GPU-Accelerated Multi-scoring Functions Protein Loop Structure Sampling

Yaohang Li
Department of Computer Science
North Carolina A&T State University
Greensboro, NC 27411, USA
e-mail: yaohang@ncat.edu

Weihang Zhu
Department of Industrial Engineering
Lamar University
Beaumont, TX 77710, USA
e-mail: Weihang.Zhu@lamar.edu

***Abstract*—Accurate protein loop structure models are important to understand functions of many proteins. One of the main problems in correctly modeling protein loop structures is sampling the large loop backbone conformation space, particularly when the loop is long. In this paper, we present a GPU-accelerated loop backbone structure modeling approach by sampling multiple scoring functions based on pair-wise atom distance, torsion angles of triplet residues, or soft-sphere van der Waals potential. The sampling program implemented on a heterogeneous CPU-GPU platform has observed a speedup of ~40 in sampling long loops, which enables the sampling process to carry out computation with large population size. The GPU-accelerated multi-scoring functions loop structure sampling allows fast generation of decoy sets composed of structurally-diversified backbone decoys with various compromises of multiple scoring functions. In the 53 long loop benchmark targets we tested, our computational results show that in more than 90% of the targets, the decoy sets we generated include decoys within 1.5A RMSD (Root Mean Square Deviation) from native while in 77% of the targets, decoys in 1.0A RMSD are reached.**

Keywords-protein structure modeling; GPU programming; sampling

I. INTRODUCTION

Loop regions are flexible in protein structures, which are active in performing important biological functions. Protein loop modeling is important in protein structural biology for its wide applications, including determining the surface loop regions in homology modeling [1], defining segments in NMR spectroscopy experiments [2], designing antibodies [3], and modeling ion channels [4, 5].

Protein loop structure prediction can be considered as a "mini" protein folding problem, whose goal is to explore the protein conformation space to search for the native. Similar to the protein folding problem, one of the key challenges in protein loop structure modeling is to sample the large conformation space. Most of the efforts in loop modeling [8, 11, 12, 13, 14] are based on optimizing a single scoring (energy) function. However, recent studies have shown that sampling multiple carefully-selected scoring functions has numerous advantages over global optimization.

In this paper, we present a protein loop structure sampling approach by integrating three scoring functions, including an atom pair-wise distance-based scoring function [6], a triplet torsion angle scoring function [7], and a soft-sphere van der Waals energy function [8]. We adopt a Multiobjective Shuffled Complex Evolution Metropolis (MOSCEM) algorithm [9] to sample loop conformations in the multiple scoring functions space and produce diversified loop decoys. To accelerate the sampling process, we implemented the protein loop backbone structure sampling program on a heterogeneous CPU-GPU platform with Compute Unified Device Architecture (CUDA). Our computational results show that the GPU platform, a cost-efficient desktop parallel system, can significantly speed up the loop structure sampling process and thus lead to prompt production of loop decoys with good quality.

The rest of this paper is organized as follows. Section 2 compares the strategies of multi-scoring functions sampling with global optimization in protein loop modeling. The protein loop backbone structure modeling algorithm using multi-scoring functions sampling and its implementation on the CPU-GPU heterogeneous platform are described in Sections 3 and 4, respectively. Section 5 shows our computational results and Section 6 summarizes our conclusions and future research directions.

II. MULTI-SCORING FUNCTIONS SAMPLING VS. GLOBAL OPTIMIZATION

According to the Anfinsen's thermodynamic hypothesis [10], what distinguishes the native conformation from the others is that the native has the minimum free energy of all accessible conformations. Based on the Anfinsen's thermodynamic hypothesis, most of the efforts [8, 11, 12, 13, 14] in protein loop structure modeling focus on globally optimizing a complicated scoring (energy) function, such as soft-sphere energy [8], Rosetta [15], OPLS-AA/SGB [16], DFIRE [17], and OPLS-AA/AGBNP [18], to discover conformations close to the native structure.

Recent studies have shown that sampling multiple scoring functions have several advantages over global optimization. First of all, when multiple scoring functions are presented, it will be easier for the sampling process to escape the deep local minima in one scoring function with the help of the other scoring functions [19]. Secondly and more importantly, integrating multiple scoring functions can tolerate the deficiencies, such as inaccuracy or insensitivity, in individual scoring functions [20, 21]. Thirdly, multi-scoring functions sampling can discover conformations at concave Pareto optimal fronts [20, 26]. Finally, sampling multiple scoring functions does not need to integrate multiple

interaction terms in a single complicated scoring function and thus can avoid the problem of potentially over-counting the same interactions [20]. However, compared to global optimization, the key disadvantage of multi-scoring functions sampling is its computational cost, which is rather expensive due to its requirement of evaluating more than one scoring functions.

III. Algorithm of Sampling Protein Loop Conformation Space using Multiple Scoring Functions

A. MOSCEM Algorithm

The MOSCEM algorithm is a population-based Markov Chain Monte Carlo (MCMC) sampler for multi-objective optimization. While it was originally designed to solve the multi-objective optimization problems for hydrologic models, MOSCEM has found its applications in many other disciplines, such as biology [22] and ecology [23]. The fundamental idea of MOSCEM is to convert the multi-scoring functions space to a single fitness function by assigning fitness to each member in the population based on Pareto dominance relationship [26]. To preserve diversity in sampling, MOSCEM employs the fitness assignment according to the number of external non-dominated points. The fitness calculation is based on the strength s_i of each non-dominated conformation L_i in population P, which is defined as the proportion of conformations in P dominated by L_i. Then, the fitness f_i of a conformation L_i is defined as

$$f_i = \begin{cases} s_i & L_i \text{ is non - dominated} \\ 1 + \sum_{j}^{i \prec j} s_j & L_i \text{ is dominated} \end{cases} . \quad (1)$$

The Metropolis scheme [24] is carried out to construct a Markov chain to sample the fitness function landscape, where the solutions with fitness $f_i < 1.0$ correspond to the ones at the Pareto optimal front of the multiple scoring functions space. The temperature annealing techniques can be used to achieve fast barrier crossing [28] and MCMC equilibrium analysis techniques can also be applied to study the convergence of the sampler. Moreover, MOSCEM divides the population into multiple complexes and evolution takes place independently within each complex, which is a natural fit to the "Single Instruction, Multiple Thread" (SIMT) model in GPU platform.

In this paper, we adopt the MOSCEM algorithm to sample the protein loop torsion angle space. Each loop conformation L_i with n residues is represented by a torsion angle vector (Φ_1, Ψ_1, …, Φ_n, Ψ_n). To simplify the sampling process, the torsion angles of ω are kept constants at their average value of 180° and the bond lengths are fixed as well.

B. Scoring Functions

Three scoring functions are incorporated to sample the protein loop conformation space, including

1) Triplet torsion angle scoring function [TRIPLET]: The triplet torsion angle scoring function [7] measures the favorability of torsion angle configurations based on the distribution of adjacent phi-psi backbone torsion angle pairs in the context of all possible triplet residue conformations derived from structural data in a large loop library.
2) Atom pair-wise distance-based scoring function [DIST]: The atom pair-wise distance-based scoring function measures the favorability of pair-wise backbone atom positions within a protein loop.
3) Soft-sphere van der Waals scoring function [VDW]: The soft-sphere van der Waals scoring function [8] estimates the degree of clashes among the loop residues as well as the potential clashes between the loop residues and the residues in the rest of the protein by calculating the atom-atom, atom-centroid, and centroid-centroid distances.

The main reason to select these scoring functions is that all these functions are backbone scoring functions with side chains implicitly considered. Evaluation of these functions is relatively fast. Moreover, these scoring functions are derived from different measures of loop conformation favorability, where there is little correlation among these scoring functions.

C. Generating New Conformation and Loop Closure Condition

A new conformation is generated from an old conformation by mutating randomly selected torsion angles. In loop sampling, the reasonable loop models are those satisfying the loop closure condition, i.e., given the loop N- and C-terminals, the loop conformation must have a certain length that bridges the ends seamlessly. New conformations generated after mutation generally do not guarantee loop closure. Therefore, we apply the Cyclic Coordinate Descent (CCD) algorithm [25] to the torsion angles in the new conformation, starting from the immediate torsion angle after the mutated ones, to close the loop.

D. MOSCEM Algorithm for Loop Torsion Angle Sampling

Putting every piece of the puzzle together, the descriptive pseudo code of the multiple scoring functions sampling algorithm for protein loop modeling is described as follows. The algorithm can be repeatedly executed to produce as many decoys as needed.

```
// Initialization
Initialize N conformations,L1(ψ1^1,ϕ1^1,...,ψn^1,ϕn^1),...,
LN(ψ1^N,ϕ1^N,...,ψn^N,ϕn^N) in the population randomly
For each conformation Lk(ψ1^k,ϕ1^k,...,ψn^k,ϕn^k)
    [CCD] Adjust torsion angles (ψ1^k,ϕ1^k,...,ψn^k,ϕn^k) of Lk for loop
closure
    [Scoring Functions Evaluation]
    For each loop conformation Lk {
        [EvalVDW]  Evaluate Svdw(Lk)
        [EvalDIST] Evaluate Sdist(Lk)
        [EvalTRIP] Evaluate Striplet(Lk)
    }
// MCMC Sampling
Repeat {
    // Fitness Assignment and Sorting
    [FitAssg] Evaluate fit(Lk)
    [FitSort] Sort L1 ,...,LN in descending order
    // Complex Partition
```

[Partition] Partition the population into M complexes $C_1,...,C_M$, where

$C_1 = (L_1, L_{1+N/M}, L_{1+2N/M}, L_{1+3N/M}, \ldots)$

$C_2 = (L_2, L_{2+N/M}, L_{2+2N/M}, L_{2+3N/M}, \ldots)$

...

$C_M = (L_M, L_{M+N/M}, L_{M+2N/M}, L_{M+3N/M}, \ldots)$

// MOSCEM algorithm

For each complexes C_i,

For each loop conformation L_j in C_i {

[Reproduction]

Generate L_j' by mutating the randomly-selected torsion angles

[CCD] Close loop in L_j'

// Scoring Functions Evaluation

[EvalVDW] Evaluate $S_{vdw}(L_j')$

[EvalDIST] Evaluate $S_{dist}(L_j')$

[EvalTRIP] Evaluate $S_{triplet}(L_j')$

[Metropolis]

Evaluate fit(L_j') against loop conformations in C_i

Accept L_j' to replace L_j with probability

$$\begin{cases} 1 & fit(L_j) \geq fit(L_j') \\ \exp(-(fit(L_j) - fit(L_j'))/T & fit(L_j) < fit(L_j') \end{cases}$$

}

[Assemble]

Assemble $C_1,\ldots,C_M$ back to the population

Adjust temperature T according to acceptance rate

} Until Expected Iterations are reached

IV. GPU Implementation of Multi-scoring Functions Protein Loop Structure Sampling

A. GPU Computing and CUDA Architecture

Compared to a CPU cluster, wherein all the CPUs in the cluster can run independently with different instruction sets, GPU parallel computing follows a different pattern, namely 'Single Instruction – Multiple Thread' (SIMT). With SIMT, a GPU executes the same instruction set on different data elements at the same time. While a CPU cluster has greater flexibility in programming and memory resources, the GPU SIMT pattern has less overhead in parallel computing, which is amenable to intensive and repetitive computation as found in parallel evolutionary algorithms.

Compute Unified Device Architecture (CUDA) is the dominant GPU computing environment in the present. In the CUDA environment, thousands of threads can run concurrently with a same instruction set. Each thread runs a same program called a 'kernel'. A kernel can employ 'registers' as fast access memory. The communication among threads can be realized with 'shared memory', which is a type of very fast memory that allows both read and write access. The communication between CPU and GPU can be done through global device memory, constant memory, or texture Memory on a GPU board. Global device memory is a relatively slow memory location that allows both read and write operations. Texture Memory is relatively fast memory that is read-only. For example, we can employ texture Memory to keep a copy of the solution vectors. Constant memory is fast read-only memory whose size is limited. Texture Memory and constant memory is fast because it is cached for quicker access. The nVidia GeForce GTX 280 GPU hardware employed in this paper has 30 multi-processors. Each multi-processor has 8 processors. This amounts to 240 data-parallel processors on one GPU board. Each multi-processor has 16K shared memory, 16K registers, 64K constant memory, and access to global device memory and Texture Memory for larger data storage. For management purpose, all threads are organized into blocks. Each 'block' can have a maximum of 512 threads. The threads in the same block can communicate through fast shared memory; while between the blocks, communication is possible only with slow global device memory [29].

B. GPU Implementation

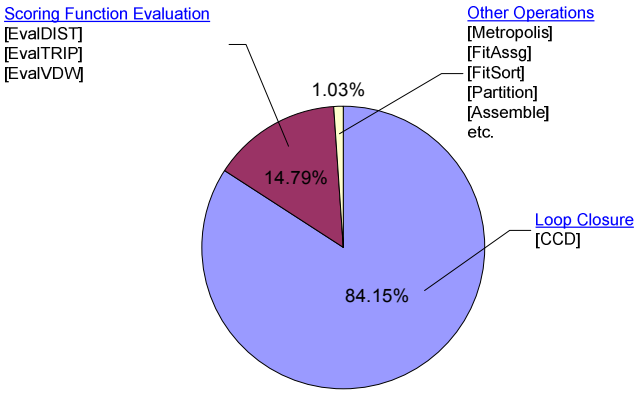


Figure 1. Time profiling of the CPU-only implementation (15,360 conformations)

The CPU-only implementation of the multi-scoring function loop structure sampling algorithm was profiled in order to prioritize the tasks for parallelization. The program was tested on loop 1cex(40:51) with a population size of 15,360 divided into 120 complexes and 100 iterations were carried out. The total execution time was about 3.5 hours on a single CPU. Figure 1 shows the wall-clock time percentages of various components in the sampling program in time profiling. The loop closure and scoring function evaluations occupy near 99% of the overall computation time. Therefore, in the implementation of the multi-scoring functions protein loop structure sampling algorithm on the heterogeneous CPU-GPU platform, we focus on parallelizing the heavy computation components, including those in scoring function evaluations and loop closure, and migrating them to the GPU. At the same time, we intend to reduce the CPU-GPU communication overhead. The other components, such as initialization, sorting, partition, and others, which occupy about 1% of the overall computation time, are kept in the CPU. Although it is possible to sort key-value paired array in a GPU, sorting in GPU in this program will not lead to significant performance improvement. This is due to the fact that the key-value paired array, which depends on the population size, in this program is relatively small.

Figure 2 illustrates a schematic implementation of the multi-scoring function loop structure sampling algorithm on the heterogeneous CPU-GPU platform. Computation components including generating conformations with loop closure, scoring functions evaluation, fitness assignment, and Metropolis acceptance are carried out in GPU while components of fitness sorting, partitioning, and assembling are kept in CPU. Synchronization is required between the GPU kernels and the CPU host code.

To parallelize the computation in the GPU, the evolution of each conformation in the population is assigned to a thread. These threads are organized as blocks in GPU kernels. The number of blocks that can be launched by a GPU is mainly determined by the number of registers and/or the amount of shared memory needed per thread. When compiling the GPU kernels, it is possible to limit the number of registers used per thread. If the actual number of variables is more than the number of registers limit, the exceeding variables will overflow to local memory and thereby degrade performance. For the scoring functions used in the multi-scoring functions loop sampling program, the number of variables is often more than the available registers. Therefore, it is important to make judicious use of the memory.

Judicious allocation of different types of GPU memory is critical to the performance of the program migration to GPU kernels. When the new conformations are generated in the GPU global device memory, they are copied into the GPU texture memory for multi-scoring functions evaluation. Once the scoring functions evaluations are complete, the scores are copied to the texture memory for fitness assignment. The atom pair-wise distance-based scoring function and the triplet scoring function are knowledge-based, which demand large amount pre-calculated data for scoring function evaluation. These pre-calculated scoring function data are also loaded into the texture memory at the beginning since they will not be changed during the function evaluation. Texture memory is used extensively to save time in accessing larger chunk of potentially un-coalesced memory that is more than the capacity of available registers and shared memory. The torsion angles and scoring function values are all organized as arrays data structure in the global memory. Coalesced read/write to the global memory is made possible by reorganizing the data. Here is an example of coalesced memory pattern in the global memory: an n-residue loop conformation is represented by a torsion angle vector (Φ_1, Ψ_1, …, Φ_n, Ψ_n), and as the atomic data structure, each pair (Φ_i, Ψ_i) is represented as a *float2* which is a built-in *struct* type of two floats in CUDA; in the global memory, the i-th *float2* from all residues are grouped and form a tile of n pairs. Constant values, such as the number of conformations, the number of complexes, and the information about atoms and residues within the rest of the protein, are copied into the GPU constant memory at the beginning of the program.

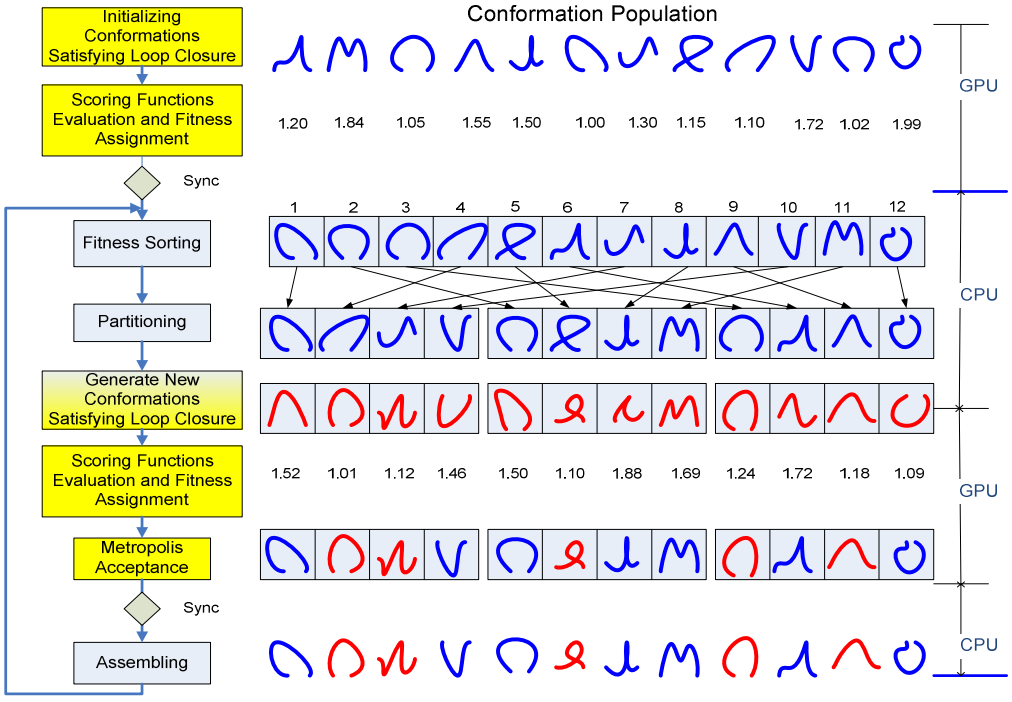


Figure 2. Schematic implementation of the multi-scoring function loop sampling algorithm on the heterogeneous CPU-GPU platform

V. COMPUTATIONAL RESULTS

The computation was carried out on the 53 long loop targets with 10 or more residues in the filtered list provided by Jacobson's loop decoy benchmark library [12]. The CPU used in the computations is an Intel 2.0GHZ quad-core processor with 8GB memory, and the GPU is an nVidia GeForce GTX 280.

A. Population Size Matters

Protein loops, particularly the long ones, have a large conformation space. To find a set of non-dominated solutions instead of a single solution in global optimization, the multi-scoring functions sampling algorithm needs to perform a multimodal search that samples as many structurally diversified conformations at the Pareto-optimal front as possible. A large population size is likely to maintain conformation diversity in the sampling process and thus lead to generation of significant number of structurally-diversified conformations at the end of the sampling trajectory. Figure 3 shows the average number of structurally distinct non-dominated conformations found in 32 independent multi-scoring functions sampling trajectories in loop target 1akz(181:192) as well as the minimum, maximum, and average RMSD of the best decoy found in these trajectories when using the population size of 100, 1000, and 10000. One can find that a larger population size leads to production of more structurally distinct non-dominated conformations and thereby a better chance of generating decoys close to the native.

However, the major drawback of using a large population size is its demand of significantly more costly operations such as scoring function evaluations and loop closing operations using CCD. The SIMT architecture in GPU provides a cost-efficient mechanism to carry out multi-scoring functions sampling with large population size.

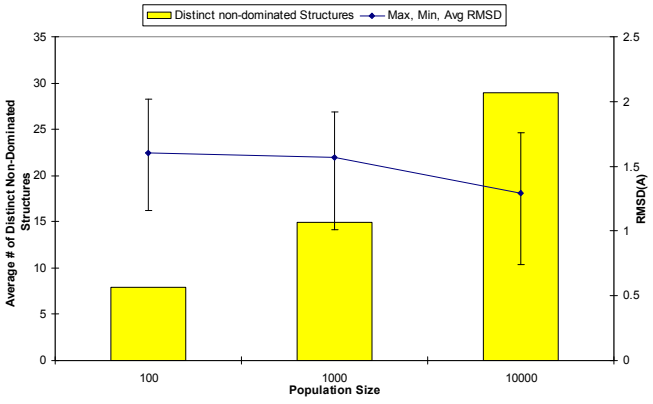


Figure 3. The average number of distinct non-dominated structures found in 32 independent multi-scoring functions MCMC sampling trajectories in loop target 1akz(181:192) and the minimum, maximum, and average RMSD of the best decoys found in these trajectories with population sizes of 100, 1000, and 10000

B. Sampling Speedup by GPU

Figure 4 shows the speedup test on the 1cex(40:51) with different population size (number of threads) and complex size (number of blocks). With 128 threads per block, we tested from 4 blocks up to 120 blocks, i.e., from 512 threads to15,360 threads, with 100 iterations. The speedup increases as the number of threads and blocks increases. For small population sizes (and thus small numbers of threads and thread blocks) there is insufficient work to fully utilize the GPU, and thus many of the GPU's processing units remain entirely idle for small population sizes. It is important to notice that the CPU sampling program using population size of 15,360 demands ~30 times more computational time than the program with population size of 512; in contrast, the computational time in CPU-GPU sampling program using population size 15,360 is only 2.39 times of the one with population size 512. This indicates that the heterogeneous CPU-GPU architecture is cost-efficient in carrying out multi-scoring functions protein loop structure sampling with large population size.

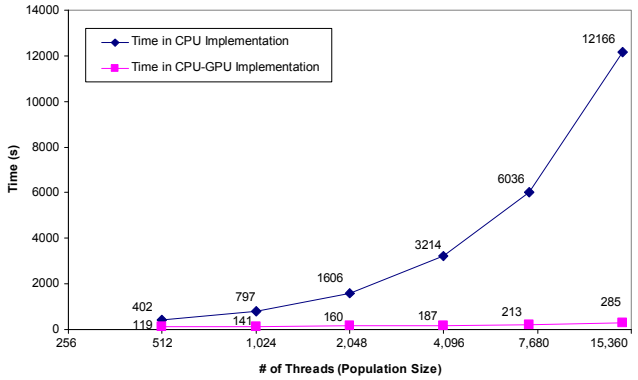


Figure 4: Comparison of computational time using different the number of threads on 1cex(40:51) with 100 iterations in CPU implementation and CPU-GPU implementation

Table I shows the speedup evaluation on a set of 12-residue protein loops with 15,360 threads and 100 iterations. One can find that the speedup is consistent at ~40 for loops in different protein.

TABLE I. SPEEDUP COMPARISON FOR SEVERAL 12-RESIDUE LOOPS (15,360 THREADS, 100 ITERATIONS)

Protein	Start Res.	End Res.	CPU Impl. Time (sec)	CPU-GPU Impl. Time (sec)	Speedup
1cex	40	51	12,166	285	42.6
1akz	181	192	21,440	532	40.3
1xyz	813	824	9,248	236	39.2
1ixh	160	171	17,790	476	37.3
153l	98	109	22,814	532	42.9
1dim	213	224	24,124	441	54.8

Since the CPU and CPU-GPU implementations use different random number sequences, they do not produce exactly structurally-identical decoys. However, our analysis found that the decoys generated by the CPU program and those generated by the CPU-GPU program lead to similar structure clusters, which indicates that the CPU-GPU implementation is functionally equivalent to the CPU implementation.

TABLE II. COMPUTATIONAL TIME OF VARIOUS GPU TASKS ON SAMPLING 1CEX(40:51) WITH 15,360 THREADS AND 100 ITERATIONS

Category	Method	#calls	GPU (μsec)	% GPU time
Kernel	[CCD]	101	208,033,000	75.2
Kernel	[EvalDIST]	101	39,579,836	14.3
Kernel	[EvalVDW]	101	23,232,000	8.39
Kernel	[EvalTRIP]	101	114,183	0.04
Kernel	[FitAssg] within Population	101	3,664,410	1.32
Kernel	[FitAssg] within Complex	100	38,969	0.01
Mem sync	memcpyHtoA	10	80,341	0.02
Mem sync	memcpyHtoD	518	89,612	0.03
Mem sync	memcpyDtoA	202	278,483	0.1
Mem sync	memcpyDtoH	706	1,512,290	0.54
Mem sync	memcpyDtoD	303	1,897	0

TABLE III. REGISTERS PER THREAD AND OCCUPANCY FOR EACH MULTIPROCESSOR (SHARED MEMORY IS NOT USED)

Kernel	Registers/thread	Occupancy
[CCD]	32	50%
[EvalDIST]	32	50%
[EvalVDW]	32	50%
[FitAssg] within Population	8	100%
[EvalTRIP]	20	75%
[FitAssg] within Complex	5	100%

Data in Table II is obtained by using the CUDA Visual Profiler, which is a profiling tool provided by the CUDA Software Development Kit (SDK). The 'GPU' column is the total amount of time on GPU in executing the task. The '% GPU time' column is found by dividing each GPU task times by the total amount of time spent in the GPU. Table 2 indicates that the GPU kernel computation takes the majority of the time to carry out various multi-scoring functions sampling operations, including CCD and evaluations of multiple scoring functions, while the memory synchronization between the CPU and GPU is maintained at a reasonable low level. In the kernel compilation settings, the number of allowed registers per thread is limited to 32. This will allow enough threads to be launched on each multiprocessor, that is, this will improve the occupancy of each multiprocessor during the kernel running. However, if a kernel in fact needs more than 32 registers per thread for the variables, the overflown variables will be in the local memory which is in fact part of the much slower device memory. This has been a concern in the kernel implementation. The CCD kernel is one such extreme case where excessive number of local variables needs to be defined. Table III shows the number of registers per thread as the result of kernel compilation and the occupancy for each kernel.

C. *Efficiency of Multi-Scoring Functions Sampling in Loop Modeling*

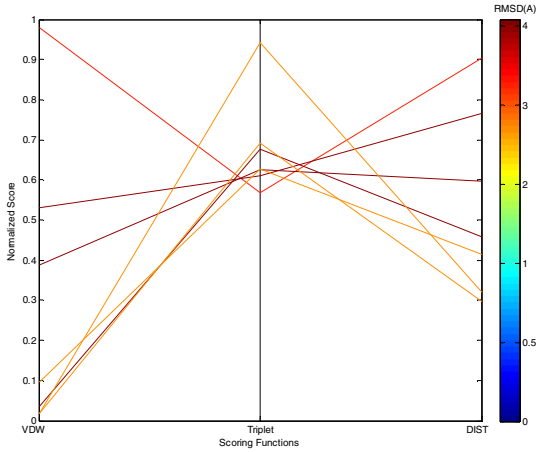
We carried out multi-scoring functions sampling on the long loop targets to produce a set of 1,000 decoys. We used a population size of 15,360 and 100 iterations for each trajectory. The distinct non-dominated conformations produced in a trajectory, i.e., the maximum deviation of whose torsion angle from those in the decoy set is at least 30°, are generated as a new decoy and added in the decoy set. The sampling trajectory was repeated with a different random number seed until 1,000 decoys are reached in the decoy set. Table IV summarizes the number of targets whose decoy set include conformations close to the native loop. One can find that in 77.4% and 90.6% of the long loop targets, the 1,000 decoys generated by multi-scoring functions sampling include good quality decoys with RMSD within 1.0A and 1.5A, respectively.

TABLE IV. NUMBER OF TARGETS THAT MULTI-SCORING FUNCTIONS SAMPLING CAN PRODUCE HIGH-RESOLUTION DECOYS

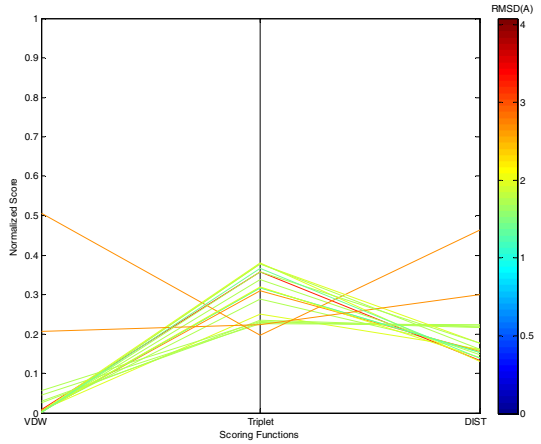
# of Residues	# of Benchmark Targets	< 1.0A	<1.5A
10	27	23	25
11	17	12	16
12	9	6	7
Total	53	41 (77.4%)	48 (90.6%)

Figure 5 shows the evolution of the non-dominated conformations during the multi-scoring functions sampling process. One can find that when the loop conformations are randomly generated (Figure 5(a)), there are few non-dominated conformations (7) and all of them are erroneous with RMSD > 2.0A. After 20 iterations (Figure 5(b)), the scores in non-dominated conformations are reduced and more non-dominated conformations (19) appears, including those with RMSD < 2.0A. When the sampling process reaches 100 iterations, 63 non-dominated conformations are found and the native-like ones (RMSD < 0.5A) emerge (Figure 5(c)). It is interesting to notice that the conformations with the lowest TRIPLET or DIST scores do not yield low RMSD values; however, sampling scoring function space composed of TRIPLET, DIST, and VDW

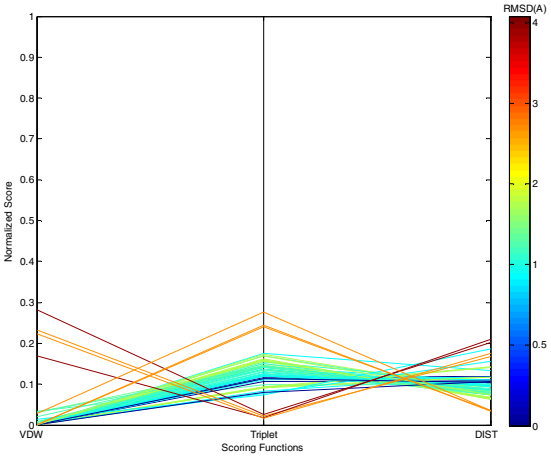
explores conformations with various compromises of these scoring functions and thus reveals decoys with good quality.



(a) Initialization (7 non-dominated conformations)



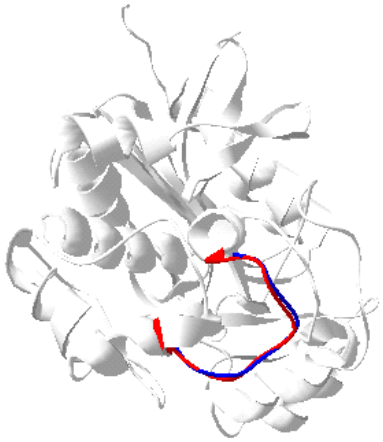
(b) After 20 iterations (19 non-dominated conformations)



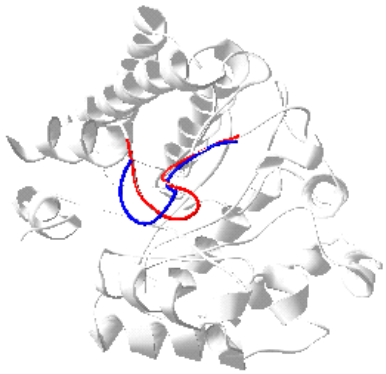
(c) After 100 iterations (63 non-dominated conformations)

Figure 5. Evolution of the non-dominated conformations in 5pti (7:17) during multi-scoring functions sampling

In more than 77% of the targets we tested, the multi-scoring functions sampling can lead to generation of decoys within 1A RMSD from the native. Figure 6(a) shows the best decoy found in 3pte(91:101) with 0.42A RMSD, which is very close to the native. The only target that we fail to obtain a decoy within 2A RMSD is 1xyz(813:824), where the best decoy found in 1xyz(813:824) with 2.15A RMSD is shown in Figure 6(b). The main reason is that the 1xyz(813:824) loop is deeply buried into the protein, which leads to strong interactions with the other atoms in 1xyz and thereby relatively high score values in all three scoring functions we used. The difficulty of modeling 1xyz(813:824) is also discussed in [27].



(a) 3pte(91:101) RMSD=0.42A



(b) 1xyz(813:824) RMSD=2.15A

Figure 6. Best decoys obtained in 3pte(91:101) and 1xyz(813:824) (Red: native, Blue: best decoy generated)

VI. Conclusions and Future Research Directions

In this paper, we present an approach to using GPU to accelerate multi-scoring functions sampling in protein loop backbone structure modeling. When a large population size at the magnitude of 10,000 is used, the sampling process typically takes around 2 days to produce 1,000 decoys in most of the loop targets. The implementation of the sampling program on heterogeneous CPU-GPU platform

can typically lead to speedup of ~40, which can significantly reduce the sampling time from ~2 days to ~1 hour. This leads to fast generation of diversified backbone decoys, where good quality decoys can be included.

Our future research directions will include incorporating more accurate scoring functions in the sampling process and modeling the difficult targets. Moreover, although we know that the population size of 15,360 can fully utilize the GTX 280 GPU we used in this study, we do not know the optimal population size for MOSCEM to sufficiently cover the Pareto optimal front. The optimal population size likely depends on the size of the loop as well as the scoring functions selected. Determining the optimal population size will be one of our research focuses in the future.

ACKNOWLEDGMENT

The work is partially supported by NSF under grants CCF-0829382 and CCF-0845702 to Y. Li and Lamar Research Enhancement Grants, a grant from Texas Hazardous Waste Research Center, and NSF grant DUE-0737173 to W. Zhu.

REFERENCES

[1] R. E. Bruccoleri, "Ab initio loop modeling and its application to homology modeling," Methods in Molecular Biology, **143**: 247-264, 2000.

[2] O. Y. Dmitriev, R. H. Fillingame, "The rigid connecting loop stabilizes hairpin folding of the two helices of the ATP synthase subunit c," Protein Science, **16**(10): 2118-2122, 2007.

[3] A. C. Martin, J. C. Cheetham, A. R. Rees, "Modeling antibody hypervariable loops: a combined algorithm," PNAS, **86**(23): 9268-9272, 1989.

[4] A. Tasneem, L. M. Iyer, E. Jakobsson, E., L. Aravind, "Identification of the prokaryotic ligand-gated ion channels and their implications for the mechanisms and origins of animal Cys-loop ion channels," Genome Biol., **6**(1) R4, 2005.

[5] V. Yarov-Yarovoy, D. Baker, W.A. Catterall, "Voltage sensor conformations in the open and closed states in ROSETTA structural models of K+ channels," PNAS, **103**: 7292-7297, 2006.

[6] A. Rojnuckarin, S. Subramaniam, "Knowledge-based interaction potentials for proteins", Proteins: Structure, Function, and Genetics **36**: 54-67, 1999.

[7] I. Rata, Y. Li, E. Jakobsson, "Backbone statistical potential from local sequence-structure interactions in protein loops," in press, Journal of Phys. Chem. B. 2009.

[8] H. Zhang, L. Lai, Y. Han, Y. Tang, "A Fast and Efficient Program for Modeling Protein Loops," Biopolymers, **41**: 61-72, 1997.

[9] J. A. Vrugt, H. V. Gupta, L. A. Bastidas, W. Boutem, S. Sorooshian, "Effective and Efficient Algorithm for Multiobjective Optimization of Hyrdologic Models," Water Resources Research, **39**(8): 1214-1232, 2002.

[10] C. B. Anfinsen, "Principles that Govern the Folding of Protein Chains," Science, **181**: 223-230, 1973.

[11] C. A. Rohl, C. E. Strauss, D. Chivian, D. Baker, "Modeling structurally variable regions in homologous proteins with rosetta," Proteins, **55**:656-677, 2004.

[12] M. P. Jacobson, D. L. Pincus, C. S. Rapp, T. J. F. Day, B. Honig, D. E. Shaw, R. A. Friesner, "A hierarchical approach to all-atom protein loop prediction," Proteins: Structure, Function, and Bioinformatics, **55**(2): 351-367, 2004.

[13] C. Zhang, S. Liu, Y. Zhou, "Accurate and efficient loop selections using DFIRE-based all-atom statistical potential," *Protein Science,* **13**: 391-399, 2004.

[14] A. K. Felts, E. Gallicchio, D. Chekmarev, K. A. Paris, R. A. Friesner, R. M. Levy, "Prediction of Protein Loop Conformations using the AGBNP Implicit Solvent Model and Torsion Angle Sampling," J. Chem. Theory Comput., **4**(5): 855-868, 2008.

[15] K. T. Simons, R. Bonneau, I. Ruczinski, and D. Baker, "Ab initio Protein Structure Prediction of CASP III Targets Using ROSETTA," Proteins, **37**(3): 171-176, 1999.

[16] W. L. Jorgensen, D. S. Maxwell, J. Tirado-Rives, "Development and testing of the OPLS all-atom force field on conformational energetics and properties of organic liquids," J. Am. Chem. Soc., **118**: 11225-11236, 1996.

[17] H. Zhou, Y. Zhou, "Distance-scaled, finite ideal-gas reference state improves structure-derived potentials of mean force for structure selection and stability prediction," *Protein Science*, **11**: 2714-2726, 2002.

[18] E. Gallicchio, R. M. Levy, "AGBNP: an analytic implicit solvent model suitable for molecular dynamics simulations and high-resolution modeling," J. Comput. Chem., **25**(4): 479-499, 2004.

[19] J. Handl, S. C. Lovell, J. Knowles, "Investigations into the Effect of Multiobjectivization in Protein Structure Prediction," Lecture Notes in Computer Science, **5199**: 702-711, 2008.

[20] Y. Li, I. Rata, E. Jakobsson, "Multi-Scoring Functions Sampling in Protein Loop Structure Prediction," submitted to Applied Mathematics and Computation, 2009.

[21] X. Gao, D. Bu, J. Xu, M. Li, "Improving Consensus Contact Prediction via Server Correlation Reduction," BMC Structural Biology, **9**:28-42, 2009.

[22] J. A. Vrugt, J. van Belle, W. Bouten, "Pareto front analysis of flight time and energy use in long-distance bird migration," Journal of Avian Biology, **38**(4): 432:443, 2007.

[23] K. A. Rose, B. A. Megrey, F. E. Werner, D. M. Ware, "Calibration of the NEMURO nutrient-phytoplankton-zooplankton food web model to a coastal ecosystem: Evaluation of an automated calibration approach," Ecological Modelling, **202**: 38-51, 2007.

[24] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of State Calculations by Fast Computing Machines", J. of Chem. Phys., **21**:1087-1092, 1953.

[25] A. A. Canutescu, R. L. Dunbrack, "Cyclic Coordinate Descent: A robotics algorithm for protein loop closure," Protein Science, **12**:963-972, 2003.

[26] K. Deb, "Multi-Objective Optimization using Evolutionary Algorithms," Wiley 2001.

[27] K. R. Rossi, C. A. Weigelt, A. Nayeem, S. R. Krystek, "Loopholes and missing links in protein modeling," Protein Sci., **16**(9):199-2012, 2007.

[28] Y. Li, V. A. Protopopescu, A. Gorin, "Accelerated Simulated Tempering," Physics Letters A, **328**(4): 274-283, 2004.

[29] nVidia, 2010, CUDA Programming Guide Version 2.3. nVidia, Available at: http://www.nvidia.com/object/cuda_get.html.