SCALABILITY OF THE PARALLEL STRONGIN ALGORITHM IN THE PROBLEM OF OPTIMIZING A MOLECULAR-DYNAMIC FORCE FIELD

K. S. Shefov [a], M. M. Stepanova [b]

St. Petersburg State University, 7-9 University emb, St. Petersburg, 199034, Russia

E-mail: [a] k.s.shefov@gmail.com, [b] mstep@mms.nw.ru

Strongin's multifactorial global search algorithm (MGSA) allows one to find an absolute minimum of a function of multiple variables on a mesh. In this contribution a parallel program is presented that implements the algorithm above applied to ReaxFF MD force field parameters search. In case of ReaxFF optimization, computation time of an objective function value significantly exceeds time of data exchange between parallel processes. One is able to speed up computation by obtaining not only one but several function values in various points simultaneously. Our software implements two levels of parallelism. To deal with function of multiple variables, one uses a scan for mapping a multi-dimensional domain of definition of a function into a one-dimensional segment. To decrease the effect of losing information of multi-dimensional points proximity, N scans are used. Function values in N different mesh points are computed in parallel. This is the first level of parallelism. To define a mesh point of a next iteration, MGSA finds a subinterval with the most probable location of the minimum and computes an objective function value in a certain point of this subinterval. Function values are also calculated in parallel in $(M-1)$ subintervals with less probability. This is the second level of parallelism. Thus the two levels allow one to compute $M \cdot N$ function values in parallel each iteration. In this contribution we research scalability of our MGSA implementation, namely, the dependence of the number of algorithm iterations and time it needs to converge on the number of CPU cores used, separately for each level of parallelism.

Keywords: numerical simulation, chemically reactive systems, reactive force field, molecular dynamics, parameter optimization, parallel algorithm, scalability, absolute extremum search

1. Introduction

Strongin's multifactorial global search algorithm (MGSA) allows one to obtain an absolute minimum of a function of multiple variables on a mesh. This paper investigates scalability of our parallel MGSA implementation, developed to search for parameters of ReaxFF molecular-dynamic force field. The feature of the implementation is two-level parallelism. The aim of the work is to justify the choice of the optimal parameters of the algorithm for a specific calculation on the available hardware.

2. The Problem

Classical molecular dynamics (MD) method is to numerically solve Newton's equations for every atom in a simulated system: $\frac{d\vec{g}}{dt} = -\nabla U(\vec{r})$, where $\vec{g}$ are impulses of atoms in the system, t is time, $\vec{r}$ are coordinates of atoms in the system, and $\nabla U(\vec{r})$ is a gradient of a force field. The force field depends on a set of parameters: $U = f(p_1, p_2, \dots, p_D)$. These parameters are defined before the solution of Newton's equations and do not alter during the simulation. The procedure of searching the force field parameters for a particular simulated system is referred to as optimization. The measure of optimality is the objective function (OF) that depend on the parameters: $T(p_1, p_2, \dots, p_D)$. The OF is determined by the deviation of any characteristics of the simulated system obtained methods of MD, from those obtained by other, more accurate methods. The process of optimization is to search the force field parameters that bring the OF the minimal value: $p_1^*, p_2^*, \dots, p_D^*: T(p_1^*, p_2^*, \dots, p_D^*) = \min_W T$, where W is the domain of parameters search having dimension of D.

3. ReaxFF Force Field

As the MD force field $U(\vec{r})$ we use ReaxFF (Reactive Force Field). It is the sum of several terms, each responsible for their own type of interaction, and has the general formula [1]:

$$E_{\text{ReaxFF}}\left(\{r_{ij}\},\{r_{ijk}\},\{r_{ijkl}\},\{q_i\},\{BO_{ij}\}\right) = E_{\text{bond}} + E_{\text{lp}} + E_{\text{over}} + E_{\text{under}} + E_{\text{val}} + E_{\text{pen}} + \\ + E_{\text{coa}} + E_{\text{tors}} + E_{\text{conj}} + E_{\text{hbond}} + E_{\text{vdWaals}} + E_{\text{Coulomb}}. \quad (1)$$

The force field depend on the distance between atoms in each atom pair, on angles formed by each three atoms, on dihedral angles formed by each four atoms, on atoms' effective charges and bond orders (BO) between atoms. Atoms' charges as well as bond orders have non-integer values and themselves depend on mutual location of the atoms of the system.

4. Objective Function

As the measure of optimality we use the objective function (2).

$$T = \sum_{k=1}^{L} \sigma_k \left| U_{\text{k}}^{\text{QC}} - U_{\text{k}}^{\text{ReaxFF}} \right| + \sum_{k=1}^{L} \sigma_{L+k} \sqrt{\sum_{\alpha=1}^{A_k} \sum_{i=1}^{3} \left(F_{\text{k}\alpha\text{i}}^{\text{QC}} - F_{\text{k}\alpha\text{i}}^{\text{ReaxFF}} \right)^2} \quad (2)$$

Here U_k are potential energies of models of the training-set (simple chemical compounds used as basis for optimization), $F_{k\alpha i}$ are components of forces acting on every atom of the models, L is the number of models in the training-set, A_k is the number of atoms in the model k, and σ_k are the weight factors. Indices QC and ReaxFF mean that the corresponding characteristics are obtained with quantum chemistry methods and MD methods, respectively. $U_{\text{k}}^{\text{ReaxFF}}$ and $F_{\text{k}\alpha\text{i}}^{\text{ReaxFF}}$ depend on the parameters $p_1, p_2, \dots, p_D$ of ReaxFF force field. In our multifactorial algorithm in addition to the main objective function (2) we take into account several more functions-restrictions on the individual groups of terms in (2). This allows one to set all weights σ_k equal to unity and not to solve the task of their optimal choice.

5. Strongin Global Search Algorithm

To optimize the objective function we use Strongin global search algorithm [2] (GSA). It allows one to find an absolute minimum of a function on a segment and is bases on probability approach. Initially GSA is formulated for functions of one variable defined on a segment. The OF $T(x)$ must satisfy the generalized Lipschitz condition on the entire search domain: $|T(x_1) - T(x_2)| \leq K\rho(x_1, x_2)$, where x_1 и x_2 are any numbers from the search interval, K is a constant, and ρ is a metric in the space L_1, $\rho(x_1, x_2) = f(|x_1 - x_2|)$, and the function f has its own inverse f^{-1}. Initial step of GSA is to compute values of the OF at the boundaries of the search domain and, if necessary, in M different arbitrary points inside it.

The general scheme of a single iteration is below.

1. Sort the points of k previous iterations in ascending order of their coordinates:
$a = x_0 < \cdots < x_i < \cdots < x_k = b$.
2. Compute for every interval (x_{i-1}, x_i), $1 \leq i \leq k$ the characteristic $R(i)$ that defines probability of finding the minimum on the given interval.
3. Sort $R(i)$ in descending order: $R(i_1) > \cdots > R(i_k)$.
4. For all the intervals having numbers $i_1, \ldots, i_M, M < k$, compute OF in points $x^{k+j} = \xi(i_j) \in (x_{i_j-1}, x_{i_j}), j = \overline{1, M}$. Computations are done in parallel. Here $\xi(i_j)$ is the position of the mathematical expectation of a minimum.
5. Check the stop condition: $|x_{i_1} - x_{i_1-1}| < \varepsilon$.

The presented scheme allows one to execute GSA in parallel using M processes. In the multivariate version of the algorithm a multivariable function is reduced to a function of one variable by means of scans of the Peano curve type. From here one can obtain another level of parallelism, applying the method rotating scans [3]. Each parallel process works with its own scan rotated relative to the main one by the angles $\pm\frac{\pi}{2}$ in some pair of dimensions. In total one is able to do $D \cdot (D-1)$ of such rotations for D-dimensional domain of definition of a function. Thus, the maximal number of processes at this level of parallelism is $N_{\max} = D \cdot (D-1) + 1$. Each of them executed GSA and at every iteration communicates its result to all the rest of the processes. Working with several scans simultaneously compensates consequences of information loss about points proximity when a multidimensional domain is mapped to a segment.

In the current work we use the non-injective Peano type curve [2] when every point of a search domain of dimension D may have up to 2^D prototypes on a segment. In this variant GSA the minimum is being searched on a mesh. The number of mesh points on a segment is $2^{D(m+1)} - 2^{Dm}$. Here m is the scan parameter that defines its complexity and the number of points of multidimensional mesh for a single dimension $(2^m + 1)$. More about the MGSA in the problem of ReaxFF optimization see in [4].

6. Implementation

The program is written in C++ with use of MPI and has two levels of parallelism: the first one is parallel work with N scans when mapping the search domain to a segment; the second one is parallel calculation of the OF value for M greatest probability characteristics $R(i)$ (see GSA general scheme). Thus, the program operates with a rectangular matrix of processes of size $N \cdot M$.

Let us note that in case of ReaxFF optimization the computation time of the OF value significantly exceeds both the data exchange time and the time of all the rest procedures inside a single MGSA iteration.

On the first level of parallelism (Figure 1) each block corresponds to a single scan. All N blocks are equivalent and perform the same sequence of routines, exchanging data between each other three times every iteration.

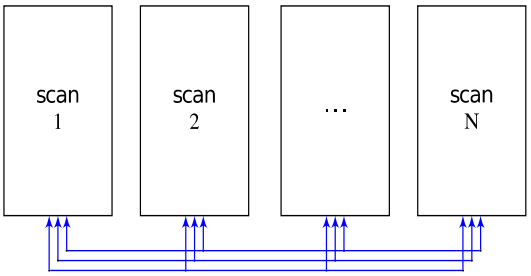

Figure 2. MGSA flowchart. 1st level of

On the second level of parallelism (Figure 2) we have one "scan-master" and $M-1$ "scan-slave" processes. All M 2nd-level processes compute the OF value, the "scan-master" in addition participates in the 1st-level data exchange and does MGSA.

The blue blocks in Figure 2 mean MPI-exchange between 1st-level parallel processes (correspond to blue arrows in Figure 1), the red ones mean exchange between 2nd-level processes inside a 1st-level block.

Notations: Y_i — search domain points (type double[D]); Z_i — objective function values (type double); V_i — MGSA indices values (type int); X_i — mesh points in segment $[0,1]$, to which the serach domain is mapped (type double).

The triplets $\{X_i, V_i, Z_i\}$ are stored in a special structure which simultaneously is a linked list for all X_i, common red-black tree for X_i, and separate red-black trees for X_i with the same V_i. This structure significantly reduces the time of a single iteration of the algorithm.

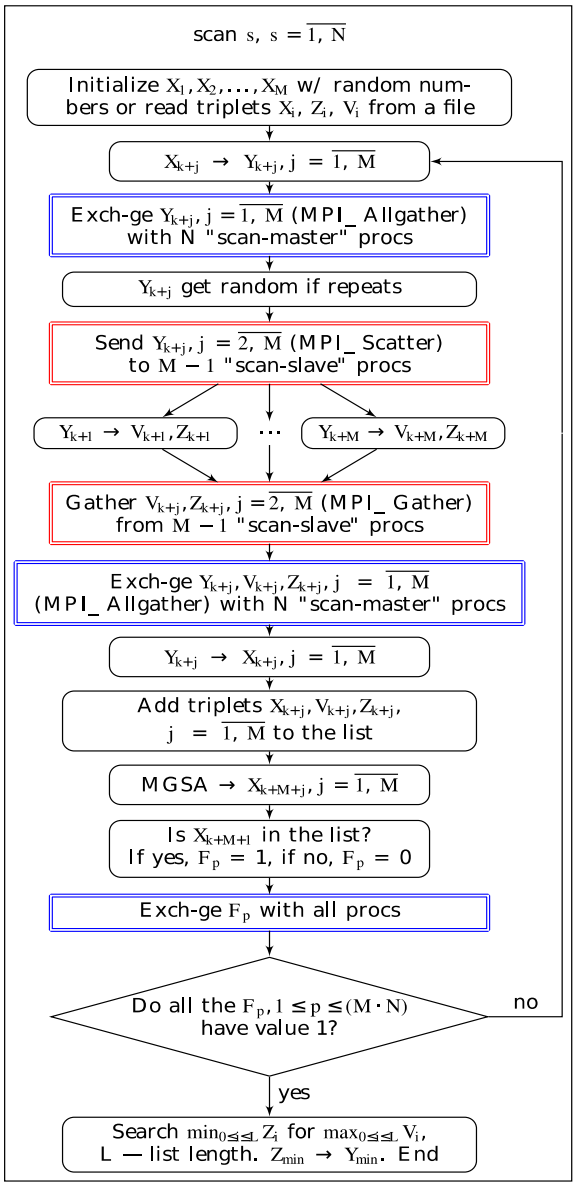

Figure 2. MGSA flowchart. 2nd level of parallelism

7. Scalability

Hardware is provided by Resource Center Computer Center of St. Petersburg State University.
The cluster: OS: CentOS Linux 7 (Core); MPICH2 v1.4.1p1.
14 nodes: 2 × 4 core CPU Intel® Xeon® E5335 2.00 GHz, 16 GB RAM;
2 nodes: 2 × 4 core CPU Intel® Xeon® E5-2603 v2 1.80 GHz, 16 GB RAM.

7.1. Average iteration time depending on the number of cores

The first level of parallelism has limited maximal number of processes. The cases $D = 11$ and $D = 4$ with averaging over 100 iterations were investigated. In Figure 3 the plots of average time of a single MGSA iteration dependencies on the number of used cores $P = N \cdot M$ are shown for 7 different cases. N is the number of cores on the 1st level of parallelism, and M — on the 2nd. The number of OF values calculated in a single iteration is equal to P. So although the time of a single iteration increases with P, the

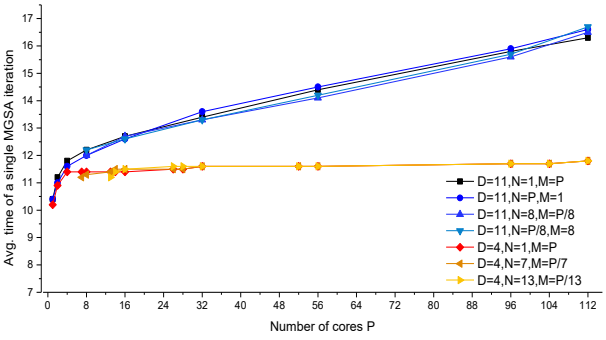

Figure 3. Avg. time of an iteration of core number

computation of an OF value accelerates. An OF value is computed for several dozens of molecules with the help of LAMMPS [5] MD simulation library. It takes from 10 to 13 seconds. Each process consumes less than 100 MB of RAM.

The chart shows two stages of growth. In the first stage, when $1 \le P \le 4$ (within 1 CPU), the increase of the duration of a single iteration is due to an increase in the calculation time of the value of the OF. This is a feature of the hardware and LAMMPS library. In the second stage, when $P > 4$, the increase of duration is caused by overhead costs for data exchange between processes and interaction with the file system. Let us note that the duration increase does not depend on the distribution of cores by the levels of parallelism. For $D < 11$ the charts' slopes are smaller, since overheads are declining with D.

7.2. Defining optimal dimension of the search domain

Let us plot the dependencies of MGSA acceleration by the number of its iterations needed for convergence on the number of parallel processes M of the 2nd level and the number of scans $N = 1$ for $1 \le D \le 5$. The acceleration $A_I(M) = \frac{I(8)}{I(M)}$, where $I(M)$ is the number of MGSA iterations needed for convergence. The number of points of the segment is chosen so that for different D it is of the same order — 10^6, and the algorithm converged in less than an hour when all the 128 cores are used. In this approach, a pure algorithm is studied without taking into account exchange and other overhead costs.

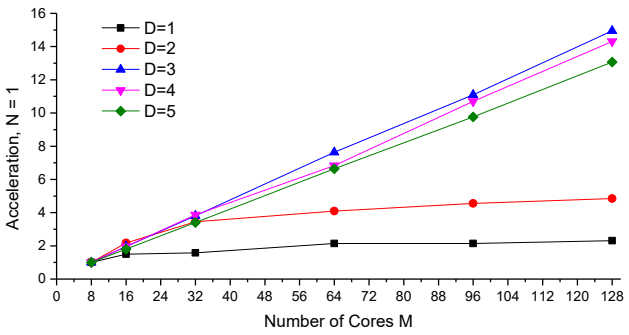

Figure 4. Scalability of MGSA for D = 2, 3, 4, 5

In Figure 4 it is seen that the dependencies for $3 \le D \le 5$ are close to linear. At the same time, when $1 \le D \le 2$, the acceleration grows much slower, since complexity of the problem is not enough to effectively use all the cores we have. When the dimension of the search domain is 3 and higher, the algorithm is well scalable for 128 cores. The optimal dimension of the search domain is $D = 4$, since for larger D and the number of mesh nodes of the order of 10^6 the mesh on multidimensional domain will be coarse.

7.3. Scalability of the 2nd-level parallelism when D = 4

For the case of $D = 4$ and three fixed numbers of scans $N = 1, 7, 13$ let us compare the dependency of MGSA acceleration by the number of iterations A_I and time A_t on the number of cores M on the 2nd level (Figure 5). $A_I(N,M) = \frac{I(1,1)}{I(N,M)}$, $A_T(N,M) = \frac{t(1,1)}{t(N,M)}$, where $I(N,M)$ and $t(N,M)$ are, respectively, the number of MGSA iterations and its operation time until convergence. In Figure 5 the blue lines indicate the dependencies by iterations ($A_I(P)$), the red lines — by time ($A_T(P)$). Linear approximation using least square method tells us of the constant scalability efficiency ($\Delta A_T/\Delta P$) no matter how many cores are used. By the difference between the plots of A_t and A_t one can see the decrease in efficiency due to overhead costs. Using of a single scan ($N = 1$) is the most efficient.

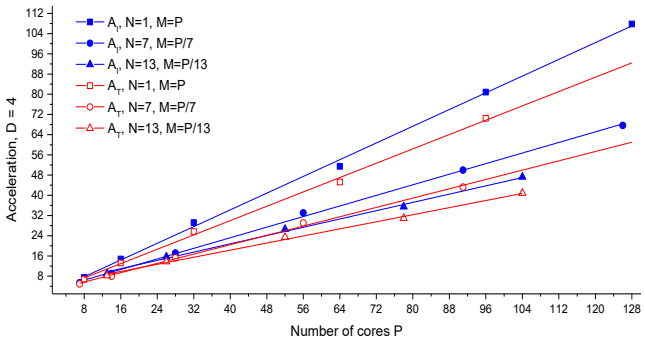

Figure 5. Scalability of MGSA for D=4; N=1,7,13

Table 1. Scalability efficiency

N	1	7	13
$\Delta A_T/\Delta P$, %	71±1.9	46±1.3	35±1.0

7.4. Scalability dependence on the objective function

Analytical formulas of individual terms of ReaxFF (2) differ a lot, therefore when optimizing groups of parameters from different terms, we will have the OF of a very different type. In Figure 6 one can see the dependencies of MGSA acceleration by iterations and by time on the number of cores for $D = 4$, $N = 1$ when optimizing 4 parameters form the terms E_{Coulomb} and E_{bond}, and 4 — from the term E_{vdWaals}. The red lines indicate the acceleration by convergence time, the blue ones — by iterations number. Scalability efficiencies for the first and the second group of parameters, obtained by LSA, are equal to, respectively, (71.6 ± 1.6) % and (70.2 ± 2.0) %. The values match with accuracy to errors, hence the effectiveness of this realization is independent of the OF type.

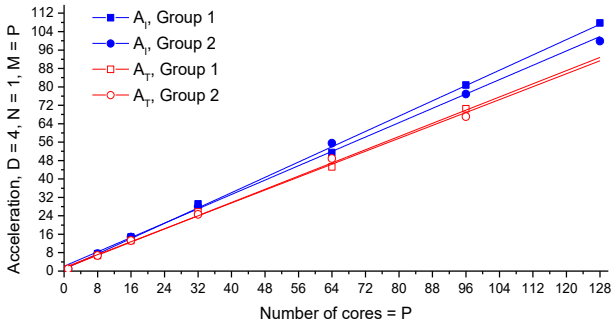

Figure 6. MGSA scalability for diff. param. groups

8. Conclusion

It is shown that the implementation of the multifactorial global search algorithm (MGSA) with meshes more than 10^6 nodes and executed on a cluster of 128 cores is scalable with constant efficiency up to 71 % when the number of optimized parameters $D > 2$.

For the given hardware it is optimal to operate with groups of 4 ReaxFF force field parameters, since MGSA converges in less than an hour.

It is more efficient to distribute resources to the 2nd level of parallelism with only one scan.

The introduction of the 2nd level of parallelism (by probability characteristics of MGSA) and the increase in the number of cores to 128 reduces the calculation time by 30 – 50 times (compared to the previously used single-level MGSA implementation [6]).

References

[1] K. Nomura, R. K. Kalia, A. Nakano, P. Vashishta, J. L. Landa. A scalable parallel algorithm for large-scale reactive force-field molecular dynamics simulation // Comp. Phys. Comm. — 2008. — Vol. 178(2). — P. 73 – 87.

[2] Strongin R.G. Chislennye metody v mnogoekstremalnykh zadachakh [Numerical Methods in Multi-Extremal Problems]. — M.: «Nauka». 1978. — 240 p. (in Russian)

[3] Strongin R.G., Gergel V.P., Barkalov K.A. Parallelnye metody resheniya zadach globalnoy optimizatsii [Parallel methods of global optimization problems solution] // Izvestiya Vuzov. Priborostroenie. — 2009. — Vol. 52. — P. 25 – 33 (in Russian).

[4] Stepanova M.M., Shefov K.S., Slavyanov S.Yu. Multifactorial global search algorithm in the problem of optimizing a reactive force field // Theoretical and Mathematical Physics. — 2016. — Vol. 187, Issue 1. — P. 603 – 617.

[5] LAMMPS package: https://lammps.sandia.gov/ (accessed 30.09.2018)

[6] Shefov K.S., Stepanova M.M., Kotriakhova S.N. Parametrization of the Reactive MD Force Field for Zn-O-H Systems // CEUR Workshop Proceedings. — 2016. — Vol. 1787. — P. 567 – 572.