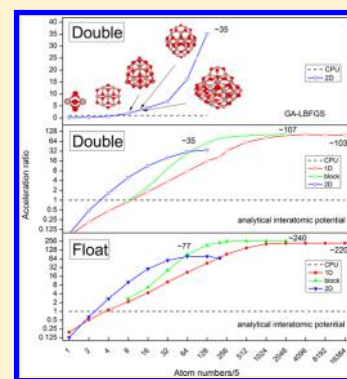


Structural Determination of $(\text{Al}_2\text{O}_3)_n$ ($n = 1-15$) Clusters Based on Graphic Processing Unit

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 $(\text{Al}_2\text{O}_3)_n$ 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 $(\text{Al}_2\text{O}_3)_n$ clusters 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 algorithm¹⁻¹⁴ 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,^{1,21,22} fast annealing evolutionary algorithm,^{23,24} random tunneling algorithm,²⁵⁻²⁷ potential deformation,^{28,29} simple linkage,^{30,31} modeling methods,³²⁻³⁴ 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 process simulations,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 $(\text{Al}_2\text{O}_3)_n$ 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 $(\text{Al}_2\text{O}_3)_n$ 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 ions⁴⁵ is calculated using eq 1.

$$V(r_{ij}) = \frac{q_i q_j}{r_{ij}} + \frac{A}{r_{ij}^{12}} + B \exp\left(\frac{-r_{ij}}{\rho}\right) - \frac{C}{r_{ij}^6} \quad (1)$$

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·Å¹² for like-charged species and 10.0 eV·Å¹² for others. The values of Buckingham interatomic pair potential parameters B , C , and ρ are shown in Table 1.^{33,46}

Received: February 6, 2015

Published: April 30, 2015

Table 1. Buckingham Interatomic Pair Potential Parameters for Al_2O_3

pair potential parameters	A (eV \AA^{12})	ρ (\AA)	B (eV)	C (eV \AA^6)
$\text{Al}^{3+}\text{-Al}^{3+}$	1.0		0.0	0.00
$\text{Al}^{3+}\text{-O}^{2-}$	10.0	0.2649	2409.505	0.00
$\text{O}^{2-}\text{-O}^{2-}$	1.0	0.1490	22764.000	27.88

We know that the IP needs to be calculated between each pair of two atoms. It means we should calculate the IP for A_{5n}^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.

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

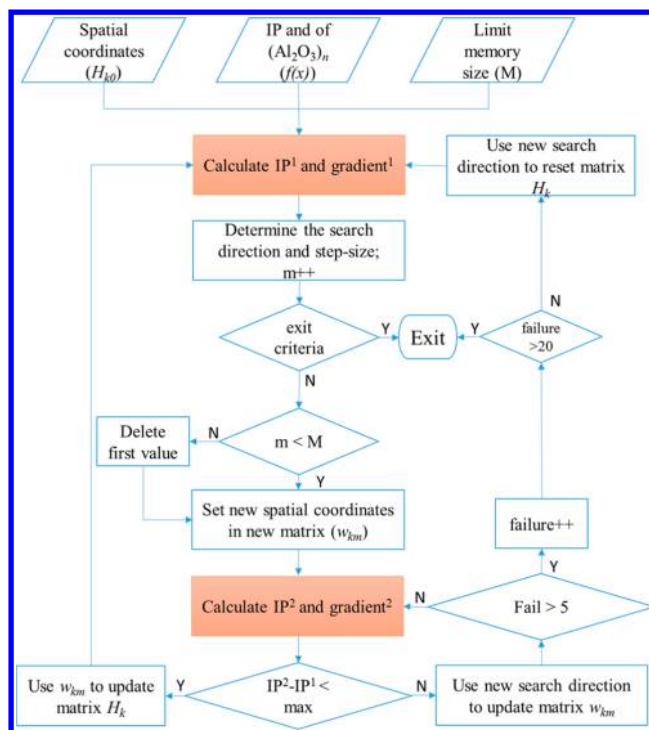
$$= \sum_{n=1}^{i=1} \sum_{n}^{j>i} \left[\frac{q_i q_j}{r_{ij}} + \frac{A}{r_{ij}^{12}} + B \exp\left(\frac{-r_{ij}}{\rho}\right) - \frac{C}{r_{ij}^6} \right] \quad (2)$$

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 local minima of $(\text{Al}_2\text{O}_3)_n$ clusters can be obtained using this method. Similar as original BFGS, LBFGS uses an approximation to the inverse Hessian matrix to steer its search through variable space, but it only maintains a history of the past m updates of the position x and gradient $\nabla f(x)$, where generally the history size m could be small. LBFGS is a universal algorithm of gradient optimization; however, we should define some given data that includes the original matrix (H_{k0}) optimization function ($f(x)$) and its gradient ($\nabla f(x)$). In our test case of $(\text{Al}_2\text{O}_3)_n$ clusters, the original matrix is the spatial coordinates of $(\text{Al}_2\text{O}_3)_n$ clusters, the optimization function is the IP and the gradient is the change ratio of IP, as shown in eq 3.

$$G(r_{ij}) = -\frac{q_i q_j}{r_{ij}^3} - \frac{12A}{r_{ij}^{14}} - \frac{B}{\rho r_{ij}} \exp\left(\frac{-r_{ij}}{\rho}\right) + \frac{C}{r_{ij}^8} \quad (3)$$

Figure 1 plots the flowchart of LBFGS. LBFGS starts with a given structure of cluster, equation of its IP, and the limit memory size M :

- (1) Calculate the IP and gradient of the original given $(\text{Al}_2\text{O}_3)_n$ cluster.
- (2) According to the values of gradient, determine the search direction and step-size along the gradient descent direction. If the exit criteria is met, terminate the program. Otherwise, if the working space w_{km} reaches the limitation, delete the first value of w_{km} and then go to step 3; else, go to step 3 directly. The exit criteria is that the root-mean-square of gradient is close to zero (10^{-4}) or the IP and gradient have been computed more than 200 000 times.
- (3) Adjust the structure of the $(\text{Al}_2\text{O}_3)_n$ cluster and store the new spatial coordinates in w_{km} according to the search direction and step size. The working space w_{km} stores not only the adjusted structure, but also the information on gradient, search direction, and step size.
- (4) Calculate the IP and gradient of $(\text{Al}_2\text{O}_3)_n$ cluster which is given by w_{km} .

**Figure 1.** Flowchart of LBFGS. Highlighted parts can be executed by GPU.

- (5) If the value of IP of step 4 is lower than that of step 1, update original coordinates using w_{km} and then go to step 1. Otherwise, adjust the w_{km} slightly and recalculate the IP and gradient within 5 attempts until the step 1 condition is satisfied. If this fails after 5 attempts, increase the flag of failure by one, change the search direction, reset the original coordinates using w_{km} , and go to step 1. If the flag of failure reaches 20, terminate the program.

Obviously, with the increasing cluster size, the amount of calculation for IP and gradient increase more rapidly. By counting the running time of each part in CPU-version LBFGS, it is proved that the calculation of IP and gradient occupies the most part of time. When the cluster sizes are 1, 2, 4, 8, 16, 32, 64, and 128, these parts occupy 35.48%, 63.64%, 88.83%, 96.80%, 99.12%, 99.49%, 99.72%, and 99.91% of the total CPU time, respectively. The occupied proportion in calculating the IP and gradient is nearly 100% while the cluster sizes is up to 128. It means that reducing the time of this part plays the key role to accelerate all the calculation.

GPU is designed for a particular class of applications with the following characteristics:³⁵ (a) Computational complexities are large. (b) Parallelism is substantial. (c) Throughput is more important than latency. Obviously, both the IP and gradient need massive calculation, they occupy the most time in LBFGS. When computing the IP and gradient, the same formulas (eqs 1–3) are used in addition to the parameters. This meets the demands of GPU parallel computing, and we try to calculate the IP and gradient with a GPU using CUDA.

In November 2006, NVIDIA introduced CUDA (Compute Unified Device Architecture), which is a general purpose parallel computing platform and programming model that leverages the parallel compute engine in NVIDIA GPUs to solve many complex computational problems in a more efficient way than on a CPU.⁴⁹ In the CUDA programming

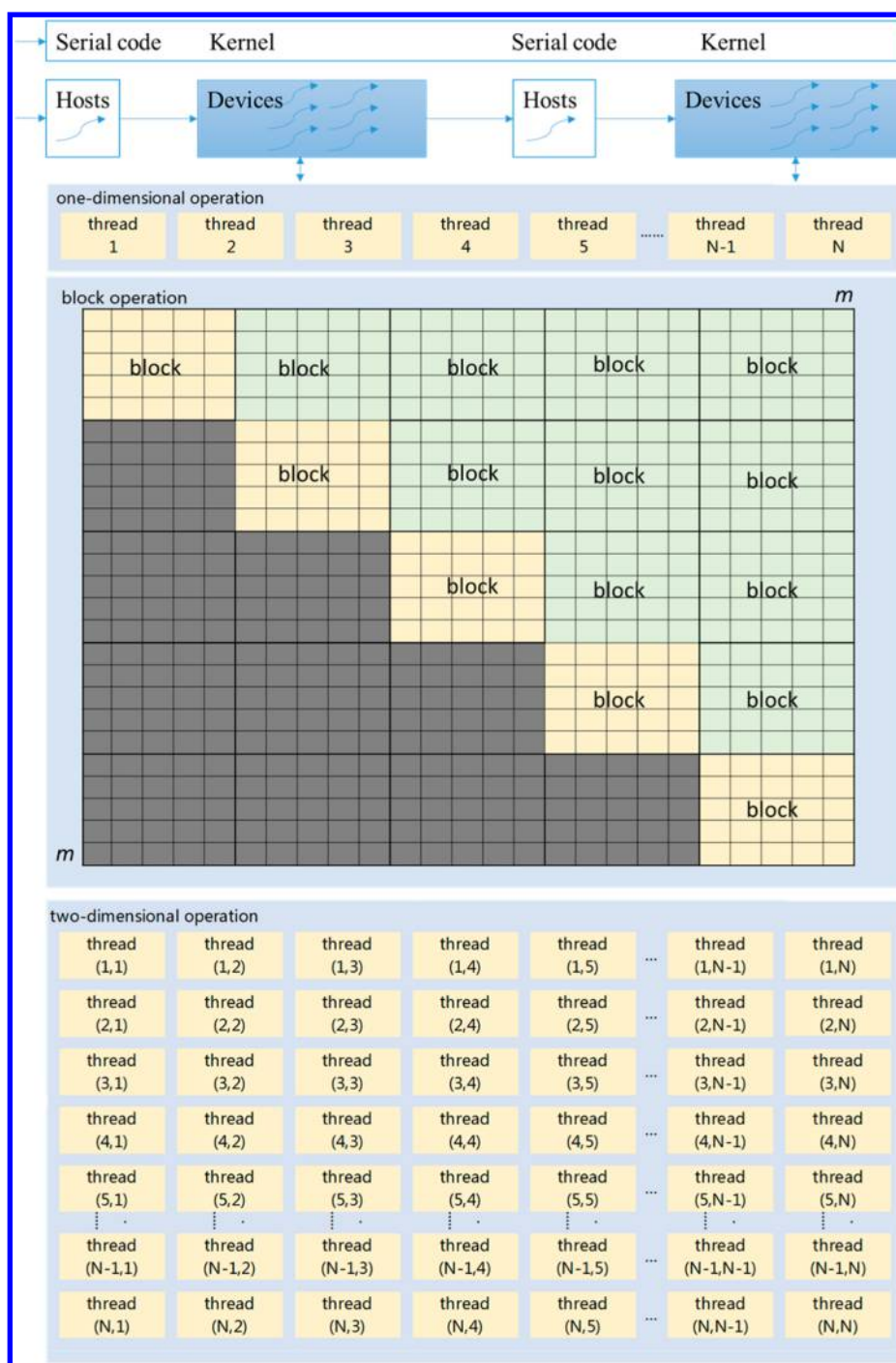


Figure 2. Sketch map of the GPU-based LBFGS.

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.

There is a shortcoming of traditional 1D operation. At a large cluster size, there is too much computations in one thread, and at the same time the thread number is too limited. To solve this problem, Walker et al.⁴¹ 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 $[m(m+1)]/2$. 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 are 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 $(\text{Al}_2\text{O}_3)_n$ clusters. GA^{15,16} 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 $(\text{Al}_2\text{O}_3)_n$ clusters according to these parameters and each individual has a different structure.

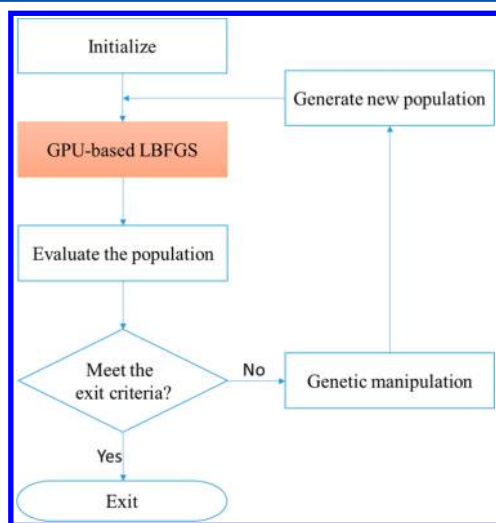


Figure 3. Flowchart of genetic algorithms. Highlighted parts can be executed by GPU.

- (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 capability⁴⁰ 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,⁴⁹ 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

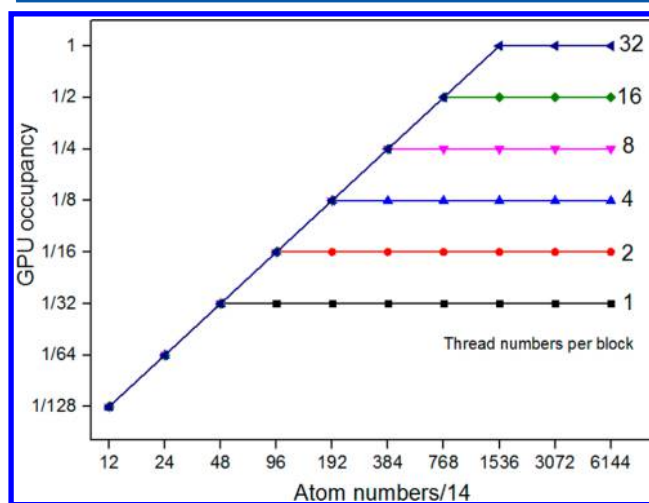


Figure 4. GPU occupancy per block with the increasing sizes of atom numbers.

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

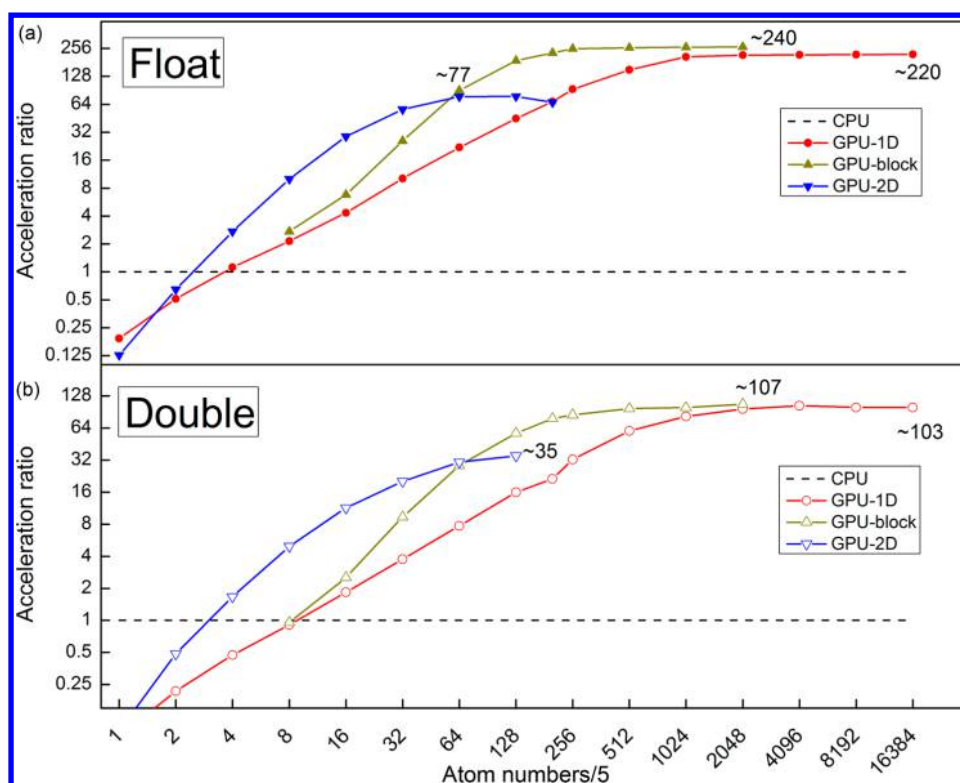


Figure 5. Acceleration ratio of 1D, block, and 2D GPU operations for computing the IP and gradient at float (top) and double (bottom) precisions as a function of the atom numbers. The CPU codes in double precision are taken as references (dashed line).

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

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

^aAARD means average absolute relative deviation.

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.

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

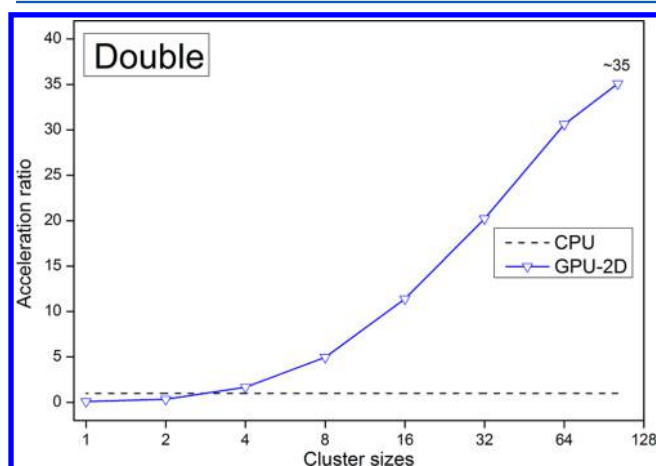


Figure 6. Acceleration ratio of GA-LBFGS in double precision as a function of cluster sizes.

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 ~ 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 Al_2O_3 , the liner isomer (1B) is 0.04 eV higher in energy, and the kite-shaped (1C) is 0.60 eV higher in energy with C_{2v} symmetry in our calculation. For $(\text{Al}_2\text{O}_3)_2$, 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_{3d} 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_s 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 $(\text{Al}_2\text{O}_3)_{10}$,

$(\text{Al}_2\text{O}_3)_n$	1	2	3	4	5	6	7	8	9	10
A										
	$D_{3h}, 0.00$	$T_d, 0.00$	$C_1, 0.00$	$D_{3d}, 0.00$	$C_{2v}, 0.00$	$C_1, 0.00$	$C_s, 0.00$	$D_{2d}, 0.00$	$C_1, 0.00$	$C_1, 0.00$
B										
	$D_{3h}, 0.04$	$C_{2v}, 0.20$	$C_{2v}, 0.02$	$C_{2v}, 0.29$	$C_1, 0.05$	$C_1, 0.09$	$C_1, 0.35$	$C_s, 0.35$	$C_s, 0.02$	$C_1, 0.13$
C										
	$C_{2v}, 0.60$	$C_{2v}, 0.74$	$C_{2v}, 0.03$	$C_{4h}, 0.77$	$C_1, 0.10$	$C_1, 0.28$	$C_s, 0.41$	$C_s, 0.83$	$C_1, 0.11$	$C_1, 0.14$
D										
		$D_{2h}, 2.69$	$D_{3h}, 0.37$	$C_1, 0.98$	$C_2, 0.29$	$C_1, 0.38$	$C_1, 0.61$	$C_1, 0.84$	$C_1, 0.18$	$C_1, 0.28$
E										
			$C_s, 2.06$	$C_1, 1.15$	$C_2, 0.34$	$C_s, 0.39$	$C_1, 0.66$	$C_1, 0.85$	$C_1, 0.18$	

Figure 7. Structures, symmetry point groups, and relative energies (eV) of the global minimum and lower-lying structures of $(\text{Al}_2\text{O}_3)_n$ ($n = 1-10$) that were calculated by GA-LBFGS: Al gray, O red.






















$(\text{Al}_2\text{O}_3)_n$	11	12	13	14	15
A	 C_s , 0.00	 C_1 , 0.00	 C_s , 0.00	 D_2 , 0.00	 C_1 , 0.00
B	 C_s , 0.15	 C_1 , 0.08	 C_1 , 0.01	 C_1 , 1.59	 C_1 , 0.01
C	 C_s , 0.98	 C_1 , 0.20	 C_1 , 0.43	 C_1 , 1.88	 C_s , 0.06
D	 C_1 , 1.00	 C_s , 0.30	 C_1 , 0.44	 C_2 , 1.90	 C_1 , 0.10
E	 C_1 , 1.00				

Figure 8. Structures, symmetry point groups, and relative energies (eV) of the global minimum and lower-lying structures of $(\text{Al}_2\text{O}_3)_n$ ($n = 11-15$) that calculated by GA-LBFGS. For Al_2O_3 , the bond lengths in angstrom are given: Al gray, O red.

10A with C_1 symmetry has the lowest energy with respect to other isomers.

Putative structures of $(\text{Al}_2\text{O}_3)_n$ ($n = 1-10$) 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 C_s symmetry which is similar to the 7A structure. For $(\text{Al}_2\text{O}_3)_{12}$, 12A with C_1 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 C_s symmetry. For $(\text{Al}_2\text{O}_3)_{14}$, 14A is a heart-shaped structure with D_2 symmetry. For $(\text{Al}_2\text{O}_3)_{15}$, 15A is the global minimum structure with C_1 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 be 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

- (1) Wales, D. J.; Doye, J. P. Global Optimization by Basin-Hopping and the Lowest Energy Structures of Lennard-Jones Clusters Containing Up to 110 Atoms. *J. Phys. Chem. A* **1997**, *101*, 5111–5116.

- (2) Wales, D. J.; Scheraga, H. A. Global Optimization of Clusters, Crystals, and Biomolecules. *Science* **1999**, *285*, 1368–1372.
- (3) Cheng, L.; Feng, Y.; Yang, J.; Yang, J. Funnel Hopping: Searching the Cluster Potential Energy Surface over the Funnels. *J. Chem. Phys.* **2009**, *130*, 214112.
- (4) Cheng, L.; Cai, W.; Shao, X. A Connectivity Table for Cluster Similarity Checking in the Evolutionary Optimization Method. *Chem. Phys. Lett.* **2004**, *389*, 309–314.
- (5) Rapallo, A.; Rossi, G.; Ferrando, R.; Fortunelli, A.; Curley, B. C.; Lloyd, L. D.; Tarbuck, G. M.; Johnston, R. L. Global Optimization of Bimetallic Cluster Structures. I. Size-Mismatched Ag–Cu, Ag–Ni, and Au–Cu Systems. *J. Chem. Phys.* **2005**, *122*, 194308.
- (6) Rossi, G.; Ferrando, R.; Rapallo, A.; Fortunelli, A.; Curley, B. C.; Lloyd, L. D.; Johnston, R. L. Global Optimization of Bimetallic Cluster Structures. II. Size-matched Ag–Pd, Ag–Au, and Pd–Pt Systems. *J. Chem. Phys.* **2005**, *122*, 194309.
- (7) Floudas, C. A.; Gounaris, C. E. A Review of Recent Advances in Global Optimization. *J. Glob. Optim.* **2009**, *45*, 3–38.
- (8) Guillén-Gosálbez, G.; Grossmann, I. A Global Optimization Strategy for the Environmentally Conscious Design of Chemical Supply Chains under Uncertainty in the Damage Assessment Model. *Comput. Chem. Eng.* **2010**, *34*, 42–58.
- (9) Luo, X.; Yang, J.; Liu, H.; Wu, X.; Wang, Y.; Ma, Y.; Wei, S.; Gong, X.; Xiang, H. Predicting Two-Dimensional Boron–Carbon Compounds by the Global Optimization Method. *J. Am. Chem. Soc.* **2011**, *133*, 16285–16290.
- (10) Chen, H.; Zhang, Y.; Gong, X.; Xiang, H. Predicting New TiO₂ Phases with Low Band Gaps by a Multiobjective Global Optimization Approach. *J. Phys. Chem. C* **2014**, *118*, 2333–2337.
- (11) Liu, Z.-P.; Hu, P. CO Oxidation and NO Reduction on Metal Surfaces: Density Functional Theory Investigations. *Top. Catal.* **2004**, *28*, 71–78.
- (12) Fu, H.; Liu, Z.-P.; Li, Z.-H.; Wang, W.-N.; Fan, K.-N. Periodic Density Functional Theory Study of Propane Oxidative Dehydrogenation over V₂O₅ (001) Surface. *J. Am. Chem. Soc.* **2006**, *128*, 11114–11123.
- (13) Zhao, J.; Buldum, A.; Han, J.; Lu, J. P. First-principles Study of Li-Intercalated Carbon Nanotube Ropes. *Phys. Rev. Lett.* **2000**, *85*, 1706.
- (14) Wang, J.; Wang, G.; Zhao, J. Density-functional Study of Au_n (*n* = 2–20) Clusters: Lowest-energy Structures and Electronic Properties. *Phys. Rev. B* **2002**, *66*, 035418.
- (15) Silva, M. X.; Galvão, B. R.; Belchior, J. C. Theoretical Study of Small Sodium–potassium Alloy Clusters Through Genetic Algorithm and Quantum Chemical Calculations. *Phys. Chem. Chem. Phys.* **2014**, *16*, 8895–8904.
- (16) Alexandrova, A. N. H₂O_n Clusters: Microsolvation of the Hydrogen Atom via Molecular ab Initio Gradient Embedded Genetic Algorithm (GEGA). *J. Phys. Chem. C* **2010**, *114*, 12591–12599.
- (17) Sankararao, B.; Yoo, C. K. Development of a Robust Multiobjective Simulated Annealing Algorithm for Solving Multi-objective Optimization Problems. *Ind. Eng. Chem. Res.* **2011**, *50*, 6728–6742.
- (18) Topper, R. Q.; Feldmann, W. V.; Markus, I. M.; Bergin, D.; Sweeney, P. R. Simulated Annealing and Density Functional Theory Calculations of Structural and Energetic Properties of the Ammonium Chloride Clusters (NH₄Cl)_n, (NH₄⁺)(NH₄Cl)_n, and (Cl[−])(NH₄Cl)_n, *n* = 1–13. *J. Phys. Chem. A* **2011**, *115*, 10423–10432.
- (19) Wang, Y.; Li, B.; Weise, T.; Wang, J.; Yuan, B.; Tian, Q. Self-Adaptive Learning Based Particle Swarm Optimization. *Inform. Sci.* **2011**, *181*, 4515–4538.
- (20) Kennedy, J. Particle Swarm Optimization. In *Encyclopedia of Machine Learning*; Springer: New York, 2010; pp 760–766.
- (21) Do, H.; Besley, N. A. Structural Optimization of Molecular Clusters with Density Functional Theory Combined with Basin Hopping. *J. Chem. Phys.* **2012**, *137*, 134106.
- (22) Liu, Y.; Wen, H.; Huang, T.; Lin, X.-X.; Gai, Y.-B.; Hu, C.-J.; Zhang, W.-J.; Huang, W. Structural Exploration of Water, Nitrate/Water, and Oxalate/Water Clusters with Basin-Hopping Method Using a Compressed Sampling Technique. *J. Phys. Chem. A* **2014**, *118*, 508–516.
- (23) Cai, W.; Shao, X. A Fast Annealing Evolutionary Algorithm for Global Optimization. *J. Comput. Chem.* **2002**, *23*, 427–435.
- (24) Xinchao, Z. Simulated Annealing Algorithm with Adaptive Neighborhood. *Appl. Soft. Comput.* **2011**, *11*, 1827–1836.
- (25) Jiang, H.; Cai, W.; Shao, X. A Random Tunneling Algorithm for the Structural Optimization Problem. *Phys. Chem. Chem. Phys.* **2002**, *4*, 4782–4788.
- (26) Shao, X.; Jiang, H.; Cai, W. Parallel Random Tunneling Algorithm for Structural Optimization of Lennard-Jones Clusters up to *n* = 330. *J. Chem. Inf. Comput. Sci.* **2004**, *44*, 193–199.
- (27) Gornov, A. Y.; Zarodnyuk, T. S. Tunneling Algorithm for Solving Nonconvex Optimal Control Problems. In *Optimization, Simulation, and Control*; Springer: New York, 2013; pp 289–299.
- (28) Piel, L.; Kostrowicki, J.; Scheraga, H. A. On the Multiple-Minima Problem in the Conformational Analysis of Molecules: Deformation of the Potential Energy Hypersurface by the Diffusion Equation Method. *J. Phys. Chem.* **1989**, *93*, 3339–3346.
- (29) Kaasbjerg, K.; Thygesen, K. S.; Jauho, A.-P. Acoustic Phonon Limited Mobility in Two-Dimensional Semiconductors: Deformation Potential and Piezoelectric Scattering in Monolayer MoS₂ from First Principles. *Phys. Rev. B* **2013**, *87*, 235312.
- (30) Locatelli, M.; Schoen, F. Simple Linkage: Analysis of A Threshold-Accepting Global Optimization Method. *J. Glob. Optim.* **1996**, *9*, 95–111.
- (31) Lorieux, M. MapDisto: Fast and Efficient Computation of Genetic Linkage Maps. *Mol. Breed.* **2012**, *30*, 1231–1235.
- (32) Northby, J. Structure and Binding of Lennard-Jones Clusters: 13 ≤ *N* ≤ 147. *J. Chem. Phys.* **1987**, *87*, 6166–6177.
- (33) Romero, D.; Barrón, C.; Gómez, S. The Optimal Geometry of Lennard-Jones Clusters: 148–309. *Comput. Phys. Commun.* **1999**, *123*, 87–96.
- (34) Li, G.; Shi, J.; Qu, X. Modeling Methods for GenCo Bidding Strategy Optimization in the Liberalized Electricity Spot Market—A State-of-the-Art Review. *Energy* **2011**, *36*, 4686–4700.
- (35) Owens, J. D.; Houston, M.; Luebke, D.; Green, S.; Stone, J. E.; Phillips, J. C. GPU Computing. *P. IEEE* **2008**, *96*, 879–899.
- (36) Anderson, J. A.; Lorenz, C. D.; Travesset, A. General Purpose Molecular Dynamics Simulations Fully Implemented on Graphics Processing Units. *J. Comput. Phys.* **2008**, *227*, 5342–5359.
- (37) Friedrichs, M. S.; Eastman, P.; Vaidyanathan, V.; Houston, M.; Legrand, S.; Beberg, A. L.; Ensign, D. L.; Bruns, C. M.; Pande, V. S. Accelerating Molecular Dynamic Simulation on Graphics Processing Units. *J. Comput. Chem.* **2009**, *30*, 864–872.
- (38) Li, H.; Petzold, L. Efficient Parallelization of the Stochastic Simulation Algorithm for Chemically Reacting Systems on the Graphics Processing Unit. *Int. J. High Perform. Comput. Appl.* **2010**, *24*, 107–116.
- (39) Götz, A. W.; Wölfe, T.; Walker, R. C. Quantum Chemistry on Graphics Processing Units. *Annu. Rep. Comput. Chem.* **2010**, *6*, 21–35.
- (40) Yasuda, K. Accelerating Density Functional Calculations with Graphics Processing Unit. *J. Chem. Theory Comput.* **2008**, *4*, 1230–1236.
- (41) Götz, A. W.; Williamson, M. J.; Xu, D.; Poole, D.; Le Grand, S.; Walker, R. C. Routine Microsecond Molecular Dynamics Simulations with AMBER on GPUs. 1. Generalized born. *J. Chem. Theory Comput.* **2012**, *8*, 1542–1555.
- (42) Salomon-Ferrer, R.; Götz, A. W.; Poole, D.; Le Grand, S.; Walker, R. C. Routine Microsecond Molecular Dynamics Simulations with AMBER on GPUs. 2. Explicit solvent particle mesh Ewald. *J. Chem. Theory Comput.* **2013**, *9*, 3878–3888.
- (43) Salomon-Ferrer, R.; Case, D. A.; Walker, R. C. An Overview of the Amber Biomolecular Simulation Package. *WIREs Comput. Mol. Sci.* **2013**, *3*, 198–210.
- (44) Eastman, P.; Friedrichs, M. S.; Chodera, J. D.; Radmer, R. J.; Bruns, C. M.; Ku, J. P.; Beauchamp, K. A.; Lane, T. J.; Wang, L. P.; Shukla, D. OpenMM 4: A Reusable, Extensible, Hardware

Independent Library for High Performance Molecular Simulation. *J. Chem. Theory Comput.* **2012**, *9*, 461–469.

(45) Woodley, S. M. Atomistic and Electronic Structure of $(X_2O_3)_n$ Nanoclusters; $n = 1-5$, $X = B, Al, Ga, In$ and Tl . *P. R. Soc. A-Math. Phys.* **2011**, *467*, 2020–2042.

(46) Richard, A.; Catlow, C. Self-Consistent Interatomic Potentials for the Simulation of Binary and Ternary Oxides. *J. Mater. Chem.* **1994**, *4*, 831–837.

(47) Liu, D. C.; Nocedal, J. On the Limited Memory BFGS Method for Large Scale Optimization. *Math. Program.* **1989**, *45*, 503–528.

(48) Byrd, R. H.; Lu, P.; Nocedal, J.; Zhu, C. A Limited Memory Algorithm for Bound Constrained Optimization. *SIAM J. Sci. Comput.* **1995**, *16*, 1190–1208.

(49) Nvidia, C. *Nvidia Cuda C Programming Guide 5.5*; NVIDIA Corporation: Santa Clara, 2013; pp 169–197.

(50) Walsh, A.; Woodley, S. M. Evolutionary Structure Prediction and Electronic Properties of Indium Oxide Nanoclusters. *Phys. Chem. Chem. Phys.* **2010**, *12*, 8446–8453.

(51) Rahane, A. B.; Deshpande, M. D.; Kumar, V. Structural and Electronic Properties of $(Al_2O_3)_n$ Clusters with $n = 1-10$ from First Principles Calculations. *J. Phys. Chem. C* **2011**, *115*, 18111–18121.