ISSN NO. 2320-5407

Journal homepage:http://www.journalijar.com

INTERNATIONAL JOURNAL OF ADVANCED RESEARCH

RESEARCH ARTICLE

Accelerated Guide Trees Construction for Multiple Sequence Alignment

Mohammed W. Al-Neama*, Naglaa M. Reda,Fayed F. M. Ghaleb

1.Education College for GirlsMosul University, Mosul, Iraq \Department of Mathematics, Faculty of Science, Al Azhar University Cairo, Egypt

2.Department of MathematicsFaculty of ScienceAin Shams University Cairo, Egypt

Manuscript Info

Manuscript History:

Received: 25 February 2014
Final Accepted: 22 March 2014
Published Online: April 2014

Key words:
Bioinformatics**,** Multiple Sequence Alignment, Guide Trees, Neighbour Joining, Parallel Computing, Multi-core.

Corresponding Author

Mohammed W. Al-Neama*

Abstract

Guide trees construction is an essential step in progressive Multiple Sequence Alignment (MSA). The most widely used MSA methods apply Neighbour Joining (NJ) algorithm.NJ requires intensive computationsand exhausts huge storage space proportional to number of sequences. The growing of biological sequences forms a significant barrier and necessitates fast and optimized computation method. Some algorithms that reduce time and space requirements were produced, but at the expense of alignment accuracy. This paper introduces a new massively parallel optimized algorithm based on converting used matrices in NJ into vectors and eliminating redundant computations, while preserving the accuracy. Complexity analysis indicates significant reductions in both time and space. Proposed algorithm have been implemented on a 2.0 GHz core i7 Intel CPU in C++ and tested on various real protein sequences. Results show how the proposed vectorization approach greatly improves the performance and achieves more than 2.5-fold speedup when aligning 8000 sequences compared to ClustalW- MPI.

I. INTRODUCTION

Multiple Sequence Alignment (MSA) is an important tool in bioinformatics. It can identify patterns or motifs to characterize protein families, and is therefore utilized to detect homology between sequences as well as to perform phylogenetic analysis. Many MSA tools have been proposed, such as MSA [1], ClustalW [2], T-Coffee [3], MAFFT [4], DIALIGN [5]and PRALINE [6]. Most of them are progressive alignment-based methods. They typically consist of three stages as clarified at Fig.(1) [7]. Stage 1 computes a Distance Matrix (D) comprised of the distance value between each pair of input sequences. Stage 2 computes a guide tree from D using some phylogeny reconstruction method. Stage 3 aligns first closely related sequences at the guide tree, and then aligns the most divergent sequences to get the final MSA.

The task of phylogeny is to derive the previous tree based on observations of the existing organisms [8]. It indicates relations between sequences, since organisms in whole world have a common predecessor. The Neighbour Joining (NJ) method [9] is widely used among the greedy approaches for constructing the guide tree because it is applicable to any type of evolutionary distance data. Its main task is to compute the branch lengths of the guide tree from D. Leaf nodes in such trees correspond to observable taxa while internal nodes represent hypothetical ancestors of the sequences. It works perfectly with small taxa sets.

However, the exponential growth of biological data demands a higher throughput. Real databases now contain enough representatives of larger protein families that exceed the capacity of most current programs. Thus, there has been a plethora of new solutions that attempt to solve this problem using caching [10,11] and parallel processing [12,13]. Also, easily accessible accelerator technologies have been used including FPGAs [14], GPUs [15] and

CUDA [16]. These heuristic approaches generally reduce the search space and make the comparison of large genomic banks faster, but at the expense of a non-negligible reduction of algorithmic accuracy.

In this work, authors face the challenge of handling huge data by using two main approaches. The first is the vectorization, where all matrices are compensated by vectors, which in turn reduces the memory requirement and speedup execution without affecting the accuracy. The second approach is parallelism, the widespread programming method nowadays that allows multiple independent processes which share the same resources, to be executed concurrently at less time. The multi-core was the candidate platform for implementation due to its availability, commonly use and almost has no communication overhead [18].

This paper proposes an accelerated algorithm for the guide tree construction. It mathematically proves that the used matrices can be converted into vectors. It also demonstrates that there are some unnecessary calculations and produces new formulas skipping repetitions. Then it introduces how computations are parallelised for multi-core use. Finally, it verifies experimentally that the proposed method is superlative over existing ones in both space and execution time without losing the accuracy.

Implementations were conducted on 2.0 GHz core i7 Intel using OpenMP that supports shared memory multi-processing programming in C++. It uses fork-join model. All OpenMP programs begin the master thread. It executes sequentially until the first parallel region construct is encountered. Then a team of parallel threads are created. The statements in the program are then executed in parallel among the various team threads. When the team complete, they synchronize and terminate, leaving the master thread [18].

The rest of this paper is organized as follows. First, related work is surveyed in section 2. Section 3 explains the sequential neighbour joining algorithm NJ. Section 4 introduces the proposed vectorized fine-grain parallel algorithm, and studies the deduction of new proposed formulas. Experimentalresults and discusses are shown in section 5. Finally, the conclusion is given.

II. RELATED WORK

The neighbour joining method is the widely used method for phylogenetic reconstruction in most popular MSA tools such as ClustalW and T-Coffee. It initially introduced by Saitou and Nei [9], and achieves computations efficiently combined with reasonable accuracy. Studier and Kepler [25] incorporated improvements that reduce its temporal complexity from $O(N^5)$ to $O(N^3)$, where N is the number of OTUs (Operational Taxonomic Units). Some attempts have been presented to improve the performance, such as QuickTree [19] and Relaxed Neighbor-Joining (RNJ) [20], but they can't reduce complexity.

Because handling large datasets could take days, and occupies more than 30% of the alignment runtime, different parallel algorithms have been developed. Du and Feng [21] proposed pNJTree parallel method for the NJ tree reconstruction using MPI running on a workstation cluster. Bullard [22] and Li [12] used the coarse-grained parallelism. They parallelized the searching of sequences having the highest divergence using MPI implementation.

Liu et al. [16] introduced a parallel algorithm that computes the two innermost loops of NJ in parallel using threads on GPU. In [23], they developed an efficient mapping onto the GPU architecture and parallelized NJ tree reconstruction using CUDA. Sahoo et al. [24] proposed an algorithm for shared memory that parallelizes computation using multiple threads.

III. NEIGHBOUR JOINING ALGORITHM

In this section, we explain in details the NJ algorithm of Studier and Kepler [25], as a base to our work. Consider aligning a dataset $\{S_1, S_2,..., S_N\}$ of size N. MSA first stage produces a symmetric diagonal matrix D of dimension NxN, where the value D(i,j)denotes the similarity distance between $S_i$andS_j. NJ takes D as input and produces the guide tree by applying the following steps:

1. First, it begins with a star graph, then iteratively finds the closest neighboring pair,
2. Second, it constructs a tree by clustering the closest neighboring pair to each other and connecting them through an internal node,
3. Then, it connects all the remaining sequences to nodein a star-like structure and computes the distances of them to the internal node,
4. Finally, it terminates when all internal nodes are inserted into the tree constituting the final binary tree.

The main task of NJ method is to compute the branch lengths of the guide tree from the pairwise distances. This task is accomplished using the following estimated formulas.

- The sum of the least-squares estimates of branch lengths (Q_{ij}), for i=1…N; j=i+1… N; k,l ≠ i,j;is given by:

$$Q_{ij} = \frac{1}{2(N-2)} \sum_{k=1}^{N} (D_{ik} + D_{jk}) + \frac{1}{2} D_{ij} + \frac{1}{(N-2)} \sum_{k<l}^{N} D_{kl} \quad \text{.....(1)}$$

- The neighbours to be joined in the treei_{min} and j_{min} of OTUs satisfying:

i_{min}= i , j_{min}= j that have the smallest sum of branch lengths

- The distance between combined OUTs (U_1) and another OTU k, for all k = 1…N; k ≠i, j,is given by:

$$D_{U_1 k} = \frac{(D_{ik} + D_{jk})}{2} \quad \text{.....(2)}$$

- The branch lengths L_{iU_1} and L_{jU_1} of OTUs neighbors, indexed by i and j, is estimated by [26]:

$$L_{iU_1} = (D_{ij} + D_{iz} - D_{jz})/2$$

$$L_{jU_1} = (D_{ij} + D_{jz} - D_{iz})/2 \quad \text{.....(3)}$$

where $D_{iz} = \frac{\sum_{k=1}^{N} D_{ik}}{(N-2)}$ and $D_{jz} = \frac{\sum_{k=1}^{N} D_{jk}}{(N-2)}, k \neq i, j$ andz represents all the OTUs except i and j.

The detailed algorithm of the sequential NJ algorithm is as follows [27]:

Input : N (the number of OTUs) and D (Distance Matrix)
Output : NJ tree

1. forh in 1 to N - 2 do
1.1 fori=0 to N-1 do
1.2 forj=i+1 to N do
1.2.1 Compute Q_{ij} according to formula (l).
1.2.2 end for
1.2.3 end for
1.3 Search the minimum Q_{ij} to get neighboursi and j.
1.4 OTUs i and j are joined as a new OTU U_h.
1.5 Compute D_{Uhk} according to formula (2).
1.6 Compute branch lengths L_{iUh} and L_{jUh} according to formula (3).
1.7 Delete OTUs i and j, and add U_h to current OTU lists.
1.9 end for
2 Join the last two OTUs.

Fig. (2)illustrates the constructed NJ tree for a given 6 OTUs, labeled A, B, C, D, E and F, which indicates their evolutionary relationships. The numbers above the lines are the computed branch lengths of each two neighbours. It starts with connecting the pair (A, B) by an internal node, because they are the closest, generating a new combined OUT (U_1). Thus, the branch lengths of the U_1 to A and B are 2 and 5, respectively. This leads to connecting C to the new OUT as ((A, B), C). The same action happens with ((D, E), F). Finally, it produces the resulted binary tree by linking the only remaining two OTUs.

IV. PROPOSED ALGORITHM

According to the above formulas, the most time consuming step is Eq.(1). Thus, the key point of optimizing NJ algorithm is to optimize the computation of Q. In fact, we observe that there are many repeated computations on the same data. For example, D_{13} is added when computing Q_{12} and Q_{14}. Also, $D_{34}$is added when computing Q_{12} and Q_{15}. In other words, there are common calculations between the computations of Q_{ij}.On the other hand, nested loops are very good candidate for parallelism. Therefore, these observations motivate us to design an optimized fast parallel algorithm of NJ. In the following, a detailed explanation of how we eliminate repeated computations and vectorize unnecessary used matrices is presented. In addition, our proposed parallel algorithm is introduced.

A. Optimization Approach

To eliminate redundant calculations during computing Q elements, we have reformulated Eq.(1) by the following derivation.

Let R(N) and total be two buffers, the former is a vector used to store the sum of every row of D_{ij} and the latter is a variable to store the sum of the whole distance matrix. They are evaluated by these equations:

$$R_i = \sum_{k=1;k\neq i}^{N} D_{ik} \quad \text{.....(4)}$$

$$total = \sum_{i=1}^{N} R_i \quad \text{.....(5)}$$

Since D_{ij} is a symmetric square matrix and Q_{ij} is a triangular matrix, then the first and third part of Eq. (1) that computes Q_{ii} can be reformulated as follows:

$$\begin{aligned} R_i - D_{ij} &= \sum_{k=1;k\neq i}^{N} D_{ik} - D_{ij} \\ &= \sum_{k=1;k\neq i,j}^{N} D_{ik} \\ R_j - D_{ij} &= \sum_{k=1;k\neq j}^{N} D_{jk} - D_{ij} \\ &= \sum_{k=1;k\neq i,j}^{N} D_{jk} \end{aligned}$$

By adding we get:

$$\begin{aligned} (R_i - D_{ij}) + (R_j - D_{ij}) &= R_i + R_j - 2D_{ij} \\ &= \sum_{k=1;k\neq i,j}^{N} (D_{ik} + D_{jk}) \end{aligned}$$

Hence,

$$\frac{1}{2(N-2)} \sum_{k=1;k\neq i,j}^{N} (D_{ik} + D_{jk}) = \frac{(R_i + R_j - 2D_{ij})}{2(N-2)} \quad \text{.....(6)}$$

On the other hand, (R_i + R_j) denotes the sums of rows and columns of i and j respectively and (total - 2(R_i + R_j) + 2D_{ij}) denotes the sums of the remaining elements of (D_{ij}) except those elements related to i and j. This leads to:

$$total - 2(R_i + R_j) + 2D_{ij} = \sum_{k<l}^{N} (D_{ik} + D_{jk}); k, l \neq i, j.$$

Thus, Eq. (1) can be reformulated as:

$$Q_{ij} = \frac{(total - R_i - R_j)}{2(N-2)} + \frac{1}{2} D_{ij} \quad \text{.....(7)}$$

Furthermore, it seems that total and N are constants for every Q_{ij} at each iteration, so they can be excluded from the search for the minimum of Q_{ij}. This would lead to a more simplified form of Eq. (7). So, we can search for minimum Q_{ij} using Q'_{ij} evaluates as:

$$Q'_{ij} = D_{ij}(N-2) - (R_i + R_j) \quad \text{.....(8)}$$

By this derivation, we conclude that there is no need to compute total any more. And the sum of R_i needs to be computed just once. This deduced conclusion leads to more optimization for distance matrix computations. Suppose that Qi_{min},j_{min} is the smallest value which indicates that i_{min} and j_{min} are the neighbours to be joined in the tree. Then according to NJ the two rows and columns of i_{min} and j_{min} will be deleted from D, and a new row and column (denoted by U_1) are added. And D_{U1k} is computed according to formula (2). But in fact, R_i doesn't need to be computed again, because most values of the remaining R_i are unchanged between every two successive iteration step. Just R_{U1} have to be computed. While for the remaining $R_i$sonly D_{iimin} and D_{ijmin} are deleted, and D_{iU1} is added. This leads to saving O(N) time for each iteration, and the formulas for R_i, D_{iz} and D_{jz} are compensated by:

$$R_i = R_i - D_{ii_{min}} - D_{ij_{min}} + D_{iU_1} \quad \text{.....(9)}$$

$$D_{i_{min}z} = R_{i_{min}} - D_{i_{min}j_{min}}/(N-2)$$

$$D_{j_{min}z} = R_{j_{min}} - D_{i_{min}z}/(N-2)$$

Finally, we have applied the vectorization technique on R and Q as we do in our previous work with D. In [28], we computed the distance for all pairs and store it in a vector DV without affecting the performance. So, to compile the work in this paper with DV, we exchanged the matrix Q by a vector, and mapped the indices i and j onto an index h. The equations evaluating R andQ have been rewritten in terms of DV as follows:

$$R_i = \sum_{j=(i-1)N-\frac{i(i-1)}{2}+1}^{iN-\frac{i(i+1)}{2}} DV(i) + \sum_{k=1}^{i-1} DV(i - \frac{k(k+1)}{2} + N(k-1))$$

, where i=1 ... N. .....(10)

$$Q(h) = (N-2)DV(h) - (R(i) + R(j))$$, where i=1…N-1; j=i+1…N; h = ((i – 1) (2n – i) / 2) + j – i..... (11)

B. Parallelization Approach

This section proposes a vectorized parallel algorithm, called NJVect, for computing the guide tree. It depends mainly on the espoused idea in this work of switching D and Q into vectors. Its main goal is to speedup computations and reduces storage, to be able to align huge datasets.

First, we benefit from the independencies between (R_i) computations. Thus, computing the elements of (R_i) according to Eq.(10) can be parallelized. Each (i) iteration is performed in parallel by using multiple threads. Then, each core computes ⌈N/P⌉ permutations of i, where (P) is the number of cores.

Subsequently, computing the elements of (Q_h), where (h=1..N(N-1)/2) using Eq.(11), and finding its minimum, can be accomplished in parallel. We use multithread fine-grain parallelism technique. Each thread calculates minimum (Q_h) independently. Then ⌈n(n-1)/2P⌉ threads at each iteration works in parallel. Finally, the global minimum with $i_{min}$and j_{min} are determined, and the rest of procedure goes on.

The pseudo code of the proposed algorithm NJVect is given below. And, Fig.(3) clarifies how this proposed parallelization approach works.

Input : N (the number of OTUs), P (number of cores), DV (Distance Vector)
Output : NJ tree

1. fori=1 to N/P do in parallel
1.2 ComputeRi according to formula (l0).
1.3 end for
2. forn = N – 2 to 1 do
2.1 forh=1to n(n+1)/2P doin parallel
2.2.1 Compute Q_h according to formula (l1).
2.2.2 Find local minimum Q_h
2.3 end for
2.4 Search the global minimum Q_h from all local minimum to get neighboursi_{min} and j_{min}
2.5 Join OTUs i and j as a new OTU U_h.

2.6 Compute DV_{Uhk} according to Eq. (2).
2.7 Compute branch lengths L_{iUh} and L_{jUh} according to Eq. (3).
2.8 Delete OTUs i_{min} and j_{min}, and add U_h to current OTU lists.
2.9 end for
3. Join the last two OTUs.

V. EXPERIMENTAL RESULTS

The proposed NJVect algorithm has been implemented in C++, using OpenMP libraries and tested using a Core-i7 processor 2.00GHz and 8 GB RAMS running on Ubuntu Linux, 64-bit. The resulted program takes the advantage of fine-grained parallelism on a loop level, in which each process spawns a team of threads to occupy the multi-core processors when encountering parallel sections of code using OpenMP.

Comprehensive experiments have been conducted on the specified platform using different real protein datasets selected from the Human Immunodeficiency Virus (HIV) dataset downloaded from NCBI site. They evaluated the implementation of our proposed algorithm NVect versus MPI-NJ, the parallel NJ code of the popular aligner ClustalW-MPI, available at http://www.mybiosoftware.com/alignment/3052.

Fig.(4) demonstrates the performance for seven variant sequence sets, using execution time (in seconds) and speedup measurements. Results show that the performance increases as the number of sequences increases. As seen, our program achieves significant speedup of almost 2.5-fold over MPI-NJ. Best speedup is achieves when the number of sequences is more than 6000 sequences. This is compatible with the recent massive work for the tree construction and indicates gaining higher throughput.

VI. CONCLUSIONS

In this paper, authors introduced and evaluated a parallel vectorized algorithm of NJ, for efficiently reconstructing guide trees using multi-core systems. Our implementation, using OpenMP with C++, has achieved remarkable time and space reductions. Results were encouraging, it shows that as the number of sequences increase, NJVect performance increase, regardless of the sequences length. The resulted program outperforms ClustalW-MPI with 2.5-fold speedup. Thus, it suited well the very rapid growth of biological sequence databases. We believe that, the more provided cores, the better achieved performance will be.

Our future work will include the upgrade of this work to be compatible with multi-core cluster systems. It will be interesting to combine the proposed NJVect with DistVect producing a new efficient MSA tool.

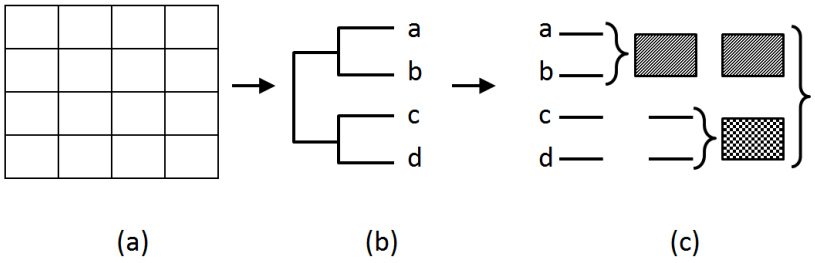


Figure (1): three stages of progressive sequence alignment:
(a) distance matrix; (b) guided tree and (c) progressive alignment along the guided tree.

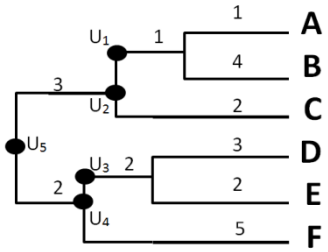
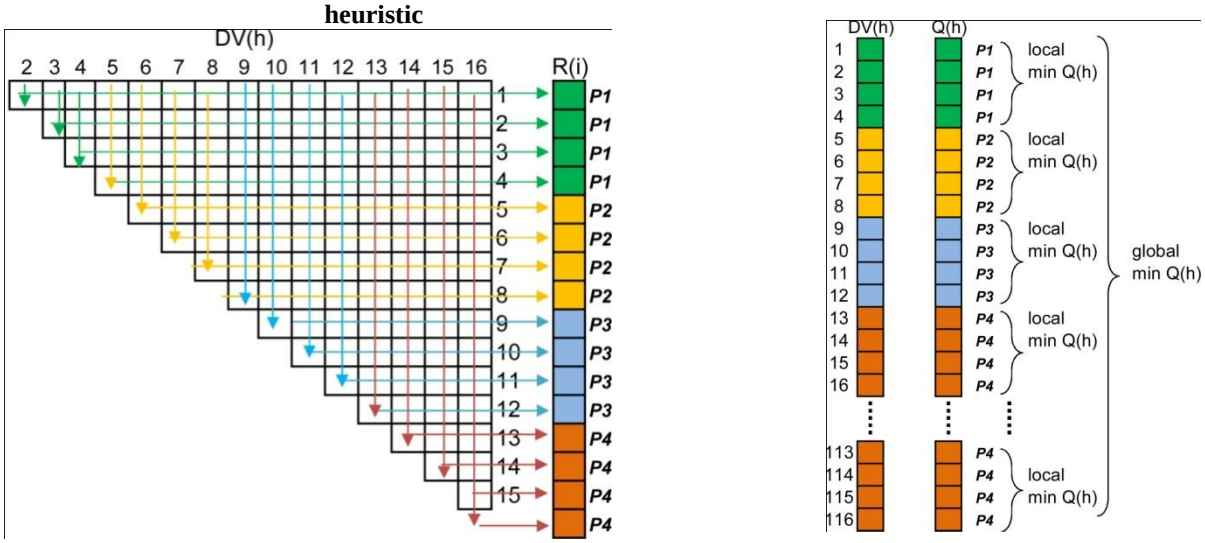


Fig. 2 constructing a guide tree using the Neighbour Joining heuristic



(a) **(b)**

Figure (3): NJVectprocessing 16 sequences using 4 cores
(a) R (i) computations (b) minimumQ (h) computes

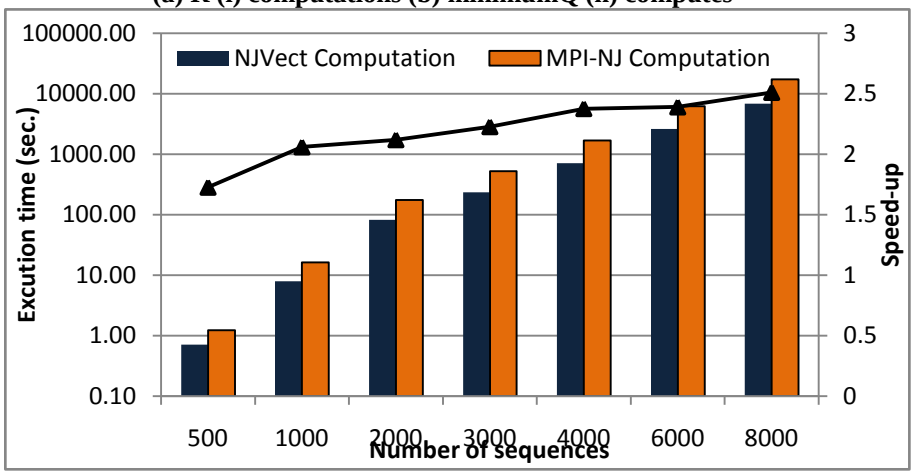


Figure (4): Performance comparison between MPI-NJ and NJVect

REFERENCES

[1] Lipman, D.J.; Altschul, S.F. and Kececioglu, J.D.: A tool for multiple sequence alignment. Proceedings of the National Academy of Sciences of the United States of America 86(12), 4412–4415. (1989).

[2] Thompson, J.D.; Higgins, D.G. and Gibson, T.J.: ClustalW: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. Nucl. Acids Res. 22(22), 4673–4680. (1994).

[3] Notredame, C.; Higgins, D.G. and Heringa, J.: T-coffee: A novel method for fast and accurate multiple sequence alignment. Journal of Molecular Biology 302(1), 205–217. (2000).

[4] Katoh, K.; Misawa, K.; Kuma, K.I. and Miyata, T.: MAFFT: A novel method for rapid multiple sequence alignment based on fast Fourier transform. Nucleic Acids Research 30(14), 3059–3066. (2002).

[5] Schmollinger, M.; Nieselt, K.; Kaufmann, M. and Morgenstern, B.: DIALIGN P: Fast pair-wise and multiple sequence alignment using parallel processors. BMC Bioinformatics. 5(128). (2004).

[6] Simossis, V.A. and Heringa, J.: PRALINE: A multiple sequence alignment toolbox that integrates homology-extended and secondary structure information. Nucleic Acids Research 33(suppl. 2), W289–W294. (2005).

[7] Michael C. Green. An Efficient Primer Selection Process CombiningProgressive And Iterative Multiple Sequence AlignmentStrategies: Clustalw And Hmmer. MSc. Thesis submitted to UniversitySan Luis Obispo. (2011).

[8] Durbin, R.; Eddy, S.R.; Krogh, A. and Mitchison G.J., Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids, Cambridge University Press, UK. (1998).

[9] Saitou N. and Nei M.: The neighbor-joining method: a new method for reconstructing phylogenetic trees, Mol. Biol. Evol., 4, 406-425. (1987).

[10] Catalyurek, U.; Stahlberg, E.; Ferreira and R., Saltzt, J.: Improving Performance of Multiple Sequence Alignment Analysis in Multi-client Environments. In: Proceedings of the First International Workshop on High Performance Computational Biology, HiCOMB. (2002).

[11] Catalyurek, U.; Gray, M.; Kurc, T.; Saltzt, J.; Stahlberg, E. and Ferreira, R.: A component based implementation of multiple sequence alignment. In: Proceedings of the ACM Symposium on Applied Computing, 122–126. (2003).

[12] Li, K.-B.: ClustalW-MPI: ClustalW analysis using distributed and parallel computing. Bioinformatics 19(12), 1585–1586. (2003).

[13] Chaichoompu, K.; Kittitornkun, S. and Tongsima, S.: MT-ClustalW: Multithreading multiple sequence alignment. In: 20th International Parallel and Distributed Processing Symposium, IPDPS. (2006).

[14] Oliver, T.; Schmidt, B.; Nathan, D.; Clemens, R. and Maskell, D.: Multiple sequence alignment on an FPGA. In: Proceedings of the International Conference on Parallel and Distributed Systems, ICPADS, 326–330. (2005).

[15] Liu, W.; Schmidt, B.; Voss, G. and Muller-Wittig, W.: Streaming Algorithms for Biological Sequence Alignment on GPUs. IEEE Transactions on Parallel and Distributed Systems. (2007).

[16] Liu, Y.; Schmidt, B. and Maskell, D.L.: MSA-CUDA: Multiple Sequence Alignment on Graphics Processing Units with CUDA. Proc. ICCS, Part I, Lecture Notes in Computer Science 5544, Baton Rouge, LO, 962–971. (2009).

[17] XiaozhongGeng.: A task scheduling algorithm for multi-core cluster systems. JCP, 7(11), 2797-2804. (2012).

[18] Du Z.H. and Feng B. pNJTree: A parallel program for reconstruction of neighbor-joining tree and its application in ClustalW. Parallel Computing, 32, 441-446. (2006).

[19] Howe KL.; Bateman A. and Durbin R. QuickTree: building huge Neighbour-Joining trees of protein sequences. Bioinformatics, 18(11):1546–1547. (2002).

[20] Sheneman L.; Evans J. and Foster J. Clearcut: A fast implementation of relaxed neighbor joining. Bioinformatics, 22(22):2823–2824. (2006).

[21] Du Z.H. and Feng B.: pNJTree: A parallel program for reconstruction of neighbor-joining tree and its application in ClustalW. Parallel Computing, vol.32, pp. 441-446. (2006).

[22] Bullard J. Panjo: a parallel neighbor joining algorithm, University of California, Berkeley, Tech. Rep. (2007).

[23] Liu Y.; Schmidt C. and Maskell D. L. Parallel reconstruction of neighbor-joining trees for large multiple sequence alignments using CUDA. Proceedings of the IEEE International Symposium on Parallel & Distributed Processing IPDPS, 1-8. (2009).

[24] Sahoo B.; Behura A. and Padhy S.: Fine Grain Construction of Neighbor-Joining Phylogenetic Tress with Reduced Redundancy Using Multithreading, International Journal of Distributed and Parallel Systems, IJDPS, 1(2), 129-140. (2010).

[25] Studier, J.A. and Kepler, K.J.: A note on the neighbor-joining method of Saitou and Nei. Molecular Biology and Evolution 5, 729–731. (1988).

[26] Mount D. W.: Bioinformatics: Sequences and Genome Analysis, Cold Spring Harbor Laboratory Press, New York (2001).

[27] Ran Z.; Qiongyao Z.; Hai J.; Zhiyuan S. and Xiaowen F.: Parallelization Mechanisms of Neighbor-Joining forCUDA Enabled Devices. Seventh ChinaGrid Annual Conference. (2012).

[28] Fayed F. M. Ghaleb; Naglaa M. Reda and Mohammed W. Al-Neama.: Fast vectorized distance matrix computation for multiple sequence alignment on multi-cores. Submitted to International Journal of Biomathematics. (2013).