PTM-MatAlign: A Fast GPU-based Algorithm for Pairwise Protein Structure Alignment

Nada M. A. Mohammed
Faculty of Computer and Information Science
Ain Shams University

Hala M. Ebeid
Faculty of Computer and Information Science
Ain Shams University

Mostafa G. M. Mostafa
Faculty of Computer and Information Science
Ain Shams University

Mahmoud E. A. Gadallah
Modern Academy for Computer Science and Information Technology

ABSTRACT
Although the pairwise protein three-dimensional (3D) structure alignment is vital in structural bioinformatics, its complexity is categorized as non-deterministic polynomial-time hard (NP-hard). Hence, researchers strive to develop algorithms to overcome the heavy computation complexity. Most of their attempts tend to achieve more accurate alignment results regardless of the computational execution time. Therefore, finding a fast alignment algorithm with accurate results is still an outstanding task. Recently, General Purpose Graphical Processing Units (GPGPUs) can execute the many time-consuming algorithms faster than the CPUs can. This paper proposes the GPU-based implementation of the MatAlign algorithm which is based on the two-level alignment of protein. This GPU implementation yields about 11 increase in speed over its CPU-based, single-core implementation on GPU GeForce GTX 860M (640 cores, 2GB RAM) and Intel Core i7-4710HQ (2.50GHz, 8GB RAM, 8 cores) CPU. In order to achieve more accurate results, PTM-MatAlign is implemented to use the Template Modeling Score (TM-score) instead of the MatAlign regular score function.

General Terms
Protein Structure Alignment, GPU Parallel Computing, Multi-core Parallel Programming.

Keywords
Structure Alignment, CUDA, GPU Parallel Computing, TM-Score, MatAlign.

1. INTRODUCTION
Since the 1970s, great attention has been directed to computational molecular biology [1]. Special attention has been given to protein 3D structure alignment since it plays a crucial role in many molecular biology fields. The protein 3D structure alignment, which can be described as superimposing protein structures to find which amino acid residues are equivalent between them, helps scientists draw useful conclusions about the history of proteins, possibility of their interactions, functional similarities, and many other operations [2]. The protein structure alignment process can be divided into two categories: local and global. The very basic difference between local and global methods is that in a local alignment, a portion of a query protein is aligned with a portion of the subject protein, whereas in a global alignment, the entire structure of both query and subject protein is aligned. Therefore, global alignment may end up with a lot of gaps in if the sizes of query and subject are dissimilar. Global methods consist of fold-based alignment approaches, while local methods can be categorized into template-based and surface-based alignment approaches.

In fold-based alignment approaches, the alignment process focuses on different levels of details to calculate different types of alignments, where protein structures are represented on the level of atom coordinates or a higher level of Secondary Structure Elements (SSEs). Since the alignment process is an NP-Hard problem, this task is usually done by heuristics techniques. The most famous heuristic algorithms among the fold-based approaches that are built on atom representations are the DALI [3], SSAP [4], TM-Align [5], FAST [6], MatAlign [7], LNA [8], CSA [9], and CAB-Align [10]. These kinds of algorithms provide accurate alignment results. However, their computation expense is very large which makes them inappropriate for large size proteins.

Among the fold-based methods that are built on SSE representations are VAST [11], CE [12], FATCAT [13], FlexProt [14],YAKUSKA [15], STRIDE [16], CLePAPS [17], ALADYN [18], DEDAL [19], MICAN [20], ASSIST [21], GR-Align [22], and PCalign [23]. Not all fold-based approaches are heuristic, PAUL [24] and [25] are proposed in the literature to find the near-optimal alignment of two proteins. Although such approaches are computationally feasible, they are too slow to be a used daily.

On the other hand, among local alignment methods, especially in template-based alignment approaches, the alignment process focuses on finding the three-dimensional arrangements of the important residues, such as catalytic dyads, triads or other catalytic centers. Some of the well-known algorithms in this approach are ASSAM [26], TESS [27], MUSTA [28], [29], [30], [31], [32], [33], and [34]. Most template-based methods have the drawback of requiring a great knowledge about protein structures because templates either have to be manually pre-determined or are resultant from information mining.

Likewise, surface-based approaches share similar methodologies with template-based approaches where in both approaches focus on finding common substructures in proteins. The main difference is that template-based alignment approaches are developed to deal with small residue patterns, whereas surface-based alignment approaches consider much larger areas. Furthermore, surface-based approaches focus on residues on the surface of the protein. Also, it needs no assumptions about the location, number or orientation of residues. The most well-known algorithms in this group are FEATURE [35], pvSOAR [36], SURFNET [37], SOIPPA [38], 3D-SURFER [39], SEGA [40], PL-PatchSurfer [41], and Layers [42].

Local and global alignment algorithms are not suitable to be used as an everyday tool due to their heavy computation. In order to mitigate this problem, new GPU-based algorithms have been developed to speed up the database searching and

alignment processes while preserving its accuracy. For database search, many algorithms have shown hopeful results with high speed up over their CPU implementations such as SATableau Search [43] which performs two-level parallelization, where the outer level runs alignments in parallel, and the inner level parallelizes each alignment. Also, ppsAlign [44] and GPU-CASSERT [45], are similar to SATableau Search in which they consist of two dynamic programming alignment levels. However, GPU-CASSERT differs in using a different representation of Protein Secondary Structure Elements (SSEs) while ppsAlign differs in aligning structures at the residue level.

This paper presents an enhanced GPU-based parallel implementation of the pairwise alignment algorithm MatAlign [7]. MatAlign was choosen for several reasons such as 1) MatAlign does not require any secondary structure information, 2) MatAlign can be easily parallelized because it consists of multiple mutually-independent dynamic programming procedures, and 3) MatAlign experimental results had shown that it had achieved better alignment results than DALI, CE and other often-used alignment algorithms.

The rest of this paper is organized as follows: Section 2 introduces a brief explanation of the MatAlign computational procedure. Moreover, it describes the PTM-MatAlign framework architecture. Also, it explains how PTM-MatAlign is enhanced to use TM-Score. Then the GPU-based implementation of the proposed algorithm PTM-MatAlign is provided. In Section 3 the experimental results of the proposed implementation are provided then discussed in Section 4. Finally, the conclusion is presented in Section 5.

2. METHODS

2.1 Preliminaries

The proposed algorithm is based on a classic alignment algorithm named MatAlign [7]. In MatAlign, the pairwise protein 3D structure alignment is reached, simply, by aligning the distance matrices of the two query proteins specified by the user instead of comparing their original 3D structure. These distance matrices are built by calculating the distance between Alpha-Carbon (C_α) atoms. Alignment of distance matrices is based on the fact that two structurally similar atoms, one from each protein, have the similar distance profiles. In principle, MatAlign applies a two-level dynamic programming approach by first mapping the protein structures into two-dimensional (2D) distance matrices and aligns them to find the initial alignment. Second, initial alignment is refined to reach the optimum alignment score.

Level 1: Finding Initial Alignment

Assume [DM_A]and [DM_B] represent distance matrices for two query proteins A and B respectively. In MatAlign first level, a score matrix SM is calculated by aligning each row from [DM_A] against each row from [DM_B] by using a match function similar to the one used in the classical Needleman-Wunsch [46]. This match function is used to determine the matching degree between two alpha carbon atoms distance values d_1 and d_2. The match function can be defined as:

$$M(d_1, d_2) = \begin{cases} \dfrac{\alpha}{|d_1 - d_2| + \alpha} & if\ |d_1 - d_2| \le T_{Match} \\ 0 & otherwise \end{cases} \quad (1)$$

where α is the score adjusting weight with default value = 0.7, and T_{Match} is the difference threshold of the distances with default value 1.6Å. This match function is used in the dynamic programming's selection step. After executing the dynamic programming, the matching score of the two given rows is reached.

Algorithm 1 shows the row-row alignment algorithm. First, the algorithm follows the same methodology of the classical Needleman-Wunsch dynamic programming algorithm [46] to create a score matrix SM. After that, another Needleman-Wunsch dynamic programming algorithm is applied on the SM to generate the initially aligned pairs. Moreover, the exactly aligned pairs are traced back from the dynamic programming's matrix F by a recursive algorithm.

Algorithm 1 A single-thread version of first level of MatAlign algorithm

1: **Procedure** GetInitialAlignment(DM_A, DM_B)

2: Let SM be the similarity matrix

3: **for** row i **in** DM_A **do**

4: **for** row j **in** DM_B**do**

5: $SM[i,j] \leftarrow$ row-row matching score of i^{th} row of $DM_A$and j^{th}row of DM_B

6: **end for**

7: **end for**

8: $GS \leftarrow 0$ //GS is the Gap Score

9: $F \leftarrow GS$ // Let F be a second similarity matrix

10: **for** row i **in** DM_A**do**

11: **for** row j **in** DM_B**do**

12: $F[i,j] \leftarrow Max\ (F[i-1,j] + GS, F[i-1,j-1] + SM[i,j], F[i-1,j] + GS)$

13: **end for**

14: **end for**

15: GetAlignment (SM, F)

16: **end procedure**

Level 2: Alignment Refining

The Root Mean Square Deviation (RMSD) [47] is the most commonly used tool to measure statistically the similarity between two protein structures. However, it is not enough in the most cases because the alignment length has to be considered too. MatAlign further refines the alignment using both the RMSD value (Δ) and the number of aligned pairs by using the same scoring function (S) used in [48] as shown in Equation (2). Since the initial alignment resulting from level 1 is not usually an optimum in terms of S, the alignment is iteratively refined until S cannot be further improved.

$$S = \frac{3 \times N}{1 + \Delta}, \qquad \Delta = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(A_{al}[i] - (R.B_{al}[i] + T)\right)^2} \quad (2)$$

Where here N is the number of aligned residues, and R and T are the rotation matrix and translation vector applied on the aligned residues of protein B in order to yield the minimum RMSD value.

For further details about performance assessments, experimental results and efficiency tests are in illustrated in

[7]. Based on the heavy computations in the row-row alignment step, it is decided to parallelize the first level of MatAlign.

2.2 The Proposed Framework Architecture

PTM-MatAlign depends only on the NVIDIA GPU architecture and does not need any complex hardware requirements. The framework of PTM-MatAlign is illustrated in Fig 1. Obviously, The PTM-MatAlign algorithm consists of 7 steps: 1) Query protein structure files A and B are read from the user. 2) Essential features are extracted from protein files. 3) Distance matrices DMA and DMB are computed based on the extracted features. 4) The first parallel phase in first alignment level is executed on all query protein A rows against query protein B rows. 5) The second parallel phase in the first alignment level is executed on the resulted score matrix to find the aligned pairs list. 6) An optimization is performed using Compute Unified Device Architecture (CUDA) dynamic parallelism to accelerate the alignment process. In 7), 8) and 9), a refinement step is applied by calculating alignment score using TM-Score to reach the optimum alignment. Steps 1, 2, 7, 8 and 9 are executed on the host (CPU), while the rest of steps, the most time-consuming parts, are implemented on the device (GPU) as kernels. Through the alignment process, input data are saved in GPU global memory, whereas the results from all GPU kernels are sent to CPU memory.

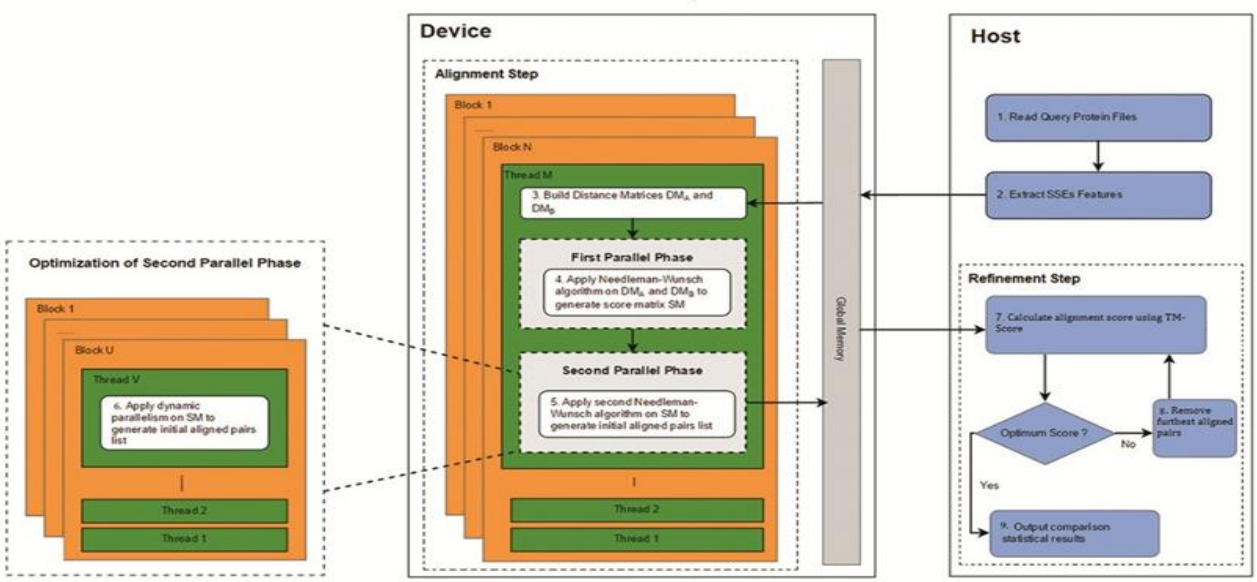

Fig 1: The PTM-MatAlign Framework Architecture with its CPU and GPU Processes.

2.3 The Proposed Algorithm (PTM-MatAlign

Even though MatAlign is much better than DALI and CE in terms of speed and accuracy [7], yet it is not as accurate as, for example, TM-Align [5]. In order to improve the alignment accuracy, it is decided to enhance MatAlign by applying TM-Score function in addition to RMSD in the score function calculation.

2.3.1 PTM-MatAlign Score Function Implementation

As previously mentioned in section 2.1, RMSD is the most commonly used tool for specifying the similarity locally between two protein structures [49]. Despite that, RMSD has a major weakness that affects the alignment accuracy where in RMSD gives equal weight to all distances between all residues. This may lead that a high RMSD could contain a small number of local structural deviations, even in similar structures. Furthermore, the average RMSD of randomly related proteins depends on the length of the compared structures. On the other hand, TM-score [50] can easily overcome this weakness by using a variation of Levitt–Gerstein (LG) weight factor [51] that gives stronger weights to small distance residues than those with larger distances.

This efficiency of TM-Score encourages us to use it in the refinement step instead of MatAlign regular score function [48] to improve the alignment results. The TM-score function is defined as following:

$$\text{TM} - \text{Score} = \text{Max}\left[\frac{1}{L_{Target}}\sum_{i}^{L_{ali}}\frac{1}{1+\left(\frac{d_i}{d_0(L_{Target})}\right)^2}\right] \quad (3)$$

where L_{Target} is the number of residues of the target protein, L_{ali} is the number of aligned residues, d_i is the distance between the i^{th} pair of aligned residues, and d_0 (L_{Target}) is a distance parameter that normalizes the distance so that the average TM-score is not dependent on the protein size. The word "Max" denotes the maximum value after optimal spatial superposition. d_0 is defined as following:

$$d_0(L_{Target}) = 1.24\sqrt[3]{L_{Target} - 15} - 1.8 \quad (4)$$

As shown in Algorithm 2, during the process of score calculating, the farthest pair of residues is picked to be removed in case of score convergence because they are not contributing to the alignment quality and keeping them may cause inaccurate alignment results. The refinement iteration is stopped when the alignment score is no longer changing.

Algorithm 2 The refinement step of PTM-MatAlign algorithm

1: **Procedure** RefineAlignment(N, A_{init}, B_{init})

2: $S_{old} \leftarrow 0$ /* The old score value */

3: Let $N\backslash$ be the updated number of aligned pairs

4: $N^{\setminus} \leftarrow N,\ A_{final} \leftarrow A_{init}, B_{final} \leftarrow B_{init}$

5: **while** (**true**)

6: S = TM-Score ($N^{\setminus}$, A_{final}, B_{final})

7: **if** (S diverges) **then break**

8: **else**

9: $S_{old} = S$

10: Remove the farthest pair from A_{final} and B_{final}

11: Decrement $N^{\setminus}$ by 1

12: **end if**

13: **end while**

14: $A_{final} \leftarrow A_{init}$, $B_{final} \leftarrow B_{init}$

15: **return** A_{final}, B_{final}, $N^{\setminus}$

16: **end procedure**

2.3.2 PTM-MatAlign Parallel Implementation

There are two main time-consuming steps that need to be parallelized. First, the heavy calculation in row-row alignment at Algorithm 1 lines 3 – 7 and second, the dynamic programming performed on the score matrix to generate the list of aligned pairs of Algorithm 1 lines 10 – 15. PTM-MatAlign parallelizes the above two steps to accelerate the alignment process.

I. First Parallel Alignment Step

Algorithm 3 describes a pseudo-code of the parallel implementation of the first alignment step. This algorithm parallelizes the row-row alignment operation, which is the most time-consuming part of the first alignment level, as one GPU kernel (i.e. GPU function). Each row from the first protein is aligned against each row from the second protein and stores the similarity results in the global memory. Since the total number of blocks that can concurrently execute a kernel depends on the maximum global memory size of the GPU, in this model, the total number of blocks B_t is determined in terms of the number of amino acids in query proteins, A and B, and the total number of threads T_t in each block where B_t= |B|/ T_t* |A|.

In terms of memory usage, each thread requires one similarity matrix of size (|A|+1)*(|B|+1). Therefore, the total memory space needed to execute all threads in parallel is |A|*|B|*(|A|+1)*(|B|+1). This amount of data exceeds the limit of GPU local and shared memory in case of large size proteins. Therefore, the only rescue is the use of global memory to overcome the limitation of GPU memory resources. In order to optimize the use of global memory, each thread remembers only the last two rows of the similarity matrix. This is satisfactory to determine the maximum score of the similarity matrix, which is needed to check whether the two query proteins are similar or not.

Algorithm 3 GPU parallel implementation of the first alignment step

1: **kernel** Row-RowAlignmentKernel(DM_A, DM_B)

2: Let SM be the similarity matrix

3: Let patch = |B| / T_t

4: **for** i ← BlockId / patch **to** |A| *in parallel* **do**

5: **for** j ← (BlockId mod patch) * T_t + ThreadId **to** |B| *in parallel* **do**

6: SM[i, j] ← score of aligning row i in DM_A against row j in DM_B

7: **end for**

8: **end for**

9: **end kernel**

II. Second Parallel Alignment Step

In this part, another dynamic programming algorithm is applied on the score matrix SM and then traced back by a recursive algorithm to generate the initially aligned pairs. Since alignment path is needed, then not only the first two rows of similarity matrix F are needed but also the whole matrix rows have to be allocated. Consequently, from the memory view, only one global similarity matrix F is represented as a one-dimension vector of type double with size [(|A|+1) * (|B|+1)]. In order to decrease the number of accesses to the GPU global memory, the similarity matrix F is not calculated cell by cell, it is divided into diagonals. From the data dependency view, each element F[i,j] depends on three elements, F[i,j-1], F[i-1,j], and F[i-1,j-1]. In other word, F[i,j] depends on the data from both same and previous rows. This kind of dependency looks like a diagonal scan over the elements. This technique is known as wave-front technique [52].

In fact, algorithms which are using wave-front techniques are usually developed by calling two nested loops where the outer loop represents matrix diagonals, and the inner loop represents the cells of each diagonal. This technique can be parallelized by implementing the inner loop as a parallel for loop. This means that the computations in all cells in each diagonal run in parallel where diagonals itself are running in sequence. So, in terms of data dependency, each matrix diagonal depends on the previous one. The parallel implementation of the wave-front technique is shown in Fig 2.

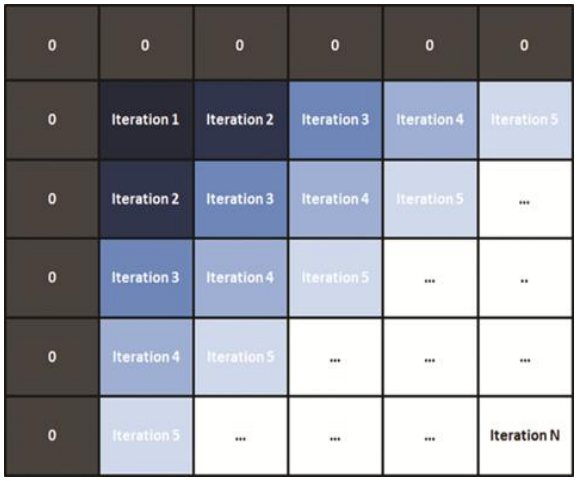

Fig 2: The Parallel Representation of Wave-Front Technique

The above-mentioned parallel methodology has some drawbacks such as the heavy load resulting from repetitive CPU-GPU data transfer process. In order to overcome this weakness, the CUDA dynamic parallel model is used to

parallelize this level. In the CUDA dynamic parallel model, GPU kernel can launch an inline nested kernel to eliminate the data transfer load from CPU to GPU and vice versa. Therefore, this model is followed as illustrated in Algorithm 4.

In Algorithm 4, the computation is split into two GPU kernels where the first kernel (parent) is responsible for calling diagonals in sequence and figuring out the number of blocks needed to parallelize each diagonal. Not all diagonals need the same number of blocks to run in parallel. Accordingly, the total number of blocks B_t is determined by the number of cells in each diagonal C_d where $B_t = C_d$. Afterward, this parent kernel calls a child kernel to calculate the value of each cell using dynamic programming. Once this calculation is done, the similarity matrix is moved from GPU memory to CPU memory to run a recursive algorithm to generate the initial alignment pairs. After that, experiments show that recursive refinement level is much faster on the CPU than the GPU since there is data dependency between each refinement step.

Algorithm 4 GPU parallel implementation of the second alignment step

1: **kernel** ParentKernel(SM, F)

2: Let P = max number of cells in all diagonals

3: Let R = number of repeats of diagonals with max number of cells

4: **for** i=1 **to** P **do**

5: ChildKernel<<<i, 1>>>(SM, F)

6: **end for**

7: **for** i=1 **to** R **do**

8: ChildKernel<<<P, 1>>>(SM, F)

9: **end for**

10: **for** i=1 **to** P **do**

11: ChildKernel<<<$P - i$, 1>>>(SM, F)

12: **end for**

13: **end kernel**

1: **kernel** ChildKernel(SM, F)

2: calculate current cell indices i and j using threadId and blockId

3: $F[i,j] \leftarrow Max(\, F[i-1,j] + GS, F[i-1,j-1] + SM[i,j], F[i-1,j] + GS)$

4: **end kernel**

3. RESULTS

The effectiveness of the PTM-MatAlign algorithm is tested on MSI GE60 2PE ApachePro with Intel core i7-4710HQ 2.50 GHz processor, 8GB of RAM. This computation environment had the Microsoft Windows 8 64-bit operating system installed, with CUDA SDK version 6.5 with compute capability 5 settled on GPU device GeForce GTX 860M (5 streaming multiprocessors with 640 processing cores, 49 KB of shared memory per block, 65 KB of total constant memory, 65536 registers per block, 2GB of total global memory).

Two types of experiments were conducted to assess the performance of the PTM-MatAlign algorithm. Firstly, it is important to measure the quality of the alignment results against its CPU-based implementation MatAlign and the most often-used structural alignment algorithms, such as TM-Align, MICAN, FATCAT, and CE using different criteria. These methods were chosen to compare with as a kind of diversity. For example, TM-Align and MICAN are using TM-Score as a scoring function. FATCAT is interested in flexible and rigid alignment. And CE is one of the classic alignment methods which is considered to be a reference. Secondly, it is important to compare the speed of PTM-MatAlign against all these algorithms. These two experiments were conducted using three diversified benchmarked datasets [53], [54], and [5]. These dataset were selected to represent different classes according to the SCOP classification [55], such as., all alpha proteins (all α), all beta proteins (all β), alpha and beta proteins (α/β), alpha and beta proteins (α+β), multi-domain proteins (alpha and beta), membrane and cell surface proteins and peptides, coiled coil proteins, and small proteins.

3.1 Experiment 1: alignment quality assessment

This experiment assesses the alignment quality of PTM-MatAlign in relation with the most often-used CPU-based structural alignment algorithms mentioned previously. Three criteria were defined to assess the alignment quality, RMSD, TM-Score, and Alignment Length

3.1.1 RMSD

RMSD is one of the most used measures between a pair of structures with a specified set of equivalent residues. As shown in Fig 3 it can be observed that PTM-MatAlign generally tends to produce a smaller RMSD value than MatAlign, TM-Align, MICAN, FATCAT and CE do over the three datasets. Since there is a consensus says that the smaller RMSD value is, the better alignment will be. It means that PTM-MatAlign achieves better-fitted alignment than those alignment algorithms do.

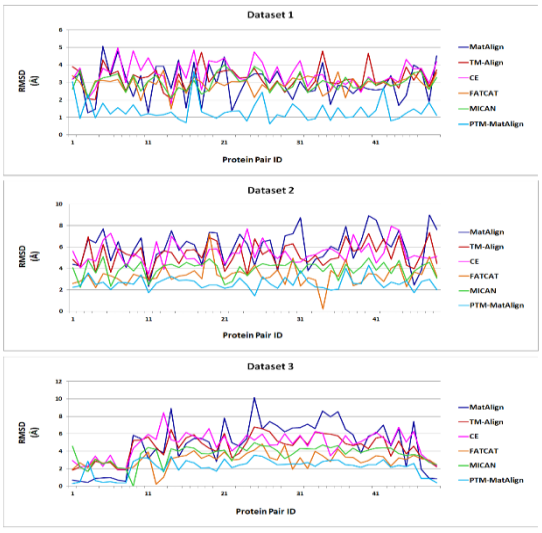

Fig 3: Distribution of RMSD values over the three benchmarked datasets (lower values mean better alignment).

3.1.2 TM-Score

A good alignment measure must consider the number of atoms in the alignment. This makes RMSD not a valid measure by itself as previously explained in Section 2.3.1. The TM-Score that was proposed by [5] overcomes RMSD

weaknesses. It was weighted according to the size of the compared structures in such a way that it varies between 0 and 1, from a bad to a good alignment. As shown in Fig 4, it can be observed that PTM-MatAlign produces highest TM-Score, over the three used datasets, which means it produces best alignment results than MatAlign, TM-Align, MICAN, FATCAT and CE do

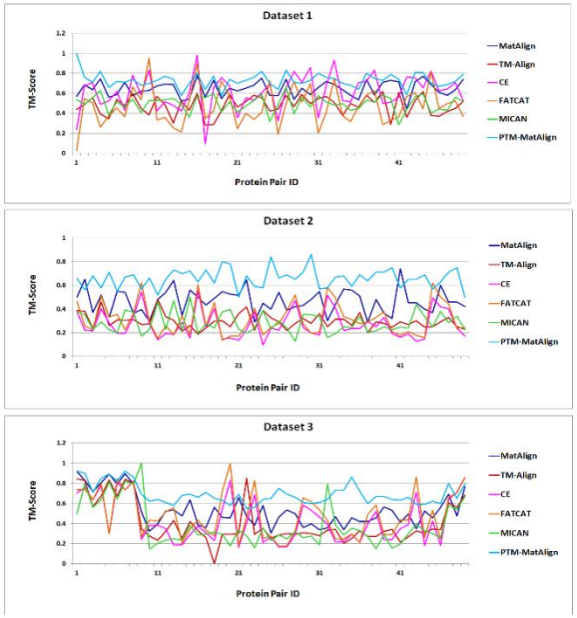

Fig 4: Distribution of TM-Score values over the three benchmarked datasets (higher values mean better alignment).

3.1.3 Alignment Length

It is not enough to produce low RMSD value to mark the alignment results as good. It has to output a large number of atoms in the alignment with a high TM-Score value. In Fig 5, it is observed that PTM-MatAlign produces longer number of atoms in the alignment, over the three used datasets, than the MatAlign, TM-Align, MICAN, FATCAT and CE do. The summary of structural alignments assessment is shown in Table 1.

Table 1. Table captions should be placed above the table

	Data Set 1			Data Set 2		
	RMSD (A)	**TM-Score**	**Alignment Length**	**RMSD (A)**	**TM-Score**	**Alignment Length**
MatAlign	2.967	0.645	53	6	0.467	136
TM-Align	3.231	0.48	58	5.338	0.296	84
CE	3.458	0.611	62	5.457	0.271	70
FATCAT	2.908	0.471	62	3.361	0.136	85
MICAN	3.003	0.499	60	3.963	0.286	74
PTM-MatAlign	**1.37**	**0.73**	**71**	**2.63**	**0.65**	**164**
MatAlign	4			7	9	

	Data Set 3			Average over all Datasets		
	RMSD (A)	**TM-Score**	**Alignment Length**	**RMSD (A)**	**TM-Score**	**Alignment Length**
MatAlign	4.816	0.531	117	4.594	0.548	102
TM-Align	4.481	0.41	91	4.35	0.395	78
CE	4.808	0.428	74	4.574	0.437	69
FATCAT	3.041	0.483	79	3.103	0.361	75
MICAN	3.668	0.386	79	3.549	0.548	71
PTM-MatAlign	**2.177**	**0.69**	**140**	**2.0627**	**0.693**	**125**

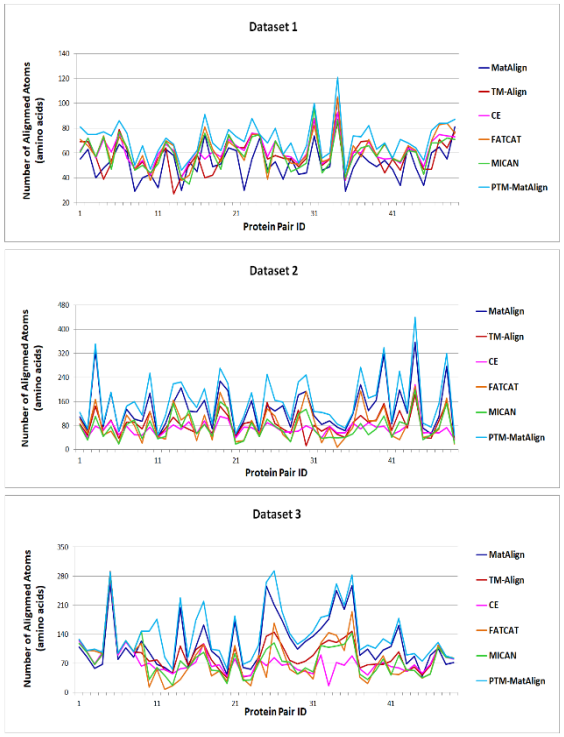

Fig 5: Distribution of alignment length over the three benchmarked datasets (higher values mean better alignment).

To illustrate the alignment accuracy visually, a classic structural alignment between 2FB4_H and 2HPD_A is executed which have a sequence identity of 30%. As shown in Fig 6, MatAlign aligns the two proteins with overall RMSD 17.775 Å with 204 aligned residues and TM-Score 0.4773. TM-Align aligns fewer residues, which results in a lower RMSD of 6.48 Å over 112 aligned residues and TM-Score 0.25506. Likewise, MICAN aligns few numbers of residues with low RMSD of 4.271 Å over 77 aligned residues and TM-

Score 0.238. FATCAT aligns the two structures with overall RMSD 3.16 Å over 32 aligned residues and TM-Score 0.0657. CE aligns the two structures with overall RMSD 5.32 Å over 57 aligned residues and TM-Score 0.0821. PTM-Align has the longest alignment coverage with overall RMSD of 4.16 Å on 228 aligned residues with the highest TM-Score 0.68. According to this alignment analysis, only PTM-MatAlign can achieve the most accurate alignment results compared with results produced by those algorithms.

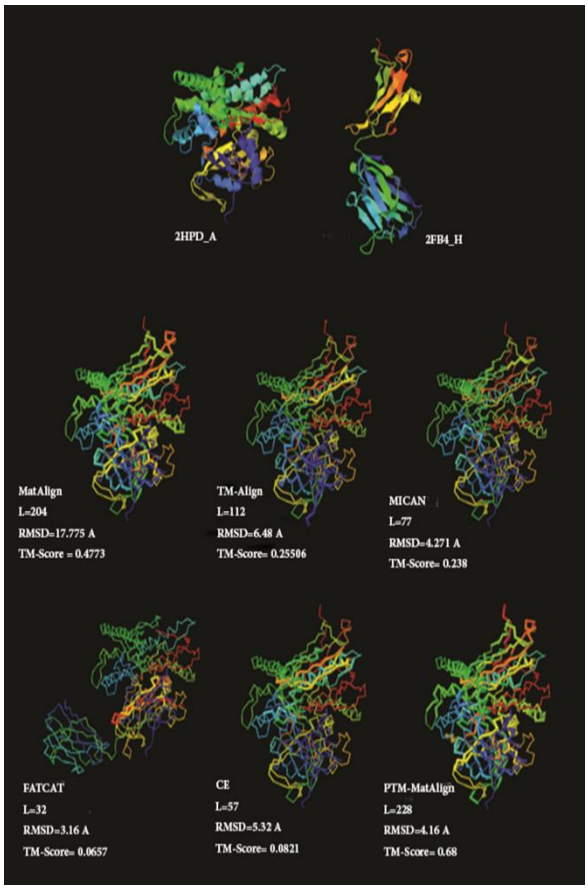

Fig 6: Visual illustrative of aligning 2hpd_a (457 residues) and 2fb4_h (229 residues) using different alignment methods.

3.2 Experiment 2: alignment speed assessment

In terms of speed, PTM-MatAlign attains better results over the three benchmarked datasets. For instance, as an average speed up over the three benchmarked datasets, it is about 11 times faster than the original single-threaded sequential algorithm MatAlign, 2 times faster than TM-Align, 8 times faster than CE, 13 times faster than FATCAT and about 28 times faster than MICAN on and Intel Core i7-4710HQ (2.50GHz, 8GB RAM, 8 cores) CUP. Fig 7 shows the execution times of the six alignment methods over the three benchmarked data sets used. Also, as shown in Fig 8,there is such a correlation between query protein length and execution time, where the greater the length of the query proteins are, the higher the execution time becomes. This is expected because most of these algorithms mainly depend on query protein length to build their methodology. The speedup of PTM-MatAlign over the comparable alignment algorithms is shown in Fig 9.

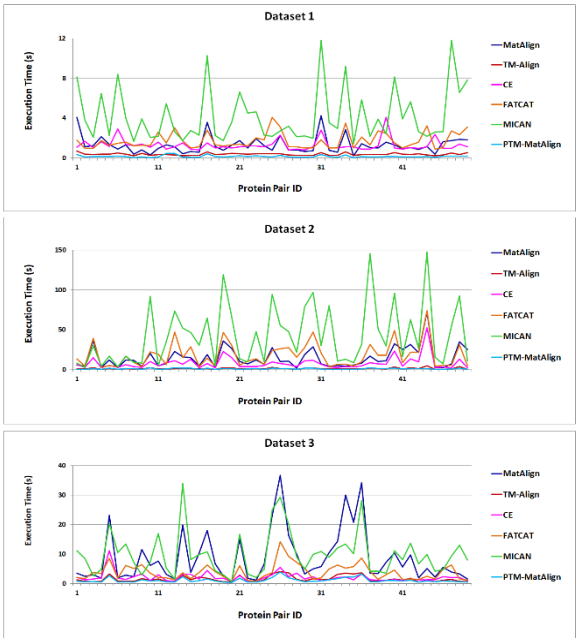

Fig 7: Distribution of alignment execution time in seconds over the three benchmarked datasets.

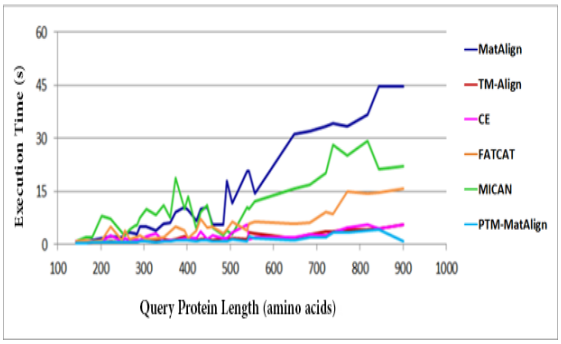

Fig 8: Distribution of alignment execution time in relation with query protein length.

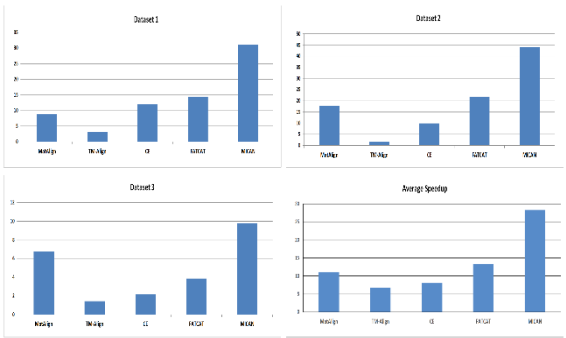

Fig 9: The speedup values for PTM-MatAlign over the other comparable alignment algorithms for the three benchmarked datasets.

4. DISCUSSION

The experiments that are performed have proven the proposed hypothesis that using GPU is more computationally efficient than using the old classic single-core techniques to perform protein structure comparison is the use of CUDA parallel architecture using a graphical processing unit. Upon comparing the execution times, GPU-based implementation, PTM-MatAlign, is much faster than the other comparable

CPU-based alignment algorithms mentioned in the results section. This is vital ever since the number of protein structures in biological databases is increasing exponential in time. For such process, the use of GPU-based implementations is mostly important because GPU devices are quite inexpensive compared to clusters. For example, a middle-class GPU device that is settled on small computation environment was used with one processor and achieved better alignment results. Therefore, GPU devices can be usefully used in implementing many bioinformatics algorithms.

The GPU-based implementation, PTM-MatAlign, consumes the fewest execution time among other algorithms because not only using the CUDA parallel architecture to implement PTM-MatAlign but also using the CUDA dynamic parallelism technique. Using the CUDA dynamic parallel model in the second alignment step, an explicit GPU kernel (Parent) was designed first. This kernel takes the responsibility for determining the number of blocks needed to parallelize cells in each diagonal in the similarity matrix. In fact, this kernel has one thread to manage the loop because the main motivation for implementing such kernel is to move these computations from CPU to GPU. Thus, it is quite easy to get rid of the kernel overhead problem and exploit the massive power of dynamic parallelism provided by CUDA.

According to alignment quality assessment results, PTM-MatAlign achieves the best alignment with refereeing to the different alignment quality measurements, RMSD, TM-Score and alignment length. Moreover, using TM-Score instead of MatAlign regular score helps in refining the alignment results in such an excellent way to produces the best alignment. According to the TM-Score value, the algorithm determines whether the alignment length will be affected or not. As a result, the proposed parallel implementation, PTM-MatAlign, produces differently aligned pair list than the original MatAlign does. As per RMSD is a function of the length of aligned pair list, then RMSD value in PTM-MatAlign will also differ from its counterpart in MatAlign. Similarly, it is worth mentioning that PTM-MatAlign produces different alignment length and different TM-Score than TM-Align and MICAN does because of the different alignment technique used.

However, comparing large size proteins using this GPU-based implementation on the proposed GPU environment is deemed controversial because the alignment and score matrices cannot be stored in the GPU global memory during computations. It is not a software-related issue because the upper limit on the size of the query proteins is determined based on the GPU memory. The larger the GPU memory is, the bigger the protein files that can be aligned. As believed, soon, the technological advances in the GPUs industry will be able to find out a solution for such problems.

5. CONCLUSION

This paper presents the PTM-MatAlign algorithm which is a GPU-based parallel algorithm for pairwise protein 3D structures alignment. It parallelizes the MatAlign algorithm with using a TM-score function to refine the alignment results. Results show that the PTM-MatAlign algorithm overcomes the speed of its CPU-based implementation by nearly 11 times. Also, it overcomes the speed of TM-Align by nearly 2 times, MICAN by 28 times, CE by 8 times and FATCAT by 13 times.

The advantages of the PTM-MatAlign algorithm are two-fold: First, the detailed comparative analysis of protein structures. Second, the avoidance of additional overhead resulted from launch configurations of kernels by using the CUDA dynamic parallel model. On the other hand, PTM-MatAlign drawback is the large GPU memory space needed due to the requirement of having the whole data when mapping proteins as 2D distance matrices. So that, the PTM-MatAlign algorithm with the proposed computational environment is currently working on middle-size query proteins with length near 900 amino acids

6. REFERENCES

[1] Morange, M. 1999. A History of Molecular Biology. BioScience, 49(11), 929-931.

[2] Burkowski, F.J. 2009. Structural bioinformatics: an algorithmic approach (Vol. 20): Chapman & Hall/CRC.

[3] Liisa , H., and S. Chris. 1993. Protein Structure Comparison by Alignment of Distance Matrices. Journal of Molecular Biology, 233(11), 123–138.

[4] Orengo, C.A., and W.R. Taylor. 1996. SSAP: sequential structure alignment program for protein structure comparison. Methods Enzymol, 266, 617-635.

[5] Zhang, Y., and J. Skolnick. 2005. TM-align: a protein structure alignment algorithm based on the TM-score. Nucleic Acids Research, 33(7), 2302-2309.

[6] Jianhua, Z., and W. Zhiping. 2004. FAST: A novel protein structure alignment algorithm. Proteins: Structure, Function, and Bioinformatics, 58(3), 618-627.

[7] Zeyar, A., and T. Kian-Lee. 2006. MatAlign: PRECISE PROTEIN STRUCTURE COMPARISON BY MATRIX ALIGNMENT. Journal of Bioinformatics and Computational Biology, 4(6), 1197-1216.

[8] Nicolas, B., and M. Pierre-Francois. 2012. LNA: Fast Protein Structural Comparison Using a Laplacian Characterization of Tertiary Structure. IEEE/ACM Transactions on Computational Biology and Bioinformatics, 9(5), 1451-1458.

[9] Inken , W., M.-D. Noel , and A. Rumen 2012. CSA: comprehensive comparison of pairwise protein structure alignments. Nucleic Acids Research, 303-309.

[10] Genki, T., and T.-S. Mayuko. 2015. CAB-Align: A Flexible Protein Structure Alignment Method Based on the Residue-Residue Contact Area. PLoS ONE, 10(10).

[11] Jean-Francois, G., M. Thomas, and H.B. Stephen. 1996. Surprising similarities in structure comparison. Current Opinion in Structural Biology, 6(3), 377-385.

[12] Shindyalov, I.N., and P.E. Bourne. 1998. Protein structure alignment by incremental combinatorial extension (CE) of the optimal path. Protein Eng, 11(9), 739-747.

[13] Yuzhen, Y., and G. Adam. 2003. Flexible structure alignment by chaining aligned fragment pairs allowing twists. Bioinformatics, 19(2), 246-255.

[14] Maxim, S., N. Ruth, and J.W. Haim. 2004. FlexProt: Alignment of Flexible Protein Structures Without a Predefinition of Hinge Regions. Journal of Computational Biology, 11(1), 83-106.

[15] Mathilde, C., B. Sophie, and P. Joël. 2005. YAKUSA: A fast structural database scanning method. Proteins: Structure, Function, and Bioinformatics, 61, 137-151.

[16] Matthias, H., and F. Dmitrij. 2004. STRIDE: a web server for secondary structure assignment from known atomic coordinates of proteins. Nucleic Acids Research, 32, 500-502.

[17] Zheng, W. 2008. CLePAPS: Fast Pair Alignment of Protein Structures Based on Conformational Letters. Journal of Bioinformatics and Computational Biology, 6(2), 347-366.

[18] Potestio, R., et al. 2010. ALADYN: a web server for aligning proteins by matching their large-scale motion. Nucleic Acids Research, 38(2), 41-45.

[19] Paweł, D., and L. Bogdan. 2011. A novel method to compare protein structures using local descriptors. BMC Bioinformatics, 12(344).

[20] Shintaro, M., S. Kengo, and C. George. 2013. MICAN: a protein structure alignment algorithm that can handle multiple-chains, inverse alignments, Ca only models, alternative alignments, and non-sequential alignments. BMC Bioinformatics, 14(24).

[21] Silvia, C., et al. 2013. ASSIST: a fast versatile local structural comparison tool. Bioinformatics, 30(7).

[22] Noel, M.-D., and P. Natasa. 2014. GR-Align: fast and flexible alignment of protein 3D structures using graphlet degree similarity. Bioinformatics, 30(9), 1259-1265.

[23] Shanshan, C., Z. Yang, and B. Charles. 2015. PCalign: a method to quantify physicochemical similarity of protein-protein interfaces. BMC Bioinformatics, 16(33).

[24] Wohlers, I., F.S. Domingues, and W.K. Gunnar 2010. Towards optimal alignment of protein structure distance matrices. Bioinformatics, 26, 2273-2280.

[25] Linial, K. 20004. Approximate Protein Structural Alignment in Polynomial Time. Proceedings of the National Academy of Sciences of the United States of America(101), 12201–12206.

[26] Peter, A.J., P.R. Andrew, and G.M. Helen. 1994. A Graph-theoretic Approach to the Identification of Three-dimensional Patterns of Amino Acid Side-chains in Protein Structures. Journal of Molecular Biology, 243(2), 327–344.

[27] Andrew , W.C., B. Neera , and T. Janet 1997. TESS: a geometric hashing algorithm for deriving 3D coordinate templates for searching structural databases. Application to enzyme active sites. Protein Science, 6(11), 2308-2323.

[28] Nathaniel, L., N. Ruth , and W.J. Haim 2001. MUSTA - A General, Efficient, Automated Method for Multiple Structure Alignment and Detection of Common Motifs: Application to Proteins. Journal of Computational Biology, 8(2), 93-121.

[29] Jonathan, B.A., and T.M. Janet 2003. An algorithm for constraint-based structural template matching: application to 3D templates with statistical analysis. Bioinformatics, 19(13), 1644-1649.

[30] Pramod , W.P., et al. 2003. Functional Sites in Protein Families Uncovered via an Objective and Automated Graph Theoretic Approach. Journal of Molecular Biology, 326(3), 955–978.

[31] Alexander , S., and R.B. Robert. 2003. Annotation in three dimensions. PINTS: Patterns in Non-homologous Tertiary Structures. Nucleic Acids Research, 31(13), 3341-3344.

[32] Nurcan, T., et al. 2011. Predicting protein-protein interactions on a proteome scale by matching evolutionary and structural similarities at interfaces using PRISM. Nat. Protocols, 6(9), 1341-1354.

[33] Kundrotas, P.J., Z. Zhengwei, and J. Joël. 2012. Templates are available to model nearly all complexes of structurally characterized proteins. Proceedings of the National Academy of Sciences of the United States of America, 109(24), 9438-9441.

[34] Kundrotas, P.J., and I.A. Vakser. 2013. Global and local structural similarity in protein–protein complexes: Implications for template-based docking. Proteins, 81(12), 2137-2142.

[35] Bagley, S.C., and R.B. Altman. 1995. Characterizing the microenvironment surrounding protein sites. Protein Science, 4(4), 622-635.

[36] Binkowski, A.T., L. Adamian, and J. Liang. 2003. Inferring Functional Relationships of Proteins from Local Sequence and Spatial Surface Patterns. Journal of Molecular Biology, 332(2), 505–526.

[37] Glaser, F., et al. 2006. A method for localizing ligand binding pockets in protein structures. Proteins: Structure, Function, and Bioinformatics, 62(2), 479–488.

[38] Xie, L., and P.E. Bourne. 2008. Detecting evolutionary relationships across existing fold space, using sequence order-independent profile–profile alignments. Proceedings of the National Academy of Sciences of the United States of America, 105(14), 5441-5446.

[39] La, D., et al. 2009. 3D-SURFER: software for high-throughput protein surface comparison and analysis. Bioinformatics, 25(21), 2843-2844.

[40] Mernberger, M., G. Klebe, and E. Hullermeier. 2011. SEGA: Semiglobal Graph Alignment for Structure-Based Protein Comparison. IEEE/ACM Transactions on Computational Biology and Bioinformatics, 8(5), 1330-1343.

[41] Hu, B., et al. 2014. PL-PatchSurfer: A Novel Molecular Local Surface-Based Method for Exploring Protein-Ligand Interactions. International Journal of Molecular Sciences, 15(9), 15122-15145.

[42] Karampudi, N.B.R., and R.P. Bahadur. 2015. Layers: A molecular surface peeling algorithm and its applications to analyze protein structures. Scientific Reports, 5.

[43] Alex , S.D., P.J. Stuckey, and A.I. Wirth. 2010. Fast and accurate protein substructure searching with simulated annealing and GPUs. BMC Bioinformatics, 11, 446.

[44] Bin, P., et al. 2012. Accelerating large-scale protein structure alignments with graphics processing units. BMC Research Notes, 5(1), 116.

[45] Mrozek, D., M. Brożek, and B. Małysiak-Mrozek. 2014. Parallel implementation of 3D protein structure similarity searches using a GPU and the CUDA. Journal of Molecular Biology, 20(2), 2067.

[46] Saul, N.B., and W.D. Christian. 1970. A general method applicable to the search for similarities in the amino acid

sequence of two proteins. Journal of Molecular Biology, 48(3), 443-453.

[47] Rachel, K., K. Patrice, and L. Michael. 2005. Comprehensive Evaluation of Protein Structure Alignment Methods: Scoring by Geometric Measures. Journal of Molecular Biology, 346, 1173-1188.

[48] Alexandrov, N.N., and D. Fischer. 1996. Analysis of topological and nontopological structural similarities in the PDB: new examples with old structures. Proteins, 25(3), 354-365.

[49] Kabsch, W. 1978. A discussion of the solution for the best rotation to relate two sets of vectors. Acta Crystallographica, 34, 827-828.

[50] Skolnick, J. 2004. Scoring function for automated assessment of protein structure template quality. Proteins, 57, 702-710.

[51] Michael, L., and G. Mark. 1998. A unified statistical framework for sequence comparison and structure comparison. Proceedings of the National Academy of Sciences of the United States of America, 95(11), 5913-5920.

[52] Dhraief, A., R. Issaoui, and A. Belghith. 2011. Parallel Computing the Longest Common Subsequence (LCS) on GPUs: Efficiency and Language Suitability. INFOCOMP.

[53] Fischer, D., et al. 1996. Assessing the performance of fold recognition methods by means of a comprehensive benchmark. Pacific Symposium on Biocomputing, 300-318.

[54] Cheng, H., B.H. Kim, and N.V. Grishin. 2008. MALISAM: A Database of Structurally Analogous Motifs in Proteins. Nucleic Acids Research, 36, 211-217.

[55] Murzin, A.G., S.E. Brenner, and T. Hubbard. 1995. SCOP: a structural classification of proteins database for the investigation of sequences and structures. J Mol Biol(247), 536–540.