization methodology

The focus of MDSimAid is to optimize PME and MG using Ewald as a benchmark. Starting parameter values for each method are generated using the desired accuracy and system size as input. Default values come from equations in the literature. The optimization works in two stages by first obtaining the greatest accuracy and then relaxing the parameters to decrease the runtime while maintaining the desired accuracy. In the case where MDSimAid is unable to reach the desired accuracy, the focus becomes minimizing the cost metric, cf. Eq. (13). An overview of MDSimAid is given in Algorithm 2, with details following in Sections 3.1 – 3.3.

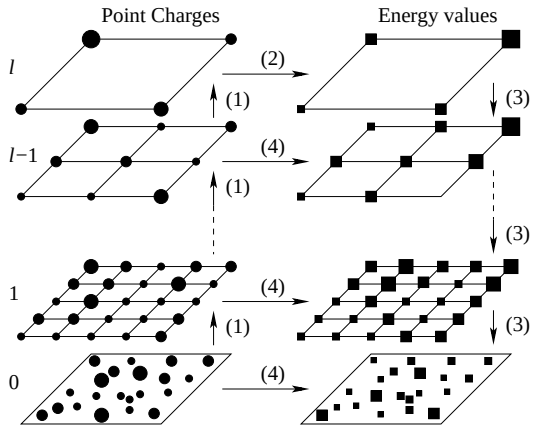


Figure 1: A pictorial view of the MG algorithm, cf. [25].

Algorithm 2 MDSimAid

1. After generating the initial configuration, perform optimizations on each parameter in the designated order to decrease the error in the simulation. At each step, store results along with an overall performance metric.

   (a) Perform a local search by varying r_c and h using small increments. (MG)

   (b) h: Using the most accurate test configuration from Step (1a) decrease the grid-size, h, until the accuracy is approximately constant. (MG)

   (c) r_c: Using the most accurate test configuration from Step (1b) for MG and from Step (1a) for PME, increase the softening distance, r_c, until the error is minimized. (MG and PME)

   (d) a: If the most accurate test configuration from Step (1c) is not within the error tolerance, increase the PME accuracy parameter, a, by a factor of 10 to decrease the error. (PME)

   (e) l: Using the five smallest values of h found in step (1c), increase the number of levels, l, until there are an insufficient number of grid points for an additional level of interpolation. (MG)

2. Having determined the best accuracy possible to this point, perform optimizations on each parameter in the designated order to decrease the runtime of the simulation.

   (a) l, h: For the set of configurations that satisfy the error tolerance, try to increase l by 1 and double h. By doubling h, we reduce the runtime and the accuracy; we compensate the accuracy loss by increasing l. (MG)

   (b) r_c: Decrease r_c until the error exceeds the desired error tolerance in order to decrease the runtime. Do this for up to five configurations within the error tolerance. (MG and PME)

3. Choose parameters that best meet user constraints while minimizing the cost metric and create necessary output files.

3.1 Optimization of MG

MG has five parameters: (*i*) the cutoff, r_c, used at the particle level; (*ii*) the grid size, h, at the finest grid; (*iii*) the number of grid levels, l; (*iv*) the smoothness of the switching function, k, (C^1, C^2, etc.); and (*v*) the interpolation order, p, from one level to the next.

MDSimAid starts MG runtime tests by evaluating an analytical model to determine a starting point. The authors of [7] suggest balancing the work between the short-range and the smooth parts. For an approximation of order p to the smooth part of the force, the suggested values are given by:

$$h = \left(1 + \frac{4}{p}\right)^{1/6} \left(\frac{64}{7}\theta\right)^{1/6} h_*, \tag{8}$$

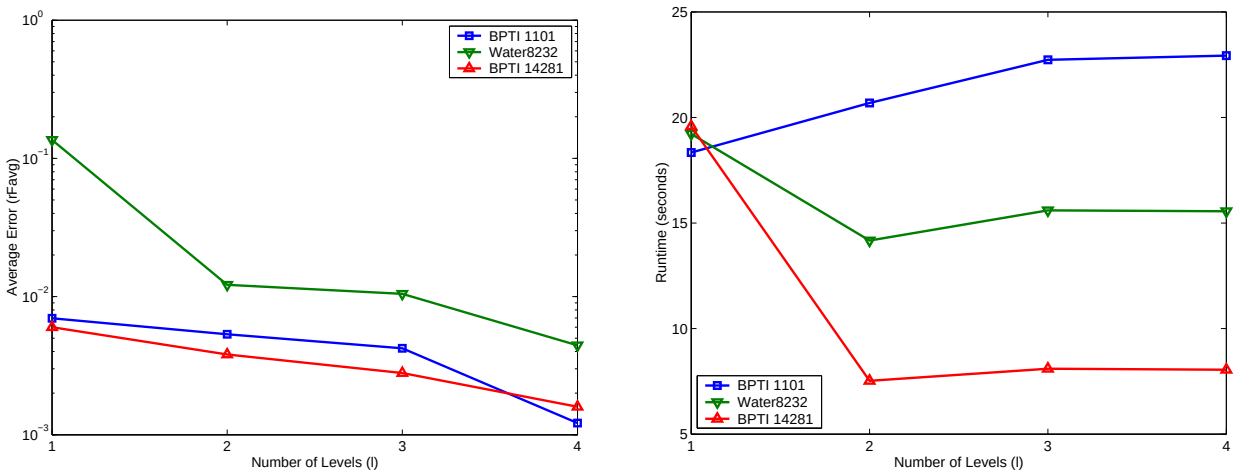
$$r_c = \left(h^p h_*^2 \frac{C_p}{\varepsilon}\right)^{1/(p+2)}, \tag{9}$$

where h_* is the distance between two nearest neighbors; θ is the ratio of the cost of calculating a pairwise interaction between grid points to that of calculating a pairwise interaction between particles; ε is the desired accuracy; and C_p is a constant determined at runtime.

Our simulations have shown that h is generally between 1 and 5 Å. Such spacing is suggested by Eqs. (8) and (9). Improvements in accuracy are obtained by decreasing h, increasing r_c (up to L), and through an increase in the number of levels, l. MG's advantage is the ability to interpolate to and from coarse and fine grids to get similar accuracy for less cost. Using additional levels is a major focus of MDSimAid's recommendations for MG.

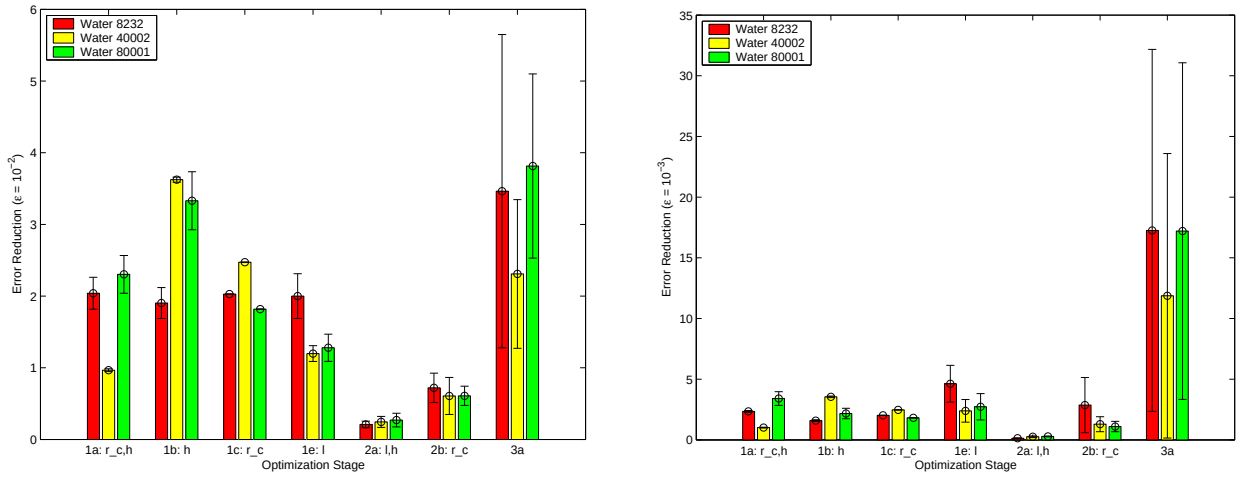
Figure 2(a) illustrates the effect of increasing l on accuracy. In a small system, a solvated BPTI 1101 atoms for example, the overhead of the interpolation outweighs the computational savings in computing the summation. For larger systems, the gain is evident.

We generate the default MG parameters and in Step (1a) perform a local search around r_c and h using small increments to look for quick improvements. The configuration from the local search that has the smallest error is used to continue the accuracy optimizations. In Step (1b), h is modified with larger increments until further changes in the error are negligible. For MG, h plays a significant role affecting both error and runtime. Step (1c) takes the configuration from the previous step and increases r_c until the error is minimized. In Step (1e), l is increased while keeping the finest grid size the same. Generally, an increase in l causes the greatest reduction



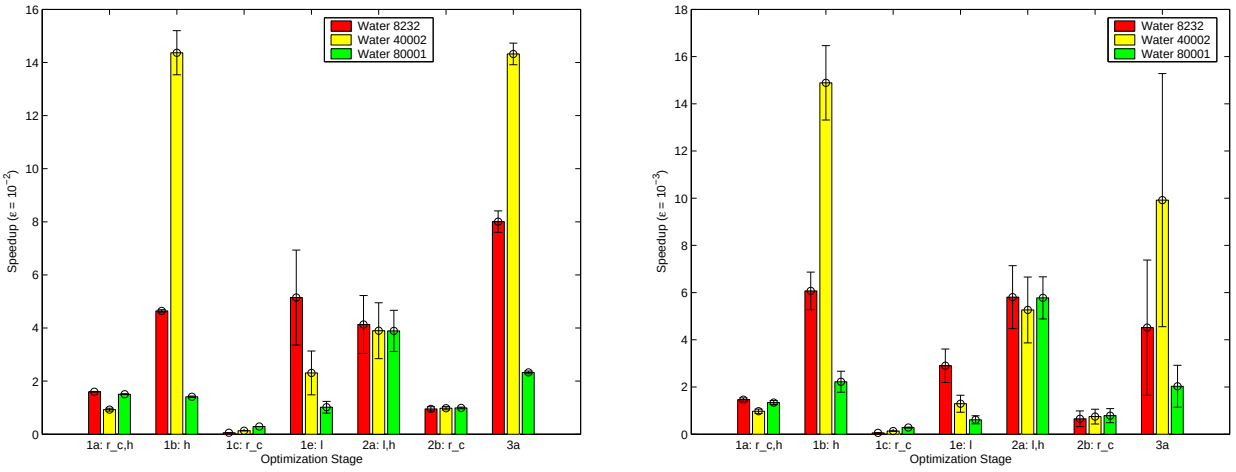
(a) (b)

Figure 2: The effect of number of levels, l, on (a) the average error, and on (b) the runtime. There are 32 grid points per dimension in the finest level grid.



(a) (b)

Figure 3: Error reduction relative to the previous step for MG. The last column is overall reduction from the initial equations. (a) shows values for $\varepsilon \geq 10^{-2}$, and (b) for $\varepsilon \geq 10^{-3}$.



(a) (b)

Figure 4: Runtime reduction relative to the previous step for MG. The last column is overall reduction from the initial equations. (a) shows values for $\varepsilon \geq 10^{-2}$, and (b) for $\varepsilon \geq 10^{-3}$.

of error when compared to changes in h or r_c, as is shown in section 4.3. However, we first increase h and r_c to allow the increase of l. If either h or r_c is too small, there will not be enough support for the interpolation to coarser grids. The order and kernel parameters for MG are not changed by MDSimAid. For these parameters, we use the default values of $p = 4$ and $k = C^2$.

For the second stage, Step (2a) increases l and h simultaneously. Making the grid coarser would normally increase the error, but increasing the levels makes up for the accuracy while also improving time significantly. Step (2b) reduces r_c for as long as the error is within the tolerance.

3.2 Optimization of Ewald

Fincham [16] and other researchers [13, 26] recommend that α vary with N such that

$$\alpha = c\sqrt{\pi}\left(\frac{N}{V^2}\right)^{1/6}, \tag{10}$$

where V is the volume of the system and c is the ratio of execution time of the real part to that of the reciprocal. Since c may vary from one platform to another, this ratio is computed by ProtoMol at run time.

In addition to α, the direct space cutoff distance r_c, controls accuracy and CPU time in Ewald. The relationship between α and r_c is given by

$$r_c = \frac{\sqrt{-\ln \varepsilon}}{\alpha}, \tag{11}$$

where ε is the desired accuracy. A typical value for ε is 10^{-5}. Large α yields a small cutoff distance, faster execution time, and lower accuracy. Since Ewald is easily optimized and highly accurate, MDSimAid uses Ewald to determine the error in PME and MG.

3.3 Optimization of PME

PME has three method parameters: the first, α, determines the amount of work that is done in real and reciprocal space; the second is the cutoff r_c used to evaluate Eq. (4); the third is h, the grid point separation used by the FFT. The accuracy and the cost are proportional to r_c and inversely proportional to h.

The α and r_c parameters have the same qualitative effects in PME as in Ewald. Petersen [13] showed that the optimal α is approximately $N^{1/3}$, whereas the optimal r_c is proportional to $\log_2^{1/2} N$. In particular, once α is determined (based on N), r_c can be estimated as follows:

$$\frac{\operatorname{erfc}(\alpha r_c)}{r_c} = \varepsilon, \tag{12}$$

where ε is the desired accuracy.

In PME, the value of h has a less significant impact on the accuracy and runtime than in MG. Default values of h are generated and are not changed in the optimization. We use cubic B-spline interpolation for PME.

The initial parameters are calculated from the analytical models, while, at the same time, an appropriate h based on the system size is determined. In Step (1c), r_c is reduced. MDSimAid then checks to see if the desired accuracy has been reached. If not, ε is reduced by a factor of 10 in Step (1d). This step has the greatest impact on the accuracy. The fastest configuration found up to that point that is also within the desired accuracy is chosen for Step (2b). Here, we reduce r_c only.

4 Results

MDSimAid significantly improves runtime and accuracy of MG with $\varepsilon = 10^{-3}$ or $\varepsilon = 10^{-2}$. It improves the accuracy of PME for $\varepsilon = 10^{-6}$ at the expense of runtime when compared to Eq. (9). It does not have a significant effect for lower accuracies using PME.

4.1 Test systems

MDSimAid's rules were obtained by testing flexible TIP3P [27] water boxes and solvated protein systems from 1,000 to 80,000 atoms. ProtoMol simulations were run under PBC using the CHARMM 28a2 [28] force field.

4.2 Evaluating accuracy

MDSimAid uses three different values for analyzing the effectiveness of each configuration, or set of parameters. The first is runtime t, which is simply the amount of time used by ProtoMol to complete one simulation step. The second, average error $rFavg$, comes from ProtoMol's built-in force comparator. Finally, the cost metric, M_e, is defined as

$$M_e = \frac{t}{|\log_{10}(rFavg)|}, \tag{13}$$

and is a simple way to combine runtime and error in a single value. M_e is the time it takes to achieve a tenfold improvement in the error. A lower M_e means either that the runtime cost to attain the desired accuracy was lower, or that a smaller error was attained with virtually the same runtime cost.

ProtoMol computes the relative error in force and potential energy by using a specified method as a reference. In the case of MDSimAid, the reference method is Ewald. Errors are computed by ProtoMol comparing Ewald to MG or Ewald to PME. The metric for relative force error is defined as follows:

$$rFavg = \frac{\sum_i \sqrt{\frac{||F_i - \tilde{F}_i||^2}{m_i}}}{\sum_i \sqrt{\frac{||F_i||^2}{m_i}}}, \tag{14}$$

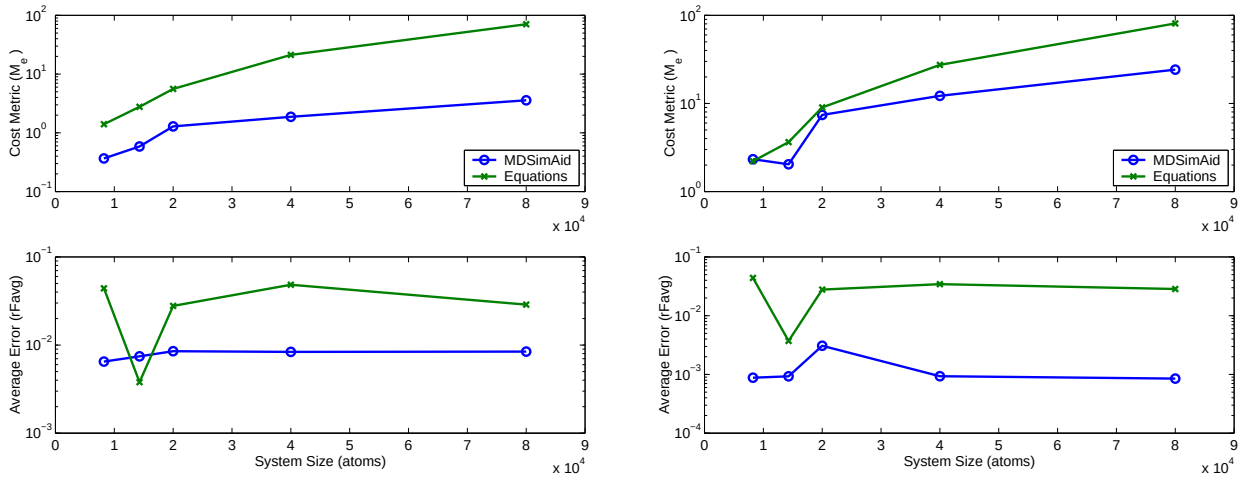
$$\tag{15}$$

where m_i is the mass of atom i; F is the force of the reference algorithm; and $\tilde{F}$ is the force of the algorithm

being tested.

4.3 Evaluation of MDSimAid's effectiveness

Figures 5(a) and 5(b) compare MDSimAid's recommendations for MG against those suggested by Eqs. (8)–(9). The tests were run with ProtoMol using MG on molecular systems of sizes 8,000 to 80,000 atoms. Results are presented showing both $rFavg$ and the cost metric M_e. MG does not show substantial advantages over PME for less than 8,000 atoms in our implementation. As the system size increases, the recommender is more efficient at optimizing M_e.



(a) (b)

Figure 5: Recommendation of analytic methods vs. MDSimAid for $\varepsilon = 10^{-2}$, (a), and $\varepsilon = 10^{-3}$, (b). The top graph displays the cost metric, M_e, while the bottom graph shows achieved accuracy.

Figures 3(a) and 3(b) show the error reduction of each optimization step for MG with respect to the previous one, as well as the overall error reduction for $\varepsilon = 10^{-2}$, and $\varepsilon = 10^{-3}$. The error bar is computed over 9 runs with different target accuracies. Similarly, Figures 4(a) and 4(b) show the speedup of each optimization step for MG with respect to the previous one, as well as the overall speedup and error bars for the corresponding accuracies in Figures 3(a) and 3(b).

These tests involve 8 water systems containing 1,029 to 80,001 atoms. Each system was tested with 18 error tolerances. For MG, we used values between 2.5×10^{-4} and 5×10^{-2}. Since PME can attain higher accuracy, we

used values between 7.5×10^{-7} and 5×10^{-4}. These values were chosen to go beyond the accuracy capabilities of each method in order to fully test MDSimAid. With a sufficient number of atoms, MG and PME can attain accuracies of 10^{-3} and 10^{-6} respectively.

For MG, optimizing h and l have the highest impact on both accuracy and runtime (Steps (1b), (1e), (2a)). Whereas, for PME, optimizing r_c has the highest impact on accuracy. We were unable to obtain significant runtime speedups for PME.

4.4 Comparison of PME and MG

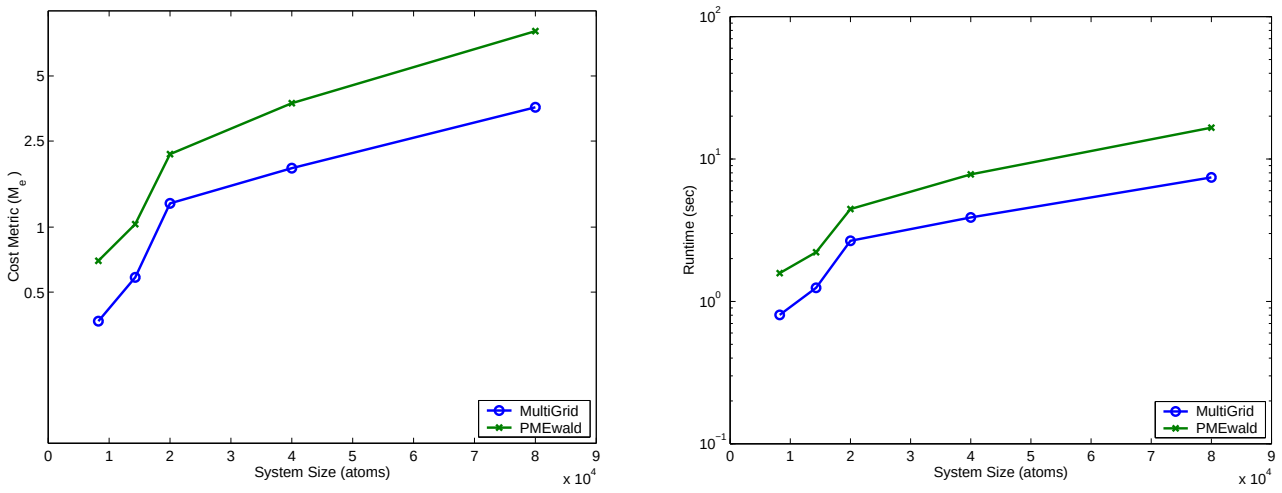
Figures 6(a) and 6(b) compare runtime and cost metric of optimized runs of PME and MG for different system sizes. For error tolerances of 10^{-2} and 10^{-3}, MG performs better than PME in all cases. For error tolerances smaller than 10^{-3}, MG is unable to match PME. Our implementation of MG in PBC is limited to accuracies of 10^{-3} at a reasonable cost. These errors in MD simulations do not prevent the ability to reproduce dynamical and structural features. The smoothness of the forces computed with MG makes up for the moderate accuracy. For instance, [29, p. 494] reports a successful 500 ps simulation of solvated melittin [30] (11,845 atoms) using MG and PBC. The radial distribution function and self-diffusion coefficient are identical to those obtained using PME or Ewald.

5 Discussion

MDSimAid has proven to be a practical tool for generating efficient parameters for PME and MG, thereby making them more accessible. This is especially true for MG. Even so, there are many features of MDSimAid that could be improved.

The methodology of MDSimAid could be extended by making it into an adaptive application that improves future recommendations. The runtime optimizations could be performed in a cluster or computational grid, and the search algorithms could be enhanced.

MDSimAid optimizes sequential versions of PME and MG, although parallel versions are available. Extending MDSimAid to optimize these parallel implementations would particularly benefit MG simulations. For instance, the parallel implementation of MG was shown to have an efficiency of 60% on a 48 processors Linux cluster with



(a) (b)

Figure 6: Comparison of optimized PME vs. MG with $\varepsilon = 10^{-2}$: (a) the cost metric, M_e, and (b) runtime, t.

a 92,000 atom system, whereas PME only achieved comparable efficiency on 10 processors [5].

MDSimAid could be extended to optimize other fast electrostatic solvers, such as FMM and others, cf. [4, 31–35]. Optimization of fast electrostatics in combination with multiple time stepping integrators is another area for further development, cf. [36]. Finally, MDSimAid should have the ability to interface with well known MD simulators, such as NAMD [37], CHARMM [28], and AMBER [38].

5.1 Acknowledgments

This work was partially supported by an NSF Career Award ACI-0135195. Computations were performed in part through a Beowulf cluster supported by NSF grant DMR-0079647 and also by an Equipment Renovation grant through the University of Notre Dame. Michael Crocker was supported by an ICSB Undergraduate Research scholarship and NSF REU ACI-0135195. Scott Hampton was supported through an Arthur J. Schmitt fellowship.

References

[1] P. Ewald, *Ann. Phys.*, **64**, 253–287 (1921).

[2] T. Darden, D. York, and L. Pedersen, *J. Chem. Phys.*, **98**(12), 10089–10092 (1993).

[3] U. Essmann, L. Perera, and M. L. Berkowitz, *J. Chem. Phys.*, **103**(19), 8577–8593 (1995).

[4] L. Greengard and V. Rokhlin, *J. Comput. Phys.*, **73**, 325–348 (1987).

[5] J. A. Izaguirre and T. Matthey, *J. Paral. Distrib. Comp.* (2004).

[6] A. Brandt, J. Bernholc, and K. Binder, Eds., *Multiscale Computational Methods in Chemistry and Physics*, Vol. 177 of *NATO Science Series: Series III Computer and Systems Sciences*, IOS Press, Amsterdam, Netherlands, 2001.

[7] R. D. Skeel, I. Tezcan, and D. J. Hardy, *J. Comp. Chem.*, **23**(6), 673–684 (2002).

[8] E. L. Pollock and J. Glosli, *Comput. Phys. Commun.*, **95**, 93–110 (1996).

[9] M. Kawata and M. Mikami, *J. Comp. Chem.*, **21**(3), 201–217 (2000).

[10] A. Y. Toukmaji and J. A. Board, *Comput. Phys. Commun.*, **95**, 73–92 (1996).

[11] P. F. Batcho, D. A. Case, and T. Schlick, *J. Chem. Phys.*, **115**(9), 4003–4018 (2001).

[12] R. Zhou, E. Harder, H. Xu, and B. J. Berne, *J. Chem. Phys.*, **115**(5), 2348–2358 (2001).

[13] H. G. Petersen, *J. Chem. Phys.*, **103**(3668–3679) (1995).

[14] M. Deserno and C. Holm, *J. Chem. Phys.*, **109**(18), 7678–7693 (1998).

[15] C. Sagui and T. A. Darden, *Ann. Rev. Biophys. Biomol. Struct.*, **28**, 155–179 (1999).

[16] D. Fincham, *Mol. Sim.*, **13**, 1–9 (1994).

[17] MDSimAid, Molecular Dynamics Simulation Aid `http://mdsimaid.cse.nd.edu/`, 2004.

[18] T. Matthey, T. Cickovski, S. S. Hampton, A. Ko, Q. Ma, M. Nyerges, T. Raeder, T. Slabach, and J. A. Izaguirre, *ACM Transactions on Mathematical Software*, **30**(3) (2004).

[19] E. N. Houstis, A. C. Catlin, J. R. Rice, V. S. Verykios, N. Ramakrishnan, and C. E. Houstis, *ACM Transactions on Mathematical Software*, (2), 227–253 (2000).

[20] E. N. Houstis, A. C. Catlin, J. R. Rice, N. Ramakrishnan, and V. S. Verykios, *Concurrency and Computation: Practice and Experience*, **14**(13–14), 1481–1505 (2002).

[21] J. Dongarra and V. Eijkhout, *Int. J. of High Perf. Comp. App.*, **17**(2), 125–131 (2003).

[22] M. Püschel, B. Singer, J. Xiong, J. Moura, J. Johnson, D. Padua, M. Veloso, and R. W. Johnson, *Int. J. of High Perf. Comp. App.*, **18**(1), 21–45 (2004).

[23] PBCAID, Pbcaid `http://monod.biomath.nyu.edu/index/software.html`, 2004.

[24] S. W. de Leeuw, J. W. Perram, and E. R. Smith, *Proc. R. Soc. Lond. A*, **373**, 27–56 (1980).

[25] T. Matthey *Framework Design, Parallelization and Force Computation in Molecular Dynamics* PhD thesis, University of Bergen, Bergen, Norway, (2002).

[26] Z. Wang and C. Holm, *J. Chem. Phys.*, **115**(14), 6351–6359 (2001).

[27] W. L. Jorgensen, J. Chandrasekhar, J. D. Madura, R. W. Impey, and M. L. Klein, *J. Chem. Phys.*, **79**, 926–935 (1983).

[28] B. R. Brooks, R. E. Bruccoleri, B. D. Olafson, D. J. States, S. Swaminathan, and M. Karplus, *J. Comp. Chem.*, **4**(2), 187–217 (1983).

[29] A. Ko MDSimAid: An automatic recommender for optimization of fast electrostatic algorithms for molecular simulations, Master's thesis, University of Notre Dame, Dec. (2002).

[30] M. Gribskov, L. Wesson, and D. Eisenberg Structure available at http://http://www.rcsb.org/pdb/, (1990).

[31] L. Y. Zaslavsky and T. Schlick, *Appl. Math. Comput.*, **97**, 237–250 (1998).

[32] C. Sagui and T. Darden, *J. Chem. Phys.*, **114**(15), 6578–6591 (2001).

[33] D. York and W. T. Yang, *J. Chem. Phys.*, **101**, 3298–3300 (1994).

[34] P. Bode and J. P. Ostriker, *Astrophysical J. Supplement Series*, **145**(1), 1–13 (2003).

[35] Z. H. Duan and R. Krasny, *J. Comp. Chem.*, **22**(2), 184–195 (2001).

[36] M. Tuckerman and B. Berne, *J. Chem. Phys.*, **94**(10), 6811–6815 (1991).

[37] L. Kalé, R. Skeel, M. Bhandarkar, R. Brunner, A. Gursoy, N. Krawetz, J. Phillips, A. Shinozaki, K. Varadarajan, and K. Schulten, *J. Comput. Phys.*, **151**, 283–312 (1999).

[38] P. K. Weiner and P. A. Kollman, *J. Comp. Chem.*, **2**, 287–303 (1981).