JCTC Journal of Chemical Theory and Computation

Multiscale Optimization of a Truncated Newton Minimization Algorithm and Application to Proteins and Protein−Ligand Complexes

Kai Zhu,[†] Michael R. Shirts,[†] Richard A. Friesner,[†] and Matthew P. Jacobson*[,‡]

Department of Chemistry and Center for Biomolecular Simulation, Columbia University, New York, New York 10027, and Department of Pharmaceutical Chemistry, University of California, San Francisco, California 94158-2517

Received April 7, 2006

Abstract: We optimize a truncated Newton (TN) minimization algorithm and computer package, TNPACK, developed for macromolecular minimizations by applying multiscale methods, analogous to those used in molecular dynamics (e.g., r-RESPA). The molecular mechanics forces are divided into short- and long-range components, with the long-range forces updated only intermittently in the iterative evaluations. This algorithm, which we refer to as MSTN, is implemented as a modification to the TNPACK package and is tested on energy minimizations of protein loops, entire proteins, and protein−ligand complexes and compared with the unmodified truncated Newton algorithm, a quasi-Newton algorithm (LBFGS), and a conjugate gradient algorithm (CG+). In vacuum minimizations, the speedup of MSTN relative to the unmodified TN algorithm (TNPACK) depends on system size and the distance cutoffs used for defining the short- and long-range interactions and the long-range force updating frequency, but it is 4 to 5 times greater in the work reported here. This algorithm works best for the minimization of small portions of a protein and shows some degradation (speedup factor of 2−3) for the minimization of entire proteins. The MSTN algorithm is faster than the quasi-Newton and conjugate gradient algorithms by approximately 1 order of magnitude. We also present a modification of the algorithm which permits minimizations with a generalized Born implicit solvent model, using a self-consistent procedure that increases the computational expense, relative to a vacuum, by only a small factor (∼3−4).

Introduction

Minimization is a core functionality in protein molecular mechanics programs. Minimization of a protein structure taken from the Protein Data Bank (PDB) is a standard step in nearly every modeling effort, and an intrinsic component of many conformational search algorithms (e.g., Monte Carlo plus minimization[1−3]). Consequently, an algorithm that can substantially reduce the computational effort for minimization can have a major impact on studies of protein structure and protein−ligand interactions.

Efforts to optimize minimization algorithms have focused primarily on the algorithm that determines the size and direction of the geometry steps. These algorithms differ in the details of how they obtain a search step, and also what kind of information they require. For example, gradient-based methods only require the first derivatives, while Newton-type methods require the second derivatives. A number of different approaches exist, such as conjugate gradient,[4] Broyden−Fletcher−Goldfarb−Shanno (BFGS),[5] and truncated Newton (TN),[6−12] all of which have been shown to have advantages and disadvantages depending upon the context (for example, the size of the molecule to be optimized and how far one is from the minimum). However, less work

* Corresponding author e-mail: matt@cgl.ucsf.edu.

[†] Columbia University.

[‡] University of California.

10.1021/ct600129f CCC: $37.00
Published on Web 02/14/2007

has been done in attempting to reduce the cost of the energy/gradient evaluations required at each step. This is in contrast to molecular dynamics simulations where extensive effort has gone into reducing the computational effort per step, via such techniques as the multiple time scale method[13,14] and fast multipole method.[15]

In the case of minimization, the issue of reducing computational effort per step is ultimately tied to which algorithm should be used to generate the geometry steps. Newton−Raphson-type algorithms that utilize second derivatives usually have a faster convergence rate while requiring more storage space. A key issue is whether second-derivative information can be made inexpensive enough to use profitably for large systems. Brute force calculation of all second derivatives, followed by inversion of the Hessian to rigorously solve the Newton−Raphson equations, is grossly inefficient when dealing with the thousands or tens of thousands of coordinates present in a typical protein minimization problem. However, more sophisticated methods such as truncated Newton, in conjunction with preconditioned iterative algorithms for solving the approximate Newton−Raphson equations defined by the truncation, are much more promising. A novel implementation of the truncated Newton method, TNPACK of Schlick and co-workers,[8−12] has shown significant speed advantages for the minimization of macromolecular systems. Its protocol for advancing the geometry involves both an inner loop (iterative solution of the truncated Newton−Raphson equations) and an outer loop (each consisting of one truncated Newton−Raphson step, at which the energy and gradient have to be calculated). The use of even approximate second derivatives can dramatically reduce the required number of outer iterations as compared to competitors such as quasi-Newton methods or conjugate gradient methods that do not utilize such information.[10,16] Solving the Newton−Raphson equations inexactly also reduces the computational expense while retaining enough accuracy and thus rapid convergence. Thus, the critical issue is the cost of energy and gradient evaluations; if these can be done with sufficiently inexpensive approximations, very large gains as compared to alternative approaches can be realized.

In this paper, we present an integrated approach to the optimization of the truncated Newton approximation as applied to protein minimization. We examine both local minimization (in which a region of the protein is kept fixed—this situation arises frequently, for example, in modeling active sites and structure prediction of the side chains and loops) and global minimization of the entire protein. The basic idea is to utilize ideas developed in the context of molecular dynamics simulations, principally, multiple time- and length-scale approximations (which we refer to, compactly, as a "multiscale" methodology), to accelerate truncated Newton minimizations. Each aspect of the truncated Newton technology is examined in detail, and an optimized set of approximations is designed. Performance is evaluated for a substantial number of test cases by comparison with alternative approaches, as well as with that of the original TNPACK employed without the enhancements described herein.

In addition to presenting a methodology for optimization of a molecular mechanics energy function in the gas phase, we also adapt our approach to the minimization of a system in a continuum solvent, specifically, the surface generalized Born (SGB) model[17,18] that we have described previously in a number of publications.[19−24] The approximations required to handle the continuum solvation calculation efficiently, while employing the same general principles as in the gas phase, are notably different in details.

Methods

Multiscale Truncated Newton Minimization Algorithm. All Newton-type methods are based on approximating the objective function locally by assuming a quadratic model and minimizing or approximately minimizing that model. We denote by x_k the current approximation to the solution vector $\mathbf{x^*}$, and by g_k and H_k the gradient and Hessian evaluated at x_k, respectively. The new estimate for x_k is then obtained from the Taylor series expansion, up to quadratic terms, along a search direction p_k:

$$E(x_k + p_k) \approx E(x_k) + g_k^T p_k + \frac{1}{2} p_k^T H_k p_k$$

To find the search direction p_k, we find the minimum of the quadratic function by solving the equation

$$H_k p_k = -g_k$$

Subsequently, a line search or trust region method is applied to find the appropriate step length λ_k and generate the next iterate according the following equation:

$$x_{k+1} = x_k + \lambda_k p_k$$

Thus, given a starting point x_0, a series of iterates $\{x_0,x_1,...,x_k,x_{k+1},...\}$ will be generated until some convergence criterion is satisfied.

The truncated Newton method distinguishes itself from other Newton-type methods by solving the Newton equation inexactly. Formally, a truncation criterion is applied, as in TNPACK:[10]

$$||H_k p_k + g_k|| \leq \Phi_k ||g_k||$$

This truncation criterion is easily satisfied at regions distant

$$\Phi_k = \min\left\{\frac{1}{k}, \left|\left|g_k\right|\right|\right\}$$

from the local minimum; as a local minimum is approached, the condition becomes more stringent, and this leads to increasingly accurate solutions for the search direction. The truncation is justified by the approximate nature of the Newton method at each iteration.

Figure 1 schematically depicts the TN algorithm. Each iteration of the outer loop serves to choose a search direction and make a move along it. The inner loop solves for the search direction using a preconditioned conjugate gradient (PCG) algorithm, which is also an iterative process. Once the search direction is chosen, a one-dimensional line search iteratively determines the step size along the search direction.

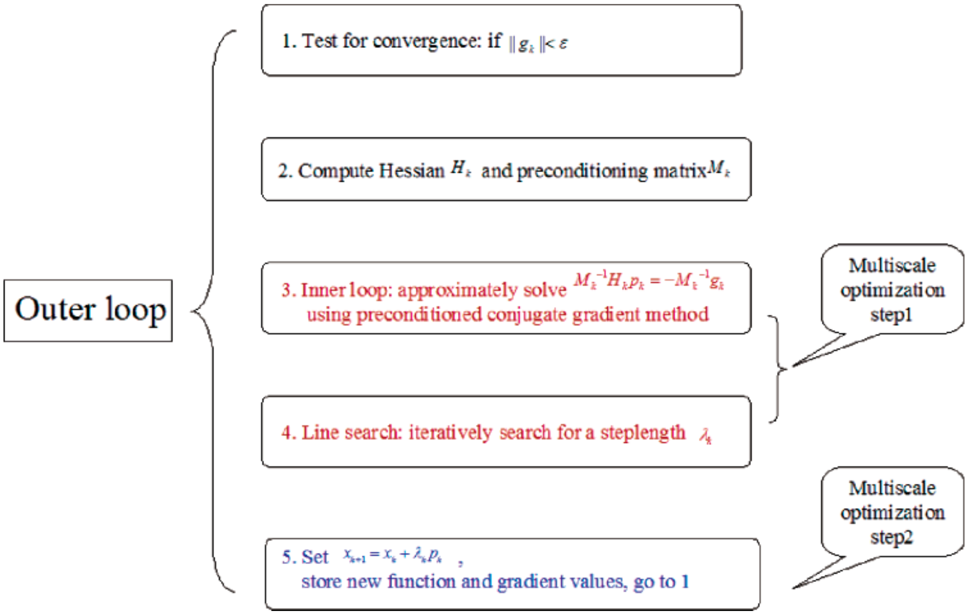
Outer loop
1. Test for convergence: if $\| g_k \| < \varepsilon$
2. Compute Hessian H_k and preconditioning matrix M_k
3. Inner loop: approximately solve $M_k^{-1} H_k p_k = -M_k^{-1} g_k$ using preconditioned conjugate gradient method
4. Line search: iteratively search for a steplength λ_k
5. Set $x_{k+1} = x_k + \lambda_k p_k$, store new function and gradient values, go to 1
Multiscale optimization step1
Multiscale optimization step2

Figure 1. Schematic of the multiscale optimization of the truncated Newton implementation. Multiscale optimization step 1 reduces the energy and gradient evaluation costs of the PCG inner loop and line search, while step 2 optimizes those of the outer loop.

The key innovation in our multiscale optimization is the division of the molecular mechanics interactions into short- and long-range components, in analogy to multiscale molecular dynamics methods such as RESPA.[13,14] That is,

$$E = E_s + E_l$$

$$g = g_s + g_l$$

$$H = H_s + H_l$$

(Subscripts 1 and s refer to the long and short components, respectively.) The short-range forces contain all covalent interactions and nonbonded interactions between atoms separated by small distances, and the long-range interactions include all other nonbonded interactions. The detailed partitioning of the interactions is discussed below. This division can significantly improve efficiency because (1) the slowly varying long-range components can be updated less frequently than short-range components and (2) the long-range components can require significantly greater computational expense than the short-range components, depending on the distance cutoffs that are used. Specifically, we invoke the following approximations:

- $H_l = 0$. The TN algorithm uses only a sparse approximation to the Hessian to aid convergence in the PCG inner loop. We use only the covalent terms and 1−4 interactions from the molecular mechanics force field in the Hessian preconditioner.
- The long-range component of the gradient, g_l, is updated infrequently, as described in detail below.
- The long-range component of the energy, when it is not computed directly, is updated from the long-range component of the gradient according to $E_l = g_l'(x - x') + E_l'$, where x', g_l', and E_l' represent the coordinates, gradient, and long-range energy, respectively, from the most recent update of the long-range interactions.

We apply the multiscale optimization to all components of the TN algorithm that require iterative evaluations of the energy and gradient. In the PCG inner loop and the line search, we never update the long-range gradient. The PCG inner loop only approximately solves for the Newton−Raphson search direction, regardless of any approximations to the energies and gradient. Likewise, the line search only approximately minimizes the energy, using truncation criteria to terminate the one-dimensional search. Thus, the additional approximations invoked by assuming that the long-range gradient is constant can in principle affect the number of outer steps required for convergence but do not affect accuracy. It should also be noted that the step size in the line search becomes small as the minimization converges on a local minimum, which means that approximating the long-range gradient as constant in the inner loop and line search becomes increasingly accurate. In the outer loop, we update the long-range energy and gradient periodically; the number of outer steps between updating the long-range forces is an adjustable parameter, discussed further below.

It is noted that TNPACK uses the local energy terms to approximate its Hessian preconditioner in the PCG inner loop. In some sense, this is similar to our approximate treatment of gradient and energy in the PCG inner loop. However, there is a significant distinction between these two approximations. The preconditioner is only a tool for accelerating convergence and does not alter the function and gradient values being provided to define the minimization task in the linear system, while the multiscale approximation does. Thus, standard convergence criteria that guarantee convergence for the original algorithm cannot be directly applied, and behavior can only be assessed in practice in comparison to the unmodified TNPACK, as well as other minimization algorithms, as we discuss below. In principle, it would be possible to use the multiscale approximation in early parts of the minimization but revert to unmodified TNPACK as a minimum is approached. However, in practice, we have found this to be unnecessary, at least with a convergence criterion of 0.001 kcal/mol/Å root-mean-squared gradient (RMSG).

Clearly, the performance of this optimization strategy depends on the definition of long- and short-range forces and the updating frequency of the long-range energy and gradient in the outer loop. The overall efficiency is a balance between minimizing the computational expense associated with the energy evaluations, by treating as many nonbonded interactions as possible as "long-range" and updating these infrequently, and minimizing the number of outer loop iterations, which will increase if the approximations employed in the energy evaluations are too severe. We have addressed this tradeoff by empirically optimizing the updating frequency for the long-range forces and the division of molecular mechanics forces into short- and long-range. In this work, we use residue-based cutoffs for distinguishing between short- and long-range nonbonded interactions. Short-range forces include all bond, angle, torsion, and 1−4 nonbonded interactions. The nonbonded interactions are partitioned into short- and long-range using distance cutoffs which depend on the amino acid types involved in the interaction. We employ an absolute cutoff of 30 Å for charged−charged residue pairs, 20 Å for charged−neutral, and 15 Å for neutral−neutral, with no smoothing. That is, all interactions beyond these distance cutoffs are ignored, because they are small. The distance cutoffs for partitioning the three types of nonbonded interactions into short- and long-range are adjustable, as discussed below. The all-atom optimized potential for liquid simulations (OPLS) force field[25,26] is used for all tests performed here. Parameters for the ligands were obtained using atom-typing capabilities provided in IMPACT.[27]

Our baseline implementation of the minimization algorithm TNPACK (Algorithm 702 in the ACM Digital Library)[7−12] uses default values for most parameters unless otherwise noted. Specifically, we utilize a residual-based truncation criterion of 0.25 for the inner PCG loop and the unconventional modified Cholesky factorization. Hessian-vector products are obtained by finite difference. We assess convergence of the minimization exclusively using the root-mean-squared gradient, calculated over all degrees of freedom included in the minimization. All minimizations are performed in Cartesian coordinates in double precision. The multiscale truncated Newton (MSTN) implementation made minimal modifications to the TNPACK Fortran source code to enable the multiscale optimization but otherwise used unmodified TNPACK with the same parameters.

Other Minimization Algorithms Used. In addition to assessing the impact of our multiscale modifications to truncated Newton, we also make compararisons to a quasi-Newton minimization algorithm (LBFGS)[5] and a conjugate gradient algorithm (CG+).[4] For these two software packages, we use the default parameters and do not attempt further optimization. The quasi-Newton algorithm is used, retaining information from the prior seven steps, and without preconditioning. The conjugate gradient algorithm is used with a positive Pollak−Ribiere update.

Minimization with Generalized Born Solvent. The generalized Born (GB) implicit solvation model is well-suited for performing rapid minimizations because the solvent-induced screening between a pair of charges is treated using an analytical formula:

$$\Delta G_{ij} = -\frac{1}{2}\left(1 - \frac{1}{\epsilon}\right)\frac{q_i q_j}{\sqrt{{r_{ij}}^2 + {\alpha_{ij}}^2\, e^{-D_{ij}}}}$$

where q_k is the partial charge on atom k, ϵ is the dielectric constant of the solvent, α_k is the Born α radius for atom k, r_{ij} is the distance between a pair of atoms i and j, $\alpha_{ij} = \sqrt{\alpha_i \alpha_j}$, and $D_{ij} = r_{ij}^2/(2\alpha_{ij})^2$. The Born α's can be computed in a variety of ways, including analytical expressions and surface or volume integration, depending on the specific implementation.

The GB pair term is of course differentiable with respect to atomic coordinates, but the resultant expression for the gradient involves derivatives of the Born α's with respect to the atomic coordinates, which must be determined numerically for GB models that calculate the Born α's by a grid-based integration. Because these derivatives are expensive to compute, we employ a self-consistent minimization, in which the Born α's are held fixed during the course of the minimization, then updated prior to another minimization, and so on until the energy ceases to decrease by more than 1 kcal/mol. This threshold is precise enough for most protein modeling purposes. In practice, self-consistency rarely requires more than two to three cycles of TN minimization, and the second and subsequent minimizations are generally very fast.

The GB implementation used here is based on the surface generalized Born implementation of Ghosh et al.[18] The updating of Born α's is quite expensive and can be the bottleneck of the simulation if there are a large number of conformations that need to be energy-minimized. In cases where only a portion of the protein or protein−ligand complex is minimized, our implementation updates only those portions of the surface and surface integrals that change.

The GB pair terms are partitioned into the short- and long-range interactions using the same criteria described above for the nonbonded interactions. The GB pair terms corresponding to atoms separated by three bonds or less are also included in the sparse Hessian preconditioner.

Results and Discussion

Test Set. One of the simplest macromolecular applications of a minimization algorithm is to simply energy-minimize an entire protein, for example, in preparation for molecular dynamics. We have chosen a set of 20 proteins for such a test, with 1162−5929 atoms. The PDB codes for these proteins are 1HOE, 2HSP, 1ECI, 1CCD, 1J8B, 3CRD, 1AB2, 1HPW, 1BUO, 1HT9, 1BZS, 1BHD, 1EJF, 1HYT, 1NJ4, 1CVM, 1DIM, 1EM2, 1AXN, and 2IF1. However, in the context of homology modeling and other modeling applica-

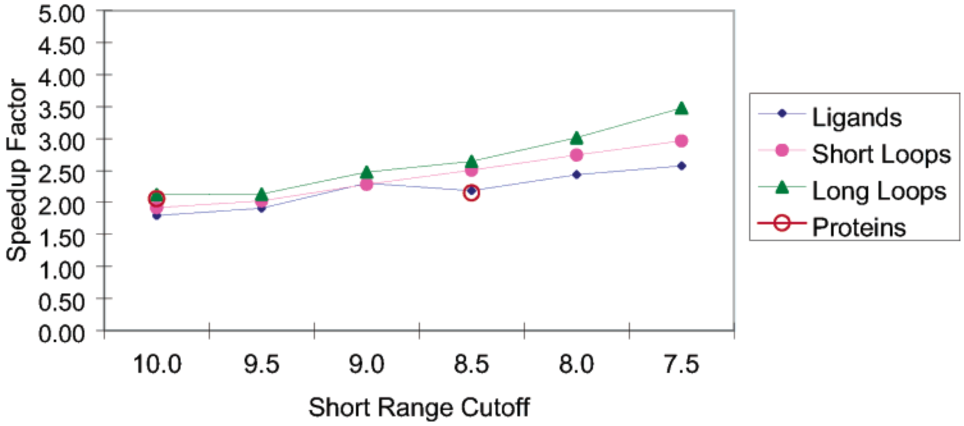

Figure 2. Average speedup factors of MSTN, with the multiscale optimization applied only to the inner loop, relative to unmodified TN on the minimization of ligands, short loops, long loops, and full proteins. The decrease in computational expense due to inner loop optimization is related primarily to the ratio of computational expense for determining the short- and long-range forces, which in turn is determined by the distance cutoff for partitioning the nonbonded interactions into short- and long-range. In these tests, the minimizations are considered converged when the RMS gradient decreases below 0.001 kcal/mol/Å. Details of the test cases used can be found in Tables 2 and 3 and in the Supporting Information.

tions, it is also common to energy-minimize relatively small portions of a protein, ranging from single side chains to loops or secondary structure elements, that is, in the context of side chain,[19,20] loop,[21,22] and helix[24] optimization. For this reason, we have also chosen a set of 25 proteins loops with 13-residue lengths from our previous work.[22] To study the possible effects of loop size, we also cut these loops in half and composed another 25 protein loops with six-residue lengths. Finally, in the context of small molecule docking and the estimation of protein−ligand relative binding affinities,[23,36−38] it is frequently desirable to energy minimize a ligand in a rigid protein receptor, and we have performed such a test on 20 ligands in a single protein receptor. Altogether, the test cases range from 51 to 17 787 degrees of freedom. All computations are performed using single PIII 1.4 GHz processors.

Inner Loop versus Outer Loop Optimization. As discussed in the Methods section, the multiscale optimization is applied in both the inner and outer loops of the truncated Newton algorithm. To assess the effects of our optimization, we perform it in two steps. In the first step, we apply the multiscale optimization only to the PCG inner loop and line search, where the long-range interactions are never updated, as discussed in the Methods section. In the second step, we also apply the multiscale optimization to the outer loop by only updating the long-range forces periodically. This is the full optimization, denoted by MSTN. We evaluate the decrease in computational expense to reach a local minimum relative to the unmodified truncated Newton minimization, which is denoted by TN. In this work, we use exclusively the RMSG as the convergence criterion for the minimization (0.01, 0.001, and 0.0001 kcal/mol/Å in various tests).

The application of the multiscale optimization to the inner loop reduces the cost per Newton−Raphson step. As shown in Figure 2, the decrease in computational expense due to inner loop optimization is related primarily to the ratio of computational expense for determining the short- and long-range forces, which in turn is determined by the distance cutoff for partitioning the nonbonded interactions into short- and long-range. For the ligand, short loop, and long loop test sets, the decrease in computational expense improves as this distance cutoff decreases, down to 7.5 Å. Below 7.5 Å, many minimizations have serious convergence problems. That is, the approximation that the search direction and line search stepsize can be determined solely by the short-range interactions begins to break down. For the full protein minimizations, the convergence problems become serious when the shor/long-range cutoff is decreased below 8.5 Å, as discussed further below. In all of the following results, the short/long-range cutoff is fixed to 7.5 Å for the ligand, short loop, and long loop test sets and 10 Å for the full protein minimizations.

The application of the multiscale optimization to the inner loop accounts for most of the reduction in the computational expense of MSTN relative to TN. As shown in Figure 3, some additional decrease in computational expense can be obtained by also only periodically updating the long-range forces in the outer loop. However, when the long-range gradient is updated less frequently than once every five outer iterations, some of the ligand and loop minimizations start to converge more slowly. Thus, we use an outer-loop updating frequency of every five steps in the remaining MSTN results reported here. The full protein minimizations require more frequent updating to remain stable, and we use an updating frequency of two in the results below.

Computational Expense of MSTN Relative to TN and Other Minimization Algorithms. The results using the optimized set of parameters are summarized in Table 1. Details of the test cases are reported in Tables 2 and 3 and in the Supporting Information. The effect of the multiscale optimization is to decrease the computational expense of the truncated Newton minimization by a factor of approximately 4 for the ligand, short loop, and long loop minimizations. The acceleration is somewhat less for full proteins, as discussed in more detail below.

It is known that Newton-type minimization algorithms have a quadratic convergence rate; that is, they converge much more quickly as they approach a local minimum.

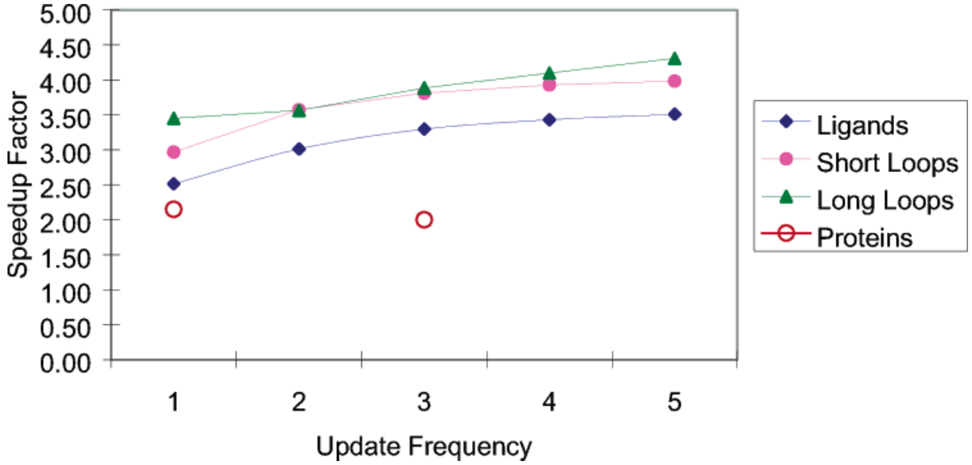

Figure 3. Speedup factors of MSTN, with the multiscale optimization applied to both the inner and outer loops, relative to unmodified TN on the minimization of ligands, short loops, long loops, and full proteins. The decrease in computational expense when applying the multiscale optimization to the outer loop is determined primarily by the number of outer-step iterations between updates to the long-range gradient ("update frequency" on the *x* axis). In these tests, the minimizations are considered converged when the RMS gradient decreases below 0.001 kcal/mol/Å, and the distance cutoff for partitioning the nonbonded interactions into short- and long-range is kept fixed at 7.5 Å (8.5 Å for proteins, see text for discussions). Details of the test cases used can be found in Tables 2 and 3 and in the Supporting Information.

Table 1. Speedup Factors of MSTN Relative to Unmodified TN, LBFGS, and CG+ with Different Convergence Criteria (Defined by the Root-Mean-Squared Gradient, in Units of kcal/mol/Å)

	different convergence criteria of RMSG (kcal/mol/Å)								
	TN/MSTN			QN/MSTN			CG/MSTN		
	0.01	0.001	0.0001	0.01	0.001	0.0001	0.01	0.001	0.0001
ligands	3.8	3.5	3.6	14.3	14.9	15.0	14.5	17.2	19.5
short loops	4.0	4.0	4.0	20.0	21.2	22.9	17.2	22.0	27.9
long loops	3.8	4.3	4.3	12.6	13.8	16.2	14.1	17.5	25.1
proteins	2.6	2.0	N/A	2.5	2.5	N/A	3.6	5.1	N/A

Truncated Newton methods, although they solve the Newton−Raphson equations approximately, largely retain this advantage. Although our optimization introduces additional approximations that alter the function and gradient, and hence standard convergence results cannot be applied, Table 1 shows that the speedup factors of MSTN do not vary significantly when three different convergence criteria, from loose to tight, are used. We can see from the table that MSTN has rather stable speedup factors and does not degrade even when the tightest convergence criterion of 0.0001 kcal/mol/Å root-mean-squared gradient is used.

The acceleration of MSTN relative to CG+ (hereafter CG) and LBFGS (hereafter QN) minimization, using the same energy function, is at least 1 order of magnitude for the ligand and loop test sets and is a somewhat more modest factor of 3−5 for the full proteins, as shown in Table 1. The QN minimization is generally somewhat faster than CG, and this advantage is more prominent when the convergence criterion is tighter, that is, 0.0001 kcal/mol/ Å.

Effects of Multiscale Optimization Other Than Decreasing Computational Expense. Up to this point, we have focused exclusively on the total CPU time required to converge to a local minimum. The number of iterations (outer loops) of the TN minimization can give a sense of whether the approximations employed in MSTN affect the rate of convergence (although the overall computational expense is of course less). As shown in Table 2 and the Supporting Information, for the ligand and loop minimizations, the numbers of iterations to reach convergence for TN and MSTN are nearly identical, indicating that not updating the long-range forces in the inner loop does not strongly affect convergence. A comparison of the corresponding minimization trajectories also shows that they are nearly identical. Full protein minimizations are discussed below.

Another question is whether MSTN finds different local minima than TN, when starting from the same initial conformation. It is important to emphasize that identifying the same local minima is not a requirement for the success of MSTN. It is well-known that different algorithms often converge to different local minima if the energy surface is very complex, even when starting from the same point. For example, the truncated Newton method can identify different local minima than those found by the full Newton method for a complex energy function. Indeed, in our tests, minimizations starting from identical initial states can diverge when running on different processors or with executables built with different compilers (this is true for several different minimization algorithms including TN).

For the ligand minimizations, the optimized MSTN and original TN algorithms identify minima with almost exactly the same final energy and structures. Usually, the minimized structures differ from each other only in the second or third

Table 2. Comparison of MSTN with TN on Short Loops in Terms of Number of Iterations, Final Energy, Final Gradient (kcal/mol/Å), ΔE (kcal/mol), and RMSD (Å)[a]

			starting grams	time (s)		iterations		final gradient		ΔE (kcal/mol)	RMSD (Å)
				TN	MSTN	TN	MSTN	TN	MSTN		
1ojq	A:167	A:172	8.38	20.44	5.04	36	37	0.00001	0.00001	0.00	0.01
1dpg	A:352	A:357	7.10	6.26	1.52	16	16	0.00001	0.00056	0.01	0.02
1xyz	A:645	A:650	8.82	6.85	2.67	18	27	0.00051	0.00008	0.01	0.01
1eok	A:147	A:152	9.71	1.98	0.51	9	9	0.00019	0.00007	0.00	0.01
1p1m	A:327	A:332	7.93	9.26	2.66	19	20	0.00025	0.00014	0.00	0.00
1ock	A:43	A:58	7.06	30.17	7.09	25	27	0.00022	0.00002	0.00	0.00
1hnj	A:191	A:196	7.65	6.35	1.44	27	27	0.00015	0.00011	0.14	0.06
1o6l	A:386	A:391	7.30	19.15	3.64	22	19	0.00002	0.00064	0.00	0.01
1bkp	A:51	A:56	15.47	5.16	1.33	15	16	0.00033	0.00004	0.00	0.01
1f46	A:64	A:69	7.91	1.88	0.43	10	9	0.00004	0.00015	0.00	0.00
1jp4	A:153	A:158	13.42	4.77	1.37	15	16	0.00019	0.00004	0.03	0.03
1nln	A:26	A:31	15.60	8.28	2.21	18	16	0.00010	0.00063	0.00	0.00
1kbl	A:793	A:798	9.72	9.83	2.11	19	18	0.00002	0.00053	0.00	0.00
1l8a	A:691	A:696	8.28	22.80	5.55	31	32	0.00015	0.00007	0.07	0.03
1cnv	_:110	_:115	8.56	23.25	5.47	42	41	0.00002	0.00015	0.00	0.00
1mo9	A:107	A:112	9.75	7.17	2.13	18	21	0.00045	0.00005	0.01	0.01
1gpi	A:308	A:313	13.30	3.50	0.84	21	17	0.00041	0.00082	0.00	0.00
1lki	_:62	_:67	7.96	4.47	1.03	17	17	0.00011	0.00037	0.00	0.01
1qqp	2:161	2:166	10.11	2.81	0.70	10	10	0.00015	0.00019	0.00	0.00
1qs1	A:389	A:394	7.06	12.56	2.91	16	15	0.00012	0.00002	0.00	0.01
1d0c	A:280	A:285	11.47	4.67	1.28	21	22	0.00046	0.00009	0.00	0.00
1krh	A:131	A:136	7.92	4.01	1.00	16	15	0.00006	0.00030	0.02	0.02
2hlc	A:91	A:96	9.20	2.42	0.62	11	11	0.00009	0.00006	0.00	0.00
1ako	_:203	_:208	7.36	4.25	1.21	23	23	0.00006	0.00060	0.03	0.00
1ed8	A:67	A:72	15.17	2.63	0.71	10	10	0.00006	0.00007	0.00	0.00

[a] ΔE is defined as the energy of the MSTN structure minus the energy of the corresponding TN structure. The RMSD is calculated using all heavy atoms of the loop region.

decimal places of the Cartesian coordinates. For the loops, the final energies are usually slightly different and, occasionally, several kilocalories per mole for long loops. Table 2 shows a summary for the short loop test set. We calculate the root-mean-square deviation (RMSD) between the local minima identified by MSTN and TN and find that these RMSDs are mostly negligible. A few structures in the long loop set have significant RMSDs up to 0.5∼1 Å, and these minimizations also have quite different final energies.

To quantify the divergence of MSTN and TN, we also compare the final energies and structures minimized by LBFGS and CG+ with TN, as shown in the Supporting Information. Similar to the comparison of MSTN and TN, LBFGS and CG+ frequently converge to the same local minima as TN. The number of cases where significant divergence is seen (i.e., different local minima as indicated by different final energies and RMSDs) between LBFGS/CG+ and TN is about the same as that in the comparisons of MSTN and TN, but the magnitude of the divergence is often greater.

Minimization of Full Proteins. The minimizations of MSTN on full proteins are quite different compared with the ligand and loop minimizations. MSTN and TN minimizations seldom converge to the same energy minima; they can take very different numbers of iterations, and the corresponding minima have significant RMSDs. This is due to the fact that the energy surface of an entire protein is so complex enough that there are many available local minima. On average, the MSTN minimization reduces the computational expense by a nontrivial factor (roughly 2), but for a small number of cases, the MSTN minimization actually requires greater CPU time to converge than unmodified TN. Some minimizations fail at the line search because no acceptable step size can be found. This situation also happens with unmodified TN, but less frequently. Simply restarting the minimization solves this problem. To obtain a meaningful comparison, we add the two times together.

Altogether, it is clear that minimizations of entire macromolecules cannot be subjected to the same level of multiscale optimization as minimizations of relatively small portions of the macromolecule, or small molecules interacting with macromolecules. We note that the advantage of unmodified TN relative to less sophisticated minimization algorithms such as CG and LBFGS is also less for full protein minimizations than for minimizations with a smaller number of degrees of freedom. As Table 1 shows, unmodified TNPACK is 3−5 times faster than LBFGS and CG+ for the test sets of ligands and loops, but it is only slightly faster than LBFGS and 1−2 times faster than CG+ for entire proteins. We speculate that this behavior may relate to the way the convergence properties of the PCG inner loop vary with the number of degrees of freedom or the complexity of the function being minimized.

Crystal Environment. We have applied MSTN in a series of applications, including side-chain optimization,[19,20] loop prediction,[21,22] helix prediction,[39] and estimation of protein−

Table 3. Minimization of Ligands in Generalized Born Solvent[a]

ligand (KEGG ID)	degrees of freedom	iterations for ddd	time for ddd(s)	iterations for SGB	pure min (s)	α update (s)	SGB total/ddd
C02717	126	12	0.85	15	1.80	1.11	3.40
C09321	87	13	0.54	16	0.96	0.86	3.35
C04303	156	29	1.34	24	3.65	1.17	3.60
C02006	102	16	0.65	19	1.33	1.00	3.61
C00355	75	9	0.26	24	1.16	0.87	7.70
C01302	108	51	2.93	22	2.39	1.04	1.17
C05364	51	7	0.17	10	0.39	0.77	6.97
C05662	63	10	0.28	13	0.60	0.81	4.95
C00852	126	23	1.33	37	4.01	1.38	4.07
C04771	72	8	0.20	10	0.42	0.87	6.35
C01850	123	10	0.62	15	1.35	1.09	3.92
C03300	138	33	2.44	18	1.79	1.14	1.20
C05983	132	18	1.01	21	1.86	1.11	2.94
C05401	105	11	0.48	27	2.12	0.94	6.41
C06019	81	18	0.76	30	2.17	0.84	3.96
C09126	153	46	2.91	62	6.55	1.49	2.77
C04711	93	10	0.34	14	0.86	0.97	5.48
C04194	222	31	5.57	20	4.89	1.23	1.10
C06585	78	14	0.37	16	0.55	0.65	3.22
C04498	123	8	0.43	10	0.86	0.92	4.12
Average							4.02

[a] The ligands are docked into the binding site of methylaspartate ammonia lyase (PDB ID: 1kkr) using Glide[40−42] and then minimized to a 0.001 kcal/mol/Å RMS gradient. The ligands are taken from the Kyoto Encyclopedia of Genes and Genomics (KEGG) database, which contains metabolite ligands, toxins, inhibitors, and pollutants.[43] The times are divided into the computational expense of the minimization steps (pure min) and the expense of updating the Born α's between each minimization in the self-consistent procedure (α update). The right column is the ratio of the total computational expense of minimization in a GB solvent versus in a vacuum.

ligand relative binding affinities.[23,36−38] In some of these applications, we have explicitly modeled the crystal environment in order to make a more direct and realistic comparison to the structure determined by X-ray crystallography. In these applications, we create and update the symmetric copies according to the space group information of the PDB file for every putative candidate during the minimization. The inclusion of crystal packing effects in this way leads to a larger computational cost, because the nonbonded energy evaluations extend over multiple copies of the asymmetric unit. Therefore, it is interesting to see the performance of MSTN compared with TN when the crystal environment is included. Using a convergence criterion of 0.001 kcal/mol/Å, the average speedup factors of MSTN2 on short and long loops are 5.0 and 5.4, respectively. Without the crystal environment, these two factors are 4.0 and 4.3, respectively. The increased speedup factors clearly occur because inclusion of the crystal packing environment increases the number of long-range interactions more greatly than the number of short-range interactions.

Minimizations in Generalized Born Implicit Solvent. The computational costs of the minimizations in implicit solvent are divided into the cost of the minimization steps and the cost of updating the Born α's between iterations of the self-consistent procedure. The latter, which requires updating and integrating over a large portion of the protein surface, can be nontrivial compared with the energy evaluations. For the four test sets, the total computational costs of minimization in implicit solvent are, on average, 4.0, 3.2, 2.7, and 2.4 times greater than in a vacuum. Table 3 shows the details for the ligand minimizations in a SGB solvent; the results for the other tests sets are shown in the Supporting Information. It is important to keep in mind that the minimizations in a vacuum and implicit solvent do not converge to the same local minimum; that is, the energy surface is different. The computational cost of each force evaluation is approximately 50% greater when using GB, because of the additional cost of calculating the pair term. The overall computational expense of the minimization steps in GB solvent is greater than this largely because of the cost of updating the Born α's.

Conclusions

We have demonstrated a simple method for accelerating macromolecular minimizations on the basis of the partitioning of forces into short- and long-range components. As with similar multiscale methods applied to molecular dynamics, the long-range forces are updated less frequently than the short-range forces. In principle, this simple idea can be applied to several different classes of gradient-based minimization algorithms, but we have focused on implementing it for the powerful truncated Newton method as implemented in the TNPACK package. The acceleration of our multiscale implementation depends on the exact partitioning of the forces into short- and long-range components and updating strategy. In the work reported here, the speedup relative to truncated Newton without the multiscale implementation is about 4−5 for a number of systems ranging from ligands to 13-residue-long loops. For entire proteins, this algorithm works less well but still shows a speedup factor of roughly 2. We also compare to commonly used conjugate gradient and quasi-Newton methods, using the same energy function but no multiscale implementation; the MSTN method is faster by 1 order of magnitude.

Finally, we have implemented a self-consistent procedure for minimizations in a generalized Born implicit solvent, which increases the computational expense, relative to a vacuum, by only a factor of ∼3. This self-consistent procedure can be understood as another multiscale approximation, which decomposes the solvent "forces" in the generalized Born model into rapidly varying (short-range pair term) and slowly varying (long-range pair term and self-term) components.

We have already deployed the MSTN technology in multiple applications, including side-chain optimization,[19,20] loop prediction,[21,22] helix prediction,[39] and estimation of protein−ligand relative binding affinities.[23,36−38] In each of these applications, we use the same energy function as in the work reported here and enumerate hundreds or thousands of local minima on the energy surface. Minimization is a rate-limiting step in each application, and the efficiency of the MSTN algorithm has been critical to the success of these works. Further optimization of the MSTN method may be possible through more rigorous theoretical and empirical studies of the convergence properties than we have attempted here.

Acknowledgment. This work was supported by NSF Grant 0346399 to M.P.J. and by NIH Grants GM52018 to R.A.F. and GM071790 to M.P.J. M.P.J is a consultant to Schrödinger, Inc.

Supporting Information Available: Detailed computational data for each of the 40 test cases, including comparisons of MSTN with TN, LBFGS, and CG+ in terms of computational time, number of iterations, final structures, final energies, and final gradients; detailed comparisons of MSTN in a vacuum and in a generalized Born implicit solvent for each test case. This information is available free of charge via the Internet at http://pubs.acs.org.

References

(1) Li, Z.; Scheraga, H. A. *Proc. Natl. Acad. Sci. U.S.A.* **1987**, *84*, 6611−6615.

(2) Pillardy, J.; Arnautova, Y. A.; Czaplewski, C.; Gibson, K. D.; Scheraga, H. A. *Proc. Natl. Acad. Sci. U.S.A.* **2001**, *98*, 12351−12356.

(3) Trosset, J. Y.; Scheraga, H. A. *Proc. Natl. Acad. Sci. U.S.A.* **1998**, *95*, 8011−8015.

(4) Gilbert, J. C.; Nocedal, J. *SIAM J. Optimiz.* **1992**, *2*, 21.

(5) Liu, D. C.; Nocedal, J. *Math. Prog.* **1989**, *45*, 503−528.

(6) Dennis, J. E. J.; Schnabel, R. B. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*; Prentice Hall: Englewood Cliffs, NJ, 1983; pp 239−255.

(7) Schlick, T. Optimization Methods in Computational Chemistry. *Reviews in Computational Chemistry*; Lipkowitz, K. B., Boyd, D. B., Eds.; VCH Publisher: New York, 1992; Vol. III, pp 1−71.

(8) Schlick, T. *SIAM J. Sci. Stat. Comput.* **1993**, *14*, 424−445.

(9) Schlick, T.; Fogelson, A. *ACM T. Math. Software* **1992**, *18*, 141.

(10) Schlick, T.; Overton, M. *J. Comput. Chem.* **1987**, *8*, 1025−1039.

(11) Xie, D.; Schlick, T. *ACM T. Math. Software* **1999**, *25*, 108−122.

(12) Xie, D.; Schlick, T. *SIAM J. Optimiz.* **1999**, *10*, 132−154.

(13) Tuckerman, M.; Berne, B. J.; Martyna, G. J. *J. Chem. Phys.* **1991**, *94*, 6811−6815.

(14) Tuckerman, M.; Berne, B. J.; Martyna, G. J. *J. Chem. Phys.* **1992**, *97*, 1990−2001.

(15) Greengard, L.; Rokhlin, V. *J. Comput. Phys.* **1997**, *135*, 280−292.

(16) Derreumaux, P.; Zhang, G.; Brooks, B.; Schlick, T. *J. Comput. Chem.* **1994**, *15*, 532−552.

(17) Gallicchio, E.; Zhang, L. Y.; Levy, R. M. *J. Comput. Chem.* **2001**, *23*, 517−529.

(18) Ghosh, A.; Rapp, C. S.; Friesner, R. A. *J. Phys. Chem. B* **1998**, *102*, 10983−10990.

(19) Jacobson, M. P.; Friesner, R. A.; Xiang, Z.; Honig, B. *J. Mol. Biol.* **2002**, *320*, 597−608.

(20) Jacobson, M. P.; Kaminski, G. A.; Friesner, R. A.; Rapp, C. S. *J. Phys. Chem. B* **2002**, *106*, 11673−11680.

(21) Jacobson, M. P.; Pincus, D. L.; Rapp, C. S.; Day, T. J. F.; Honig, B.; Shaw, D. E.; Friesner, R. A. *Proteins* **2004**, *55*, 351−367.

(22) Zhu, K.; Pincus, D. L.; Zhao, S.; Friesner, R. A. *Proteins* **2006**, *65*, 438−452.

(23) Kalyanaraman, C.; Bernacki, K.; Jacobson, M. P. *Biochemistry* **2005**, *44*, 2059−2071.

(24) Li, X.; Jacobson, M. P.; Friesner, R. A. *Proteins* **2004**, *55*, 368−382.

(25) Jorgensen, W. L.; Maxwell, D. S.; Tirado-Rives, J. *J. Am. Chem. Soc.* **1996**, *118*, 11225−11236.

(26) Kaminski, G. A.; Friesner, R. A.; Tirado-Rives, J. *J. Phys. Chem. B* **2001**, *105*, 6474−6487.

(27) Banks, J. L.; Beard, H. S.; Cao, Y.; Cho, A. E.; Damm, W.; Farid, R.; Felts, A. K.; Halgren, T. A.; Mainz, D. T.; Maple, J. R.; Murphy, R.; Philipp, D. M.; Repasky, M. P.; Zhang, L. Y.; Berne, B. J.; Friesner, R. A.; Gallicchio, E.; Levy, R. M. *J. Comput. Chem.* **2005**, *26*, 1752−1780.

(28) Sherman, W.; Day, T. J. F.; Jacobson, M. P.; Friesner, R. A.; Farid, R. *J. Med. Chem.* **2006**, *49*, 534−553.

(29) Huang, N.; Kalyanaraman, C.; Irwin, J. J.; Jacobson, M. P. *J. Chem. Inf. Model.* **2006**, *46*, 243−253.

(30) Bernacki, K.; Kalyanaraman, C.; Jacobson, M. P. *J. Biomol. Screen.* **2005**, *10*, 675−681.

(31) *Glide*; Schrodinger Inc: New York, 2004.

(32) Friesner, R. A.; Banks, J. L.; Murphy, R. B.; Halgren, T. A.; Klicic, J. J.; Mainz, D. T.; Repasky, M. P.; Knoll, E. H.; Shelley, M.; Perry, J. K.; Shaw, D. E.; Francis, P.; Shenkin, P. S. *J. Med. Chem.* **2004**, *47*, 1739−1749.

(33) Goto, S.; Okuno, Y.; Hattori, M.; Nishioka, T.; Kanehisa, M. *Nucleic Acids Res.* **2002**, *30*, 402−404.

CT600129F