Structural Determination of \((\text{Al}_2\text{O}_3)_n\) \((n = 1–15)\) Clusters Based on Graphic Processing Unit

Qiyao Zhang and Longjiu Cheng*

Department of Chemistry, Anhui University, Hefei, Anhui 230039, People’s Republic of China

**ABSTRACT:** Global optimization algorithms have been widely used in the field of chemistry to search the global minimum structures of molecular and atomic clusters, which is a nondeterministic polynomial problem with the increasing sizes of clusters. Considering that the computational ability of a graphic processing unit (GPU) is much better than that of a central processing unit (CPU), we developed a GPU-based genetic algorithm for structural prediction of clusters and achieved a high acceleration ratio compared to a CPU. On the one-dimensional (1D) operation of a GPU, taking \((\text{Al}_2\text{O}_3)_a\) clusters as test cases, the peak acceleration ratio in the GPU is about 220 times that in a CPU in single precision and the value is 103 for double precision in calculation of the analytical interatomic potential. The peak acceleration ratio is about 240 and 107 on the block operation, and it is about 77 and 35 on the 2D operation compared to a CPU in single precision and double precision, respectively. And the peak acceleration ratio of the whole genetic algorithm program is about 35 compared to CPU at double precision. Structures of \((\text{Al}_2\text{O}_3)_a\) at \(n = 1–10\) reported in previous works are successfully located, and their low-lying structures at \(n = 11–15\) are predicted.

1. INTRODUCTION

In the fields of chemistry, due to the large number of local minimum structures of molecular and atomic clusters, it is usually difficult to locate the lowest energy conformation. The global optimization algorithm\(^1–14\) has been widely used to solve this problem. In the past years, various methods have been developed for global structural prediction, such as the genetic algorithm (GA),\(^15,16\) simulated annealing algorithm,\(^17,18\) particle swarm optimization,\(^19,20\) basin-hopping and its variants,\(^21,22\) fast annealing evolutionary algorithm,\(^23,24\) random tunneling algorithm,\(^25\) potential deformation,\(^26–28\) simple linkage,\(^29,30\) modeling methods,\(^31–34\) and so on. However, as structural prediction is a nondeterministic polynomial (NP) problem, the computation is very time-consuming at a large size in all global optimizations.

Graphic processing unit (GPU) computing may be more suitable for structural optimization at a large size. Over the past years, there has been a remarkable increasing in the performance and capabilities of GPUs.\(^35\) The modern GPU is not only a powerful graphics engine, but also a parallel computation processor in scientific computing, in which the peak arithmetic and memory bandwidth is much better than that of the central processing unit (CPU). This means that it has practical significance on adding cores rather than increasing single-thread performance. These features promote a new area which is called general-purpose computing on GPU (GPGPU) in high-performance calculations. Nowadays, GPU computing has been widely used in the fields of chemistry, such as for molecular dynamics simulations,\(^36,37\) chemical reaction simulation processes,\(^38\) quantum chemistry,\(^39,40\) and so on. Many molecular dynamics software tools have already achieved impressive speedup using GPUs, such as AMBER,\(^41–43\) OpenMM,\(^44\) and so on. However, so far there is no report on the structural global optimization of atomic and molecular clusters based on GPUs, and we believe that GPU computing would have a good performance on structural determination.

GPGPU may have a good performance in algorithms of global optimization, such as GA, which is the most common global optimization method for structural prediction of atomic/molecular clusters. In this paper, using \((\text{Al}_2\text{O}_3)_a\) clusters as test clusters, we developed a GPU-based GA method for structural prediction and achieved high acceleration ratio compared to the CPU-based GA method.

2. THEORY AND COMPUTATIONAL METHOD

A. Potential Energy Function. The potential energy function of \((\text{Al}_2\text{O}_3)_a\) clusters is based on the rigid-ion model (RM) in which we consider each aluminum ion as a point charge of +3e and each oxygen ion as a point charge of −2e. The analytical interatomic potential (IP) between aluminum and oxygen ions\(^45\) is calculated using eq 1.

\[
V(r_{ij}) = \frac{q_i q_j}{r_{ij}} + \frac{A}{r_{ij}^{12}} + B \exp\left(-\frac{r_{ij}}{\rho}\right) - \frac{C}{r_{ij}^{6}}
\]

(1)

The IP has four different energy terms. The first is the coulomb contribution, where \(q_i\) and \(q_j\) represent the charge of ions \(i\) and \(j\), and \(r_{ij}\) represents the distance between two ions. The following three terms are the Lennard-Jones terms and Born–Mayer terms, respectively. The values of parameter \(A\) is 1.0 eV Å\(^{12}\) for like-charged species and 10.0 eV Å\(^{12}\) for others. The values of Buckingham interatomic pair potential parameters \(B\), \(C\), and \(\rho\) are shown in Table 1.\(^33,46\)

Received: February 6, 2015
Published: April 30, 2015

DOI: 10.1021/acs.jcim.5b00069
J. Chem. Inf. Model. 2015, 55, 1012–1020
We know that the IP needs to be calculated between each pair of two atoms. It means we should calculate the IP for \( A_{12}^2 \) times and count the half sum of them, as shown in eq 2, to completely describe an \( (\text{Al}_2\text{O}_3)_n \) cluster.

\[
E = \sum_{n=1}^{N} \sum_{j>i} V(r_{ij})
\]

\[
= \sum_{n=1}^{N} \sum_{j>i} \left[ \frac{q_i q_j}{r_{ij}} + \frac{A}{r_{ij}^6} + B \exp\left(\frac{-r_{ij}}{\rho}\right) - \frac{C}{r_{ij}^8} \right]
\]

\[\text{(2)}\]

**B. Gradient Optimization Based on GPU.** The limited-memory Broyden–Fletcher–Goldfarb–Shanno (LBFGS) algorithm is a gradient optimization algorithm\(^{47,48}\) in the family of quasi-Newton methods. It is particularly suitable for optimization problems with a large number of variables, such as searching local maximum or minimum energy of cluster. The optimization problems with a large number of variables, such as quasi-Newton methods. It is particularly suitable for solving many complex computational problems in a more memory-efficient way than on a CPU.\(^{49}\) In the CUDA programming model, we can efficiently leverage the parallel computing platform and programming model that NVIDIA introduced CUDA (Compute Unified Device Architecture), which is a general purpose parallel computing platform and programming model that leverages the parallel computing engine in NVIDIA GPUs to solve many complex computational problems in a more memory-efficient way than on a CPU.\(^{49}\) In the CUDA programming model, we can efficiently leverage the parallel computing platform and programming model.
model, the smallest executable unit is a thread. A number of threads can compose a block and a number of blocks can compose a grid. Each thread has its own local memory, and threads per block have a same shared memory. There is a limitation for the number of threads per block, and for GPUs after 2010, a block may contain up to 1024 threads.

CUDA is an extension of the C language, called CUDA C, where the function is executed for \( N \) times in parallel by \( N \) different CUDA threads. The GPU is treated as device, and the CPU, as host. The kernels execute on a GPU, and the rest of the C codes execute on a CPU. In this work, we try to design a GPU-based LBFGS using CUDA C. The calculation of IP and gradient are using the GPU memory, and the rest of the codes are using the CPU memory. The calculation of IP and gradient in LBFGS is regarded as the kernel, and the rest can be executed on a CPU, as a tagging in Figure 1. There are three patterns in our GPU-based LBFGS, named as one-dimensional (1D), block, and two-dimensional (2D) operations. Figure 2 gives the sketch map of the three operations of the kernel.

In the test cases for \( (\text{Al}_2\text{O}_3)_n \) clusters, there are \( N \) atoms.

First, we design a traditional one-dimensional (1D) operation. It is a linear operation. We use one block with \( N \) threads, and each thread computes the IP and gradient of one atom with all other atoms. It should be noted that, there is a thread number limitation in one block. When \( N \) is greater than the limitation (1024 in this work), multiple blocks should be used.

Figure 2. Sketch map of the GPU-based LBFGS.
There is a shortcoming of traditional 1D operation. At a large cluster size, there is too much computation in one thread, and at the same time the thread number is too limited. To solve this problem, Walker et al. developed a block operation. The $N \times N$ matrix of IP and gradient is divided into $m^2$ blocks ($m = N/32$), and the width of the block is 32 (32 threads can reach the idealized maximum acceleration ratio). Only on-diagonal and half off-diagonal titles need to be calculated, and the real block number is $\lfloor m(m + 1)/2 \rfloor$. There are 32 threads in one block, and one thread calculates only 32 pairs of IP and gradient. The total number of threads is $N(m + 1)/2$, which is $(m + 1)/2$ times of that in 1D operation.

At a small cluster size, the degree of parallelism in both 1D and block operations is not sufficient due to the limited number of threads. Thus, we design a 2D operation for the calculation of small size. In 2D operation, there are $N$ blocks, and each block contains $N$ threads. The total number of threads is $N^2$, and the $N \times N$ matrix of IP and gradient is calculated at the same time in parallel.

It should be noted that, the concepts of 1D, block, and 2D mean the patterns in computing the 2D matrix of IP and gradient. The 2D operation here is different to the operation with a 2D thread index. In 1D, block, and 2D operations, all threads use the same formula but different parameters. Different tasks are assigned for different threads to calculate the IP and gradient between atoms in parallel by their thread identifications.

C. Computational Process. In our test-case, we choose GA for structural determination of $(\text{Al}_2\text{O}_3)_x$ clusters. GA is one of the most common methods for global optimization. It belongs to the larger class of evolutionary algorithms (EA), which generate solutions to optimization problems using techniques inspired by natural evolution, such as inheritance, mutation, selection, and crossover. The flowchart of GA is shown in Figure 3, which has five steps.

1. Define the parameters of GA such as cluster size, population size, genetic algebra, proportion of genetic manipulation, and so on, and then, stochastically produce a population of $(\text{Al}_2\text{O}_3)_x$ clusters according to these parameters and each individual has a different structure.

2. Optimize each cluster in the population by the GPU-based LBFGS.

3. Evaluate the population. If the iteration number reaches the preset maximum genetic algebra, exit. Otherwise, else go to the next step.

4. According to the parameters of GA, use genetic manipulation for some clusters, which include crossover, mutation, and select.

5. Replace the worst cluster by a newly generated one in the population, perform GPU-based LBFGS to optimize the new cluster, and then go to step 3.

In CPU-version GA, LBFGS occupies most of the calculation time. When the cluster sizes are 1, 2, 4, 8, 16, 32, 64, and 128, LBFGS occupies 76.17%, 94.94%, 99.06%, 99.81%, 99.95%, 99.99%, 99.99%, and 99.99% of the total time, respectively. This means that using GPU-based LBFGS is the key role to improve the acceleration ratio of GA.

D. Development Platform. Our development platform is the CentOS release 5.5 (Final) which is one of the Linux release version. The compilers are gcc-4.1.2 for C codes and nvcc-5.0 for the kernels. The CPU is Intel Xeon E5620, and the GPU is NVIDIA Tesla C2050, where the compute capability of the Tesla C2050 is 2.1.

3. RESULTS AND DISCUSSION

A. Acceleration Ratio. In ideal conditions, the GPU computing in parallel consumes only one time step. However, considering the data transmission between devices and hosts, it needs more time steps in actual calculation, and the acceleration ratio cannot reach the ideal number.

GPU occupancy number is an important factor for acceleration ratio. According to the NVIDIA CUDA C programming guide, Tesla C2050 consists of 14 stream multiprocessors (SMs). Each SM has 48 warps and each warp has 32 threads. Thus, as shown in Figure 4, in order to reach the idealized maximum acceleration ratio, at least 32 threads is needed in one block and the thread number should exceed 21504 ($14 \times 48 \times 32$). Thus, in block operation the minimum width of a block is 32 for the ideal acceleration ratio.

First, we only compute the IP and gradient in GPU and CPU codes to compare the acceleration ratio. The computational
capability of the GPU in this work is 2.1, and the speed of single precision code is about twice that of the double precision code.

Considered both single and double precisions, 1D, block, and 2D operation, there are six versions about GPU codes (1D-single, 1D-double, block-single, block-double, 2D-single, and 2D-double). GPU codes cannot deal with too large a system because of the limitation of GPU memory. The IP and gradient of cluster can be calculated when the cluster size is about hundreds of thousands in 1D and block operation and hundreds in 2D operations. But the accelerate ratio is not increasing while the cluster size is up to several thousands. There is little difference in running time between the single and the double precision computation in CPU, so we only choose the CPU computation with double precision. All of the codes above are running with O3 optimization and each program calculates the IP and gradient 1000 times to reach a reliable running time.

In Figure 5, we found GPU computing is faster than CPU computing in case of the cluster size $n$ is up to 3 or 4. In 1D operation, the peak acceleration ratio for GPU computing is about 103 in double precision, and the rate is about 220 in single precision. On the block operation, the peak acceleration ratio of GPU computing is about 107/240 in double/single precision. On the 2D operation, the peak acceleration ratio of GPU computing is about 35/77 in double/single precision. Due to the limitation GPU memory, 2D operation cannot be used in computing the cluster at too large sizes. The performance of block operation is better than 1D operation at medium cluster sizes, but they are the same at large cluster sizes. However, the performance of 2D operation is much better than that of 1D and block operations for small clusters because of high GPU occupancy.

Table 2. Accuracy of GPU and CPU Codes in Single- and Double-Precisions$^a$

<table>
<thead>
<tr>
<th>cluster sizes</th>
<th>average IP: single-precision</th>
<th>AARD (%)</th>
<th>average IP: double-precision</th>
<th>AARD (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>GPU</td>
<td>CPU</td>
<td>AARD (%)</td>
<td>GPU</td>
</tr>
<tr>
<td>1</td>
<td>-33.5898</td>
<td>-33.5899</td>
<td>0.00</td>
<td>-33.5899</td>
</tr>
<tr>
<td>2</td>
<td>-17.0117</td>
<td>-17.0123</td>
<td>0.04</td>
<td>-17.0123</td>
</tr>
<tr>
<td>4</td>
<td>-79.5255</td>
<td>-79.5271</td>
<td>0.02</td>
<td>-79.5267</td>
</tr>
<tr>
<td>8</td>
<td>28.8061</td>
<td>28.7926</td>
<td>-0.47</td>
<td>28.8035</td>
</tr>
<tr>
<td>16</td>
<td>494.993</td>
<td>495.050</td>
<td>0.11</td>
<td>494.985</td>
</tr>
<tr>
<td>32</td>
<td>3104.12</td>
<td>3103.11</td>
<td>-0.33</td>
<td>3104.11</td>
</tr>
<tr>
<td>64</td>
<td>6498.62</td>
<td>6497.68</td>
<td>-0.14</td>
<td>6498.54</td>
</tr>
<tr>
<td>102</td>
<td>44620.7</td>
<td>44610.8</td>
<td>-0.22</td>
<td>44617.2</td>
</tr>
<tr>
<td>128</td>
<td>23031.1</td>
<td>23025.2</td>
<td>-0.25</td>
<td>23031.1</td>
</tr>
<tr>
<td>203</td>
<td>83885.8</td>
<td>83614.3</td>
<td>-3.25</td>
<td>83885.8</td>
</tr>
</tbody>
</table>

$^a$AARD means average absolute relative deviation.
The global structural determination is an NP problem, and only the systems with a small cluster size can be dealt with. The speed of single precision computation is about two times as that of double precision computation. However, as shown in Table 2, the single precision computation has low performance in accuracy. Thus, in searching the structure of \((\text{Al}_2\text{O}_3)_n\) cluster \((n = 1−15)\), we choose double precision GPU and 2D operation.

The structures of \((\text{Al}_2\text{O}_3)_n\) clusters are determined by the GPU-based GA and the running time is also recorded for CPU serial computing and GPU parallel computing. As shown in Figure 6, the peak acceleration ratio for the whole GA-GPU computing is about 35 in 2D operation double precision, which is close to the acceleration ratio in IP-GPU computing. GA-GPU computing has a positive acceleration ratio at the cluster size \(n = 3\), and after that point, the acceleration ratio grows rapidly until reaches the peak value \(\sim 35\).

### B. Geometry Structures

Using the GA-LBFGS method based on GPU, we located the putative lowest-energy structures of \((\text{Al}_2\text{O}_3)_n\) \((n = 1−15)\) clusters.

\((\text{Al}_2\text{O}_3)_n\) \((n = 1−5)\): As shown in Figure 7, the global minimum (1A) is a state with \(D_{3h}\) symmetry for \(\text{Al}_2\text{O}_3\), the linear isomer (1B) is 0.04 eV higher in energy, and the kite-shaped (1C) is 0.60 eV higher in energy with \(C_2\) symmetry in our calculation. For \((\text{Al}_2\text{O}_3)_3\), the global minimum (2A) is a cage with \(T_d\) symmetry. The 2B isomers with \(C_{2v}\) symmetry is 0.20 eV higher in energy, 2C isomers with \(C_{2v}\) symmetry is 0.74 eV higher in energy, and 2D isomers with \(C_{2h}\) symmetry is 2.69 eV higher in energy. For \((\text{Al}_2\text{O}_3)_3\) 3A is a tea-cozy structure with \(C_1\) symmetry, and the first three isomers are nearly degenerate in energy. For \((\text{Al}_2\text{O}_3)_4\), the superimposed isomer (4A) with \(D_{2d}\) symmetry that was assumed to be the global minimum structure, it has two surfaces and each surface consists of three deformed-hexagons. For \((\text{Al}_2\text{O}_3)_5\), 5A is in the lowest energy with \(C_{2v}\) symmetry.

\((\text{Al}_2\text{O}_3)_n\) \((n = 6−10)\): For large clusters, as shown in Figure 7, the structures are more complex. 6A is the lowest energy isomer for \((\text{Al}_2\text{O}_3)_6\) with \(C_1\) symmetry. For \((\text{Al}_2\text{O}_3)_7\), the triple-tower-layer isomer (7A) with \(C_1\) symmetry is the global minimum structure. For \((\text{Al}_2\text{O}_3)_8\), the global minimum (8A) is a cage with \(D_{2d}\) symmetry, it is consisted of four surface and each surface is a deformed-hexagon as 4A. For \((\text{Al}_2\text{O}_3)_9\), 9A is the most stable isomer with \(C_1\) symmetry, and for \((\text{Al}_2\text{O}_3)_{10}\)
10A with C₁ symmetry has the lowest energy with respect to other isomers.

Putative structures of \((\text{Al}_2\text{O}_3)_n\) \((n = 10)\) clusters have been reported in the literature.\(^{41,50,51}\) We find that all the reported structures have been successfully located by our GA-GPU computing, indicating reliability of our method. The energy orders of some clusters may be different due to the differences for the method in calculating the energy. \((\text{Al}_2\text{O}_3)_n\) \((n = 11-15)\). We also predict the structures for \((\text{Al}_2\text{O}_3)_n\) \((n = 11-15)\) clusters which have not been reported, using GPU computing. As shown in Figure 8, for \((\text{Al}_2\text{O}_3)_11\), the global minimum (11A) is a layered tower with C₅ symmetry which is similar to the 7A structure. For \((\text{Al}_2\text{O}_3)_12\), 12A with C₁ symmetry is the most stable isomer, in which some atoms can overlap in the specific perspective. For \((\text{Al}_2\text{O}_3)_13\), 13A is a bottle-shaped structure with C₅ symmetry. For \((\text{Al}_2\text{O}_3)_14\), 14A is a heart-shaped structure with D₂ symmetry. For \((\text{Al}_2\text{O}_3)_15\), 15A is the global minimum structure with C₁ symmetry which is a bit like the composition of 9A.

**C. Discussion.** In this work, both 1D, block, and 2D operations in GPU computing are investigated, where 1D and block operation is linear and 2D operation is planar. For large cluster sizes, peak acceleration ratio can achieved in 1D and block operations when the occupancy rate of GPU reaches 100%, while at a small cluster size, the performance of 2D operation is much better than that of 1D as the former one has a higher occupancy rate of GPU. In our calculations, the \((\text{Al}_2\text{O}_3)_n\) cluster is not so large, so 2D operation has better performance.

In our GA-GPU codes, only the IP codes are replaced in GPU. To achieve a higher acceleration ratio, the whole LBFGS codes should be put into GPU. Furthermore, there are too many complex logical operations in LBFGS, and it is unreasonable to do so. A gradient-based local minimization method suitable for GPU computing is expected for higher performance.

## 4. CONCLUSIONS

In summary, we treat the \((\text{Al}_2\text{O}_3)_n\) clusters as the RM and compute the IP of them to express the cluster energy using the GPU-based GA-LBFGS to get the global minimum structures of \((\text{Al}_2\text{O}_3)_n\) clusters. For \((\text{Al}_2\text{O}_3)_n\) clusters, all the optimization structures are in good accordance with the structures found in the previous work when \(n\) is from 1 to 10 and every optimization structure has high symmetry when \(n\) is from 11 to 15. In calculation of the analytical interatomic potential and gradient, the peak acceleration ratio in a GPU is about 220 times compared to a CPU in single precision, and the value is 103 for double precision. The peak acceleration ratio is about 240 and 107 on the block operation, and it is about 77 and 35 on the two-dimensional operation compared to a CPU in single and double precision, respectively. This work can not only be applied in \((\text{Al}_2\text{O}_3)_n\) clusters but also to other cluster systems, and we continue this work to find a more efficient GPU computing solution.

### AUTHOR INFORMATION

#### Corresponding Author
*E-mail: clj@ustc.edu.*

#### Notes
The authors declare no competing financial interest.

#### ACKNOWLEDGMENTS

This work is financed by the National Natural Science Foundation of China (21273008) and by the 211 Project of Anhui University. The calculations were carried out at the High-Performance Computing Center of Anhui University.

#### REFERENCES