A GPU based Conformational Entropy Calculation Method

Qian Zhang[1], José M. García[2], Junmei Wang[3], Tingjun Hou[1*] and Horacio Pérez-Sánchez[4*]

[1]*Institute of Funcional Nano & Soft Materials (FUNSOM) and Jiangsu Key Laboratory for Carbon-Based Functional Materials & Devices, Soochow University, Suzhou, Jiangsu 215123, China*

s20080512@gmail.com, tingjunhou@hotmail.com

[2]*Computer Engineering Department, School of Computer Science, 30100 University of Murcia, Spain*

jmgarcia@ditec.um.es, horacio@ditec.um.es

[3]*Department of Biochemistry, The University of Texas Southwestern Medical Center, 5323 Harry Hines Blvd., Dallas, TX 73590, USA*

junmwang@yahoo.com

[4]*Computer Science Department, Catholic University of Murcia (UCAM) 30107 Murcia, Spain*

horacio.ucam@gmail.com *(corresponding author)

Abstract. Conformational entropy calculation, usually computed by normal mode analysis (NMA), is a time-consuming step in MM-PB/GBSA calculations. Here, instead of NMA, a solvent accessible surface area (SAS) based model was employed to compute the conformational entropy. A new fast GPU-based method called MURCIA (Molecular Unburied Rapid Calculation of Individual Areas) was used instead of the traditional Shrake-Rupley algorithm to accelerate the calculation of SASA for each atom. MURCIA employs two different kernels to determine each atom's neighbors. First one (K1) uses just brute force for the calculation of atomic neighbours, while the second one (K2) uses an advanced algorithm involving hardware interpolations via GPU texture memory unit for such purpose. These two kernels have their own advantages depending on the protein size. The algorithm is extensively evaluated in three protein datasets, and achieves good results for all of them, being this method around two orders of magnitude faster than the former algorithm at the size of 10^5. Finally, further improvements of the algorithm are discussed.

1 Introduction

Free energy calculation is one of the central interests in computational chemistry and computational biology. Many approaches have been developed for the prediction of binding free energies, such as thermodynamic integration (TI), free energy perturbation (FEP), MM/PBSA, MM/GBSA, etc.[1-5]. The main idea of TI and FEP is calculating the ΔG between two similar structures, and then sum the ΔGs between those transient states of two different structures. It takes usually a huge amount of time when using these two algorithms and this impedes its application for large scale computation. Besides, MM-GBSA and MM-PBSA are becoming more and more popular due to its high efficiency. In MM-PB/GBSA, the free energy of a molecule is calculated using the following equations:

$$G = H_{gas} + G_{solv} - TS \tag{1}$$

$$G_{solv} = G_{solv}^{pol} + G_{solv}^{nonpol} \tag{2}$$

$$S_{conf} = S_{trans} + S_{rot} + S_{vib} \tag{3}$$

Where, in Eq.1, H_{gas} represents the gas phase enthalpy, and can be replaced by E_{gas}, as the PV term is negligible for a molecule in condensed phase. G_{solv} consists of two parts, the polar and nonpolar solvation free energies. The polar part is calculated by the PB/GB model while the nonpolar part is estimated by the solvent accessible surface area. S_{conf} was decomposed into three components; S_{trans} and S_{rot} are translational and rotational energies separately, and they can be calculated by the standard equations. As for the vibrational entropy (S_{vib}), Normal Mode Analysis (NMA) was often used to approximate this part assuming that the vibrational movement around the energy well is harmonic.

Since normal mode analysis is time-consuming, we have developed [6] a solvent accessible surface area (SASA) based approach to calculate S_{conf}. Our method shows good accuracy when compared to NMA. We used SASA to fit S_{conf}, and Shrake-Rupley algorithm [7] was applied to generate a mesh of points to represent the sphere of each atom. Then the distance between points in the surface of each atom and center of each atom were calculated to determine which sphere points are accessible to the solvent. Afterwards, SASA can be computed taking into account the proportion of all points accessible to the solvent.

Here, in order to accelerate the calculation of SASA, a GPU (Graphics Process Unit) based method named MURCIA [8] (Molecular Unburied Rapid Calculation of Individual Areas) was used to replace the Shrake-Rupley algorithm.

There are many molecular simulation packages that have been developed on GPUs. For example, AMBER includes full GPU support for Molecular Dynamics simulations in PMEMD [9], and VMD, a molecular visualization program, employs GPUs for displaying and animating large biomolecular systems using 3-D graphics and built-in scripting [10].

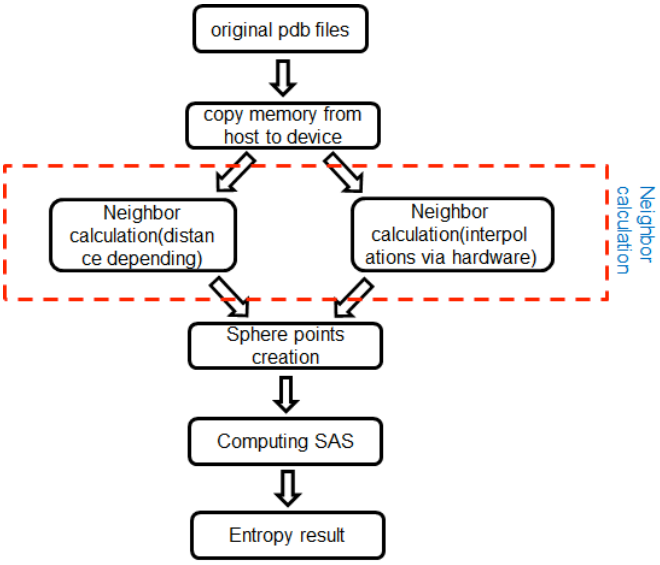

Fig. 1. Flow chart for entropy calculation using MURCIA. Neighbour calculation is carried our either using distances (K1) or via hardware interpolation (K2).

The flow chart of this algorithm for entropy calculations is shown in Figure 1. The key point of this method is to calculate the neighbour atoms and then the SAS of each atom. Once the SAS of each atom has been computed, the following formula is used to calculate the entropy of the protein.

$$S = \sum_{i=1}^{N} w_i(SAS_i + kBSAS_i) \tag{4}$$

$$BSAS_i = 4\pi(r_i + r_{prob})^2 - SAS_i \tag{5}$$

Where w_i has already been calculated in our previous work [6], r_{prob} is 0.8Å, and k is set to 0.461, which was found to be the most suitable one. S represents S_{conf}, and it is the weighted sum of each atom's SAS and buried SAS.

2 Materials and Methods

This section describes how the calculation of entropy is carried out starting from the values of radii and positions of the protein atoms. First, SAS calculation using an improved version of MURCIA [8] is described and then conformational entropy is calculated.

2.1 Copy memory from host to device

Since we use GPU for the SASA calculations, one important step is to copy protein data from host memory to device memory on GPU. In CUDA, we should perform memory allocation, and when calculations are finished, free this memory. In the MURCIA method protein data is read and stored on CPU, and then the size of required GPU memory is calculated and reserved. Finally, all protein data is copied from host to device.

2.2 Neighbour calculation

Neighbour calculation is necessary for computing SASA, since it can accelerate the conduction of calculation. There are two different kernels that we used for the following parts, K1 and K2, as depicted in Figure 1.

In K1, each block performs calculations for an atom *i*. In one block, there are 128 threads by default. For atom *i*, 128 threads are used to calculate distances between other atoms. If the distance is smaller than the cutoff radius, we then consider the atom as a neighbour. If the number of atoms is small than the max blocks size, the block size will be equal to the atoms' number, if not, it will be set as the max block size.

For K2, we will first divide the whole system into several cubic grids, each grid's edge is $2r_{\max}$. And then each atom is set a hash value, atoms which have the same hash value are regarded as in the same cell. Atoms in the surrounding and native cells are regarded as neighbours.

2.3 Sphere points creation

A regular spherical grid of points is placed over the surface of each atom. This idea can simplify the calculation of SAS by just computing the portion of sphere points which fit the condition we set with its neighbour atoms.

There are two ways in which we can set the sphere points or atomic grid. One is to import a sphere points file which contains predefined grid coordinates, and only 72 and 500 point models are available. The other is to use the grid coordinates that we defined in the grids.h file. The number of points ranges from 12 to 1000.

2.4 Compute SAS

After all the preparations have been done, this part is quite easy for calculations.

Same as the generation of sphere points, in this part for K1, each thread is used to qualify whether this point is in the solvent accessible surface of the corresponding atom by computing the distance between this point and the atom's neighbours. If the distance is bigger than the value we set, the out points' number *d_nonburied* increases one. After this calculation is finished, we will get the SAS_i for atom *i*.

$$SAS_i = d\_nonburied / N\_sphere \tag{6}$$

Where *N_sphere* represents the number of each atom's sphere points.

For K2, there is a little difference with respect to k1. The whole system is divided into many cells, and in each cell, there are several atoms. For each sphere point of one atom, the distance is calculated between this point and other atoms which are in the same cell or its neighbour cells. If the distance is not larger than the cutoff, the out point's number increases. The rest of the procedure follows as with K1.

2.5 Conformational entropy

After the calculation of SAS is carried out for each atom, it will be an easy step to calculate the conformational entropy by using formulas 4 and 5. These two formulas are explained and tested in our previous study [6]. Here we used a GPU-based algorithm to accelerate the process of entropy calculation, and we obtained significant improvements in terms of processing speed.

3 Results and Discussions

We used three data sets to test our method. Data Set I contains 12 protein decoys generated by the Rosetta software package (http://www.rosettacommons.org) , Data Set II include relatively 120 larger proteins randomly chosen by us from RCSB protein data bank [11]. Data Set III contains 512 pdb files downloaded from RCSB whose chains' number ranges from 15 to 25.

The first data set was used in our previous study to evaluate the algorithm of entropy calculations.

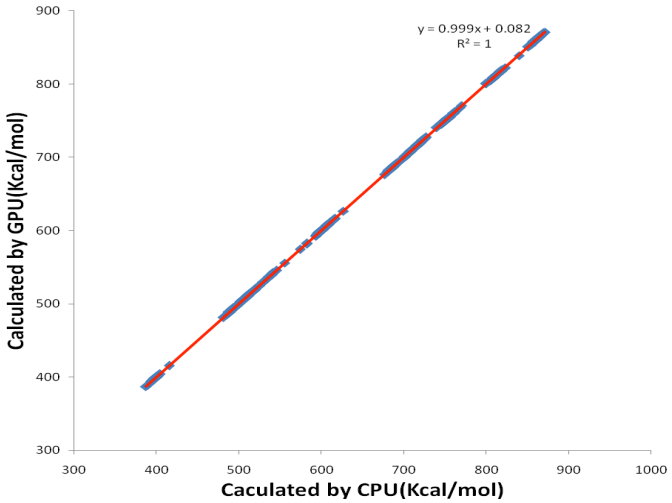

Fig. 2. Running time ratio between CPU and GPU for small proteins when using K2.

The conformational entropies for the proteins in test set I were predicted based on the SASA values calculated by the CPU-based Shrake-Rupley method and those calculated by the GPU-based MURCIA method. We found that the conformation entropies predicted by the CPU-based and GPU-based methods are almost the same with r^2 of 1, as shown in Figure 2. Therefore, the new method based on MURCIA just performs as well as the previous one in predicting the conformational entropies for test set I.

The performance of the CPU-based Shrake-Rupley method and the GPU-based MURCIA method for calculating SAS was compared by averaging the results from 5 independent calculations. The running time ratio between CPU and GPU *versus* the number of atoms is shown in Figure 3. As we can see, the ratio arises as a quadratic form, which means our method will get faster than the previous one when protein size increases. We can find that when the number of atoms increases, the time ratio between CPU and GPU becomes larger. We should mention that for small proteins, the performance of the GPU-based method is not optimal, because GPU can generate a lot of threads at the same time and each atom can calculate their surrounding atoms and SASA in one thread. So for proteins with small number of atoms (10 to 100 atoms), the calculations of the GPU-based method cannot be substantially accelerated compared with those of the CPU-based method. For the second data set, running time obtained is linear with r^2 of 0.9342 (data not shown).

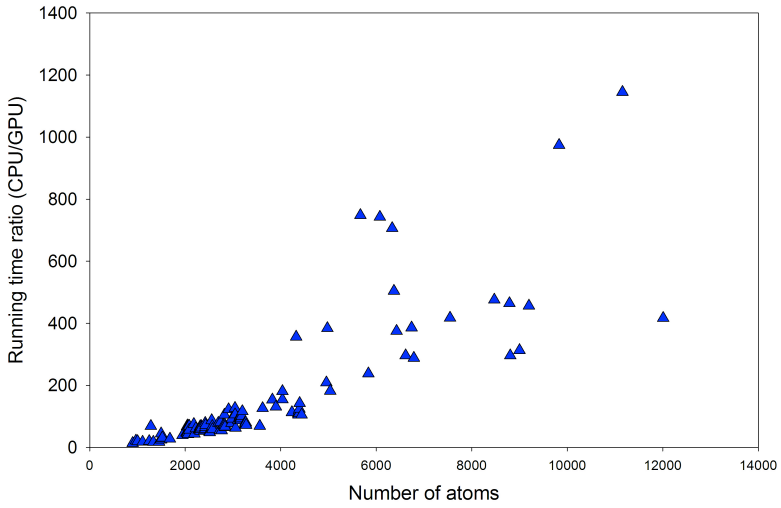

Fig. 3. Running time ratio between CPU and GPU for proteins in the 10^4 size range when using K2.

All the above GPU-based results were generated by K2, which computes the neighbour atoms in a very efficient way. We also used the same data set to test K1,

and the results are illustrated in Figure 4. When the number of atoms is less than 500, the K1 kernel is faster than the K2 kernel, but once the number exceeds 500, K2 outperforms K1 in terms of processing speed.

The proteins in the third data set are larger than those in the second data set. This set was used to characterize the difference of the computing ability between K2 and K1 when the number of atoms exceeds 10,000 or even 50,000. The three data sets' results are illustrated in Figure 4, and we can see from the figure that, when K1 was used to generate the neighbour atoms, our new method based on MURCIA can be regarded as quadratic, and if K2 was used, the MURCIA-based method is just linear with r^2 of 0.99.

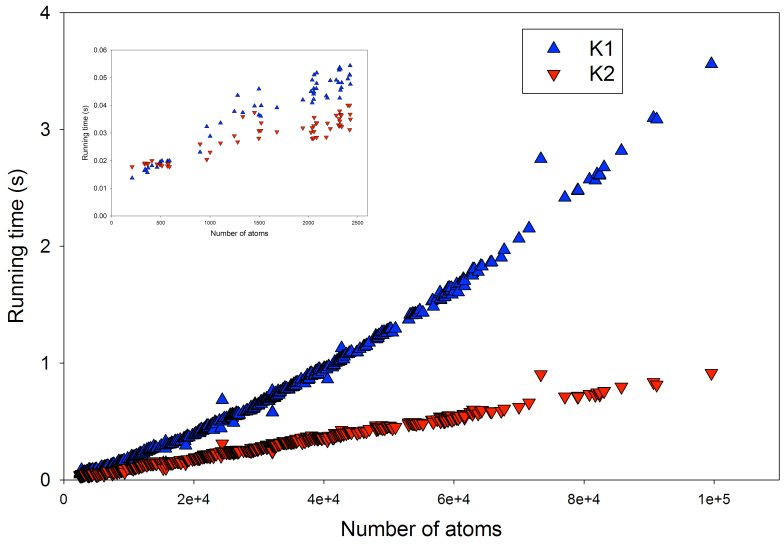

Fig. 4. Running time (seconds) in SAS calculation for K1 and K2 kernels in the 1-10^5 protein size range, and in the 1-700 range (subfigure).

According to the three data sets, we can conclude that when the atom number is smaller than 500, using K1 to get each atom's neighbours is a good choice. Once the number goes over 500, K2 will be faster than K1. And this difference will become larger when protein size increases due to the different behaviour, quadratic *vs.* linear.

4 Conclusions

We have developed a GPU-based method for the calculation of the conformational entropy of molecules. This method can speed up the computing time in a very impressive way. We have tested this method using three data sets. The first set proved that the calculated entropies predicted by the new method are almost the same to those predicted by the method reported in our previous study. Tests for the second data set suggest that this method is much faster than the previous one. The third data set was used to manifest the difference between K1 and K2. K1 is quadric and K2 is linear, and therefore K2 will be faster than k1 when the number of atom exceeds 500.

It will be a very efficient way to use our method to compute the conformational entropy of biomolecules, especially for large structures, and we hope it will be helpful in calculating binding free energies and some other applications of Computer Aided Drug Design (CADD) in the future.

Acknowledgments. This research was supported by the National Science Foundation of China under grants 20973121 and 21173156, the Fundación Séneca (Agencia Regional de Ciencia y Tecnología, Región de Murcia) under grant 15290/PI/2010, by the Spanish MEC and European Commission FEDER under grants CSD2006-00046 and TIN2009-14475-C04.

5 References

1. Aaqvist, J., *Ion-water interaction potentials derived from free energy perturbation simulations.* Journal of Physical Chemistry, 1990. **94**(21): p. 8021-8024.
2. Eldridge, M.D., et al., *Empirical scoring functions: I. The development of a fast empirical scoring function to estimate the binding affinity of ligands in receptor complexes.* Journal of computer-aided molecular design, 1997. **11**(5): p. 425-445.
3. Head, R.D., et al., *VALIDATE: A new method for the receptor-based prediction of binding affinities of novel ligands.* Journal of the American Chemical Society, 1996. **118**(16): p. 3959-3969.
4. Srinivasan, J., et al., *Continuum solvent studies of the stability of DNA, RNA, and phosphoramidate-DNA helices.* Journal of the American Chemical Society, 1998. **120**(37): p. 9401-9409.
5. Straatsma, T. and H. Berendsen, *Free energy of ionic hydration: Analysis of a thermodynamic integration technique to evaluate free energy differences by molecular dynamics simulations.* The Journal of chemical physics, 1988. **89**: p. 5876.
6. Wang, J. and T. Hou, *Develop and Test a Solvent Accessible Surface Area-Based Model in Conformational Entropy Calculations.* Journal of chemical information and modeling, 2012. **52**(5): p. 1199-1212.

7. Shrake, A. and J. Rupley, *Environment and exposure to solvent of protein atoms. Lysozyme and insulin.* Journal of molecular biology, 1973. **79**(2): p. 351-364.
8. E. J. Cepas Quiñonero, H.P.-S., J.M. Cecilia, J.M. García, *MURCIA: Fast parallel solvent accessible surface area calculation on GPUs and application to drug discovery and molecular visualization.* NETTAB 2011 workshop focused on Clinical Bioinformatics, 2011: p. 52-55.
9. Götz, A.W., et al., *Routine microsecond molecular dynamics simulations with amber on gpus. 1. generalized born.* Journal of Chemical Theory and Computation, 2012. **8**(5): p. 1542.
10. Humphrey, W., Dalke, A. and Schulten, K., *VMD - Visual Molecular Dynamics.* J. Molec. Graphics, 1996. **14**: p. 33-38.
11. Berman, H.M., et al., *The protein data bank.* Nucleic acids research, 2000. **28**(1): p. 235-242.