Parallel PoMSA for Aligning Multiple Biological Sequences on Multicore Computers

Sara Shehab, Sameh Abdulah, and Arabi Keshk
Computer Science Department,
Faculty of Computers and Information,
Menofia University, Egypt

***Abstract*—The in-depth analysis of the evolutionary of the organisms requires understanding the functional relationship between their biological sequences. Building a homology and evolutionary model on these biological sequences could provide a clear picture of how these organisms developed by the time. Bioinformatics field helps to process, analyze and interpret biological data to automate processing workflow. Multiple Sequence Alignment (MSA) is one of the critical operations which allows a better understanding of the biological relation between sequences by trying to match similar sequences through an automated alignment process. This automation process is provided through a set of well-designed algorithms such as MAFFT, MUSCLE, Clustal-Omega, ...etc. With the explosion of data in different fields, parallel processing through shared memory machines, GPGPU accelerators, and distributed memory systems, becomes a necessity. Therefore, In this work, we propose a multicore version of our sequential MSA algorithm, PoMSA which is called Parallel PoMSA (PPoMSA). The proposed algorithm depends on the accuracy performance gained from the previous proposed sequential version with parallel support that impacts both the accuracy and the execution time performance of the alignment process. The evaluation results show that the proposed algorithm satisfies a higher alignment score compared to existing state-of-art algorithms: Clustal-Omega, MAFFT, and MUSCLE. Moreover, the scalability performance of the proposed algorithm has been evaluated using a different number of cores on a manycore machine and shows strong scalability with larger number of cores.**

I. Introduction

Finding similarity in RiboNucleic Acid (RNA) and DeoxyriboNucleic Acid (DNA), and amino acid (protein) is one of the fundamental objectives of bioinformatics through applying sequences alignment algorithms [4]. Sequences similarity are utilized for identifying evolutionary, structural or functional similarities among the biliogical sequences in a set of related RNA, DNA or proteins [1]. Sequences are used to illustrate evolutionary relationships among organisms. Thus, Sequence alignment is commonly used in several areas of biology including molecular biology and structural biology. Sequence alignment process is the process of aligning three or more DNA, RNA, or protein sequences to show the similar and different regions which scientifically called Multiple sequence alignment (MSA). The final score of aligned sequences can detect the degree of similarities and difference. The higher score value represents a higher similarity degree between the sequences.

In literature, many algorithms have been developed with a goal of maximizing the similarity score and get the maximum match between the target sequences. In this case, the aligned sequences can be used effectively in the process of inferring sequence homology. In literature, many techniques and strategies have been shown to obtain MSAs, such as progressive alignment methods [14], [5], iterative methods [3], [7], dynamic programming [2], genetic algorithms [9], greedy algorithms [17], Markov chain processes [10], and even simulated annealing methods [18]; see [16] for a review.

In [12], [26], we have presented a new heuristic algorithm, Precise Position-based Multiple sequence alignment (PoMSA), where a position matrix has been used to guide the gap insertion operations to satisfy a higher degree of matching between target sequences. This position matrix can be used to reconstruct the given set of sequences in a more aligned format. This paper is an extension of our work in [12]. PoMSA uses different strategy compared to what other MSA algorithms used to align a given set of sequences. The PoMSA algorithm is not a progressive-alignment approach, as all sequences are aligned simultaneously. In contrast to existing heuristic alignment methods, which start from entirely unaligned sequences, the PoMSA algorithm represents unaligned sequences as a position matrix which is aligned by filling gaps into certain positions.

Here, we provide a parallel version of the PoMSA algorithm which is called Parallel PoMSA (PPoMSA). The proposed parallel algorithm depends on the partitioning strategy proposedby [12]. We exploited the partitioning mechanism to distribute the work between available cores on multicore systems. Both accuracy and scalability experiments have applied on a multicore system to validate the effectiveness of the proposed algorithm. Three datasets have been used in the evaluation process, BAliBase [15], OXBench [11], and SMART [8]. The proposed PPoMSA algorithm is compared with the state-of- Art algorithms such as Clustal-Omega [14], MAFFT [6], and MUSCLE [3].

II. Related Work

This section provides a detailed background on our baseline algorithm (PoMSA) and a set of state-of-the-art MSA algorithms that are used for evaluating our proposed parallel version of PoMSA algorithm.

A. *Position-based Multiple Sequence Alignment (PoMSA)*

In [12], a novel heuristic multiple sequence alignment algorithm has been proposed which is called PoMSA. PoMSA algorithm depends on generating a position matrix instead of a distance matrix that is widely used by existing MSA algorithms. A full scan process is applied over the position matrix to insert gaps in certain positions to increase the similarity between sequences.

Given S is the set of sequences as input; S is inserted into a matrix M where the number of rows is fixed and equals the number of sequences S. Initially, the number of columns equals the length of largest sequence length L. If a certain sequence length l_i is less than L, n gaps are added to the end of this sequence where $n = L - l_i$ (i.e., sequences adjustment step). We call the generated matrix as *position matrix*. Firstly, PoMSA scans the position matrix column by column. Assuming a pre-determined threshold ϵ, If the number of a certain sequence bases (i.e., adenine (A), cytosine (C), guanine (G), thymine (T)) per column c_i equals or larger than ϵ, this means that we need to set this base as the dominant base for this column. The sequences that have another base in this column should move the unmatched bases in a way that increases the overall matching for this column. For unmatched cases, if the previous column c_{j-1} contains a base that equals to current column c_j dominant base, a single gap should be inserted into column c_{j-1} and next bases should be shifted one step forward for this sequence. To keep sequences adjusted a single gap should be added to the end of unchanged sequences. If the next column c_{j+1} have a dominant base, a single gap is added at the end of such a sequence and a single gap also should be added at column c_i for all other sequences. Algorithm 1 illustrates the whole operation of the PoMSA algorithm.

Algorithm 1 : PoMSA Algorithm

```
1:  Input: $S \Rightarrow$ set of given sequences.
2:  Build a position matrix $M$.
3:  Adjust matrix $M$ by adding $L - l_i$ gaps.
4:  for col:0 to $L$ do
5:      for row:0 to $S.length$ do
6:          $dominant \Rightarrow$ getDominant($M$, $col$)
7:          if $M[row[col] \neq dominant$ then
8:              if $M[row[col-1]$ = $dominant$ then
9:                  Add gap at M[row][col-1]
10:                 add gap to the end of other sequences.
11:             end if
12:             if $S[row][col+1]$ = $dominant$ then
13:                 Add gap at the end of sequence M[row].
14:                 Add gap at index $col$ for other sequences.
15:             end if
16:         end if
17:     end for
18: end for
```

B. Clustal-Omega

Clustal-Omega is a progressive alignment approach to align multiple biological sequences. Progressive algorithms have been described as "greedy algorithms" where any incorrect aligning at the beginning of the algorithm cannot be corrected in the next iterations. T-Coffee algorithm was the first milestone towards flexible MSA progressive algorithms which is more accurate than previous progressive algorithms. However, T-Coffee algorithm was not able to align more than a few hundreds of sequences (i.e., weak scalability).

Clustal-Omega is the first algorithm that is able to give better accuracy such as T-Coffee algorithm, and at the same time, it is able to scale to cover thousands of sequences even with a single core. The scalability comes from aligning all N sequences using $O(N^2)$ memory requirement using the guide tree structure. In [20], Fabian et al., report that Clustal-Omega is used to align around 200K sequences using a single core. The idea of Clustal-Omega depends on *mBed*, which is a fast and accurate clustering algorithm for large numbers of sequences [21]. *mBed* can help on designing guide trees for Clustal-Omega with $O(NlogN)$ instead of usual $O(N^2)$ for existing algorithms.

C. MAFFT

Clustal-Omega is not the only MSA algorithm that can handle a large number of sequences. MAFFT can speedup aligning thousands number of sequences but with less accuracy. MAFFT is a progressive method based on Fast Fourier Transform (FFT) for multiple sequence alignment. FFT helps to discover fast matched parts in a given set of sequences.

The algorithm generates a distance matrix and a guide tree which shows the similarity distances between the sequences. In MAFFT, alignment is performed using three main steps, all-to-all comparison, progressive alignment, and iterative refinement. All-to-all comparison aims at performing a set of pairwise comparisons to generate the initial distance matrix and the guide tree that are used by the progressive alignment step. The Progressive alignment is combining the results from the first step to perform the main MSA process. iterative refinement step is used to improve the accuracy of the progressive alignment step.

MAFFT software has three heuristic running modes, progressive method (FFT-NS-2), the iterative refinement method (FFT-NS-i, L-INS-i, E-INS-i, and G-INS-i), and the structural alignment method for RNA (Q-INS and X-INS-i) [6], [19]. Each mode is suitable for a certain number of sequences and the user priority (i.e., fast or accurate computation). This option usually used in the case of a very large number of sequences and fast computation priority from the user. In the iterative refinement method, accurate alignment is considered a priority and several alignment iterations are required. Further, structural alignment method is more suitable for RNA alignment proteins with low sequence similarity.

In [24], a parallel *MAFFT* algorithm has been proposed. The proposed implementation includes the three main steps of the sequential MAFFT algorithm using POSIX threads library. All-to-all comparison has been parallelized by distributing the pairwise alignment to the available physical cores. In progressive alignment, parsing the guide tree is parallelized every level, i.e., each tree parent depends on its child which prevents parallelization over the whole tree nodes. Finally, if the iterative refinement is used, two different approaches have been proposed, *best-first approach* and *hill-climbing approach*, where both of them aim at dividing each alignment into two sub-alignments and the two sub-alignments are realigned. The refinement step is performed according to the tree dependent iterative strategy proposed in [13].

In the *best-first approach*, The realignments are performed for all possible alignments on the iterative tree. The alignment with the highest objective score is selected for another iteration. Once no high score alignment is found, the algorithm terminates. In the *hill-climbing* approach, each process has its local realignments of the original alignment and the better score is replaced the original alignment is kept locally by each process. In this case, many realignments of the same alignment are used in the next iteration [24].

D. MUSCLE

MUSCLE is another progressive algorithm which is widely used to align a set of given biological sequences. The algorithm includes three main steps, draft progressive, improved progressive, and refinement step.

The algorithm have a set of unaligned sequences as an input. A distance matrix is generated by calculating *K-mer* distance for each pair of sequences. Based on the generated matrix, Unweighted Pair Group Method with Arithmetic Mean (*UPGMA*) is used to cluster the sequences by producing a binary tree. *UPGMA* algorithm is a simple bottom-up hierarchical method that is widely used for clustering purpose [22].

A set of progressive alignment steps are performed on each pair of sequences based on the generated binary tree. Because the *K-mer* distance estimation technique is based on approximation. The resulted binary tree is not optimal which impacts the first alignment process. Thus, another distance estimation technique (i.e., *Kimura* distance [23]) is used by MUSCLE to re-estimate the corresponding binary tree. The *UPGMA* algorithm is used one more time to cluster the distance matrix generated by the *Kimura* technique. Another progressive alignment is applied to the resulted binary tree and produce a new set of aligned sequences. This is optimized by computing alignments only for sub-trees whose branching orders changed relative to first generated tree. Finally, another refinement step is applied to optimize the resulted aligned sequences set.

III. Proposed Work

We propose a parallel version of the PoMSA that was implemented under Linux using OpenMP with C++. In the parallel version, the generated position matrix is divided into a set of partitions based on a pre-determined user threshold ϵ. The position matrix scheme helps in finding the correct position to insert or delete gaps to obtain a high degree of similarity. This threshold determines the percentage of any sequence base on a certain column col_j. Each partition is processed by one process. The final output is generated by merging the output of different partitions.

Figure 1 shows the overall operation of the PPoMSA algorithm. The algorithm starts with a set of input sequences. An adjustment process is applied to all sequences to have the same number of bases in each sequence. The new set of sequences is divided based on ϵ threshold and a separate process is initiated for each partition. In the case of cores availability, processes are distributed between available core cores. Each process will apply the origin PoMSA algorithm on each partition and the final output can be obtained by merging the outputs in one matrix.

IV. Performance Evaluation

In this section, we aim at evaluating the proposed parallel algorithm with several objectives. First, evaluating the quality of sequential PoMSA compared to existing MSA algorithms.

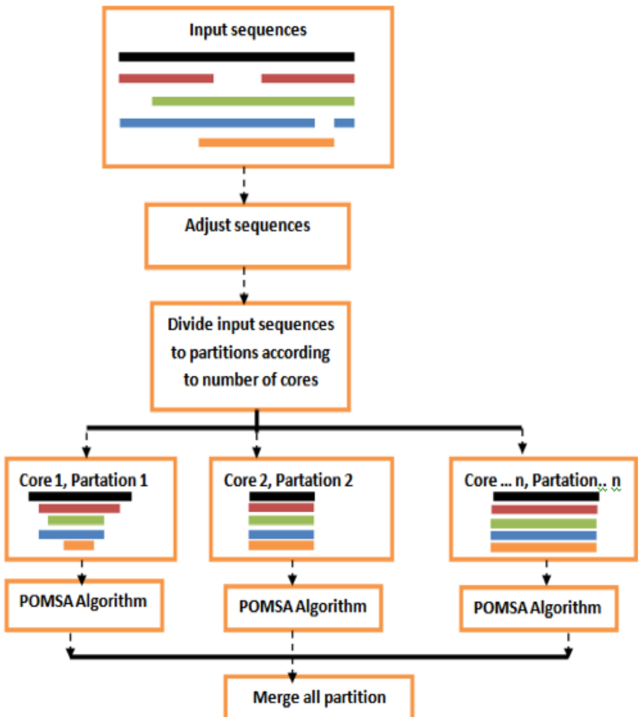

Fig. 1: Proposed Parallel PoMSA.

Second, evaluating the performance of parallel PoMSA algorithm with different number of cores. Finally, evaluating the accuracy of PPoMSA algorithm with different thresholds and different number of partitions.

All the experiments have been performed on 8-core Intel Xeon (E5506) machine, running at 2.13 GHz, with 8GB RAM memory. The implementation is done under Linux using C++ and OpenMP library for multithreading. Three datasets have been used for evaluation, BAliBASE [15], OXBench [11], and SMART [25].

A. Performance comparison of PoMSA with the State-Of-The-Art algorithms

In [12], we evaluated the accuracy effectiveness of the PoMSA algorithm compared to existing algorithms using only the sum-of-pair (SP) score. In this section, we extend our evaluation to cover more evaluation metrics, Quality (Q) score and Total Column (TC) score. The Q score defines the number of correctly aligned residue pairs divided by the number of residue pairs in the reference alignment. The TC score defines the number of correctly aligned columns divided by the number of columns in the reference alignment [3]

Here, we are using three different biological DNA datasets for our experiments, BAliBase, OXBench and SMART. We compare PoMSA versus three other algorithms, MUSCLE, Clustal Omega, and MAFFT. Tables I, II, and III show the Q/TC scores of each algorithm on different data file. The results show that PoMSA always performs better than other algorithms with high similarity Q/TC scores with different datasets.

TABLE I: QAND TC Score Using BAliBase Dataset.

	Q/TC			
File Name	PoMSA	MUSCLE	Clustal-Omega	MAFFT
BB11025	0.561/0.503	0	0.043/0	0.088/0
BBS11002	0.577/0.152	0.348/0	0.354/0	0.343/0.014
BBS11008	0.444/0.169	0.256/0.124	0.243/0.124	0.277/0.141
BBS11025	0.4/0.243	0.0461/0.064	0	0.092/0.111
BBS12009	0.579/0.377	0.53/0.348	0.631/0.358	0.518/0.324
BBS12039	0.777/0.292	0.386/0	0.44/0	0.459/0.104

TABLE II: QAND TC SCORES ON OXBench Dataset.

	Q/TC			
File Name	PPoMSA	MUSCLE	Clustal-Omega	MAFFT
4t3	0.678/0.469	0.762/0	0.047/0	0.081/0
22t42	0.519/0.193	0.368/0.003	0.425/0	0.371/0.003
22t49	0.374/0.04	0.175/0	0.187/0	0.179/0
471	0.488/0.269	0.372/0	0.37/0	0.376/0.003
512	0.508/0.272	0.157/0	0.168/0	0.171/0
512s1	0.763/0.68	0.296/0	0.33/0	0.305/0

B. PPoMSA Performance Evaluation

Parallel algorithms should show strong scalability with increasing the number of processes on manycore systems. In this experiment, we analyze the performance in term of execution time and scalability. We conducted a set of experiments on BAliBase, OXBench and SMART datasets. Figures 2, 3, and 4 show a strong scalability with increasing the number of used cores from 1, 2, 3, up to 4 cores. All the values in the figures represent time in milliseconds. In these set of experiments, Hyper-threading was disabled and the affinity was set to the physical cores. The results show execution time decreasing with larger number of cores with linear speedup from one to four cores. Although the number of sequences in different datasets is relatively small, PPoMSA algorithm is able to provide strong scalability.

C. PPoMSA Accuracy Evaluation Using Different Thresholds

Based on PoMSA algorithm, reducing the similarity threshold leads to a larger number of partitions which impacts both accuracy and execution time with parallel systems. This set of experiments shows how TC and Q scores vary with different similarity threshold. The experiment uses three threshold levels, 50%-75%, 75%-95%, and ABOVE 90%. Tables IV, V, and VI show the results on our target datasets, BAliBase, OXBench, and SMART. 50%-75% threshold should provide

TABLE III: QAND TC scores on SMART Dataset.

	Q/TC			
File Name	PPoMSA	MUSCLE	Clustal-Omega	MAFFT
AXH	0.504/0.272	0.283/0.0611	0.386/0.15	0.326/0.169
LCCL	0.761/0.627	0.392/0.141	0.395/0.141	0.4/0.152
POU	0.637/0.369	0.339/0	0.346/0	0.306/0
PRE-C2HC	0.5/0.306	0.384/0.261	0.389/0.252	0.49/0.374
SAND	0.728/0.505	0.631/0.261	0.572/0.144	0.4/0.023
TAFH	0.424/0.294	0.069/0	0.186/0.179	0.151/0.129

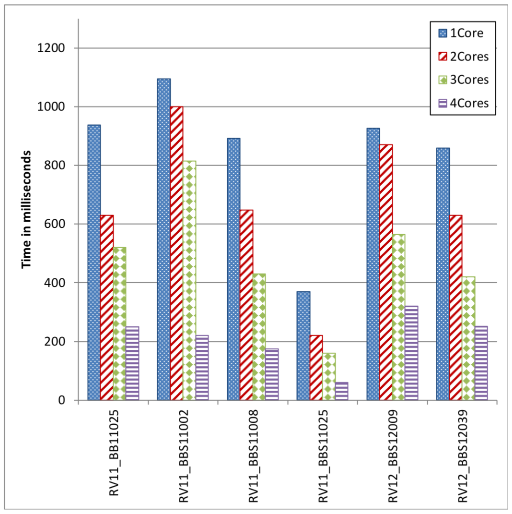

Fig. 2: Execution Time with Different Number of Cores on BAliBase dataset.

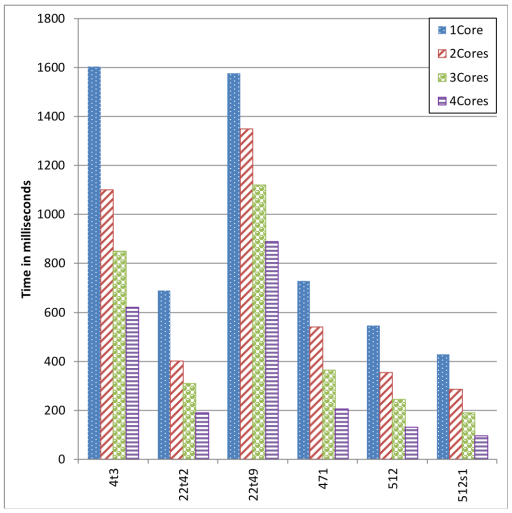

Fig. 3: Execution Time with Different Number of Cores on OXBench dataset.

a larger number of partitions, however, it will impact the accuracy as shown from the tables. On the contrary ABOVE 90% similarity will provide less number of partitions but with higher accuracy impacts. If a large threshold is able to produce enough number of partitions that can feed all available cores, both accuracy and execution time will positively be impacted.

As shown in these tables we notice that when the threshold value increases the TC and Q score increases so the accuracy will be improved with increases of threshold values.

V. Conclusion

We present a parallel version of PoMSA, a new and efficient data parallel strategy for large-scale multiple sequence alignment on current multi-core systems. Because of the bad of accuracy and execution time some algorithms have, the Parallel PoMSA provide solution to finish this task. In the parallel version we test results using core i5 machine and divide the

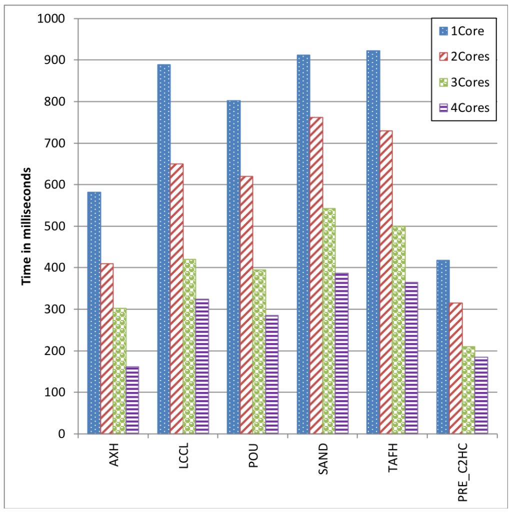

Fig. 4: Execution Time with Different Number of Cores on SMART dataset.

TABLE IV: Q and TC Scores on BaliBase Dataset Using Different Threshold Levels.

	Q/TC		
File Name	(50%-75%)	(75%-95%)	ABOVE 90%
BB11025	0.0952/0.214	0.561/0.503	0.672/0.794
BBS11002	0.0238/0.29	0.577/0.152	0.876/0.937
BBS11008	0.169/0.444	0.444/0.169	0.757/0.836
BBS11025	0.0952/0.214	0.4/0.243	0.672/0.794
BBS12009	0.26/0.437	0.579/0.377	0.824/0.887
BBS12039	0.0290/0.217	0.777/0.292	0.9/0.953

TABLE V: Q and TC Scores on OXBench Dataset Using Different Threshold Levels..

	Q/TC		
File Name	(50%-75%)	(75%-95%)	ABOVE 90%
4t3	0.0504/0.25	0.678/0.469	0.698/0.801
22t42	1/0.293	0.519/0.193	0.91/0.958
22t49	0.00333/0.105	0.374/0.04	0.75/0.963
471	0.269/0.488	0.488/0.269	0.91/0.943
512	0.272/0.508	0.508/0.272	0.645/0.766
512s1	0.377/0.528	0.763/0.68	0.68/0.763

TABLE VI: Q and TC Scores on SMART Dataset Using Different Threshold Levels.

	Q/TC		
File Name	(50%-75%)	(75%-95%)	ABOVE 90%
AXH	0.272/0.504	0.504/0.272	0.686/0.777
LCCL	0.627/0.761	0.761/0.627	0.743/0.864
POU	0.369/0.637	0.637/0.369	0.898/0.947
PRE-C2HC	0.306/0.5	0.5/0.306	0.892/0.936
SAND	0.505/0.728	0.728/0.505	0.73/0.858
TAFH	0.294/0.425	0.424/0.294	0.53/0.739

input sequences according to number of available process and also apply the partitioning scheme to the input sequences, the results show that the execution time will decreased up to 50% compared to other algorithms and sequential version and also SP score will be increased in the case of increasing number of partitions. We also compute the accuracy and evaluate the TC and Q score for PoMSA and other algorithms MAFFT, Clustal-Omega and MUSCLE, the result shows that PoMSA achieved high accuracy when compared to these algorithms.

References

[1] Timothy L Bailey. Discovering sequence motifs. In *Bioinformatics*, pages 231–251. Springer, 2008.

[2] Yonatan Bilu, Pankaj K Agarwal, and Rachel Kolodny. Faster algorithms for optimal multiple sequence alignment based on pairwise comparisons. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 3(4), 2006.

[3] Robert C Edgar. MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic acids research*, 32(5):1792–1797, 2004.

[4] Martin Gollery. Bioinformatics: Sequence and genome analysis, david w. mount. cold spring harbor, ny: Cold spring harbor laboratory press, 2004, 692 pp., , paperback. isbn 0-87969-712-1. *Clinical Chemistry*, 51(11):2219–2219, 2005.

[5] DG Higgins, J Heringa, et al. T-Coffee: A novel method for fast and accurate multiple sequence. *Journal of molecular biology*, 302(1):205–217, 2000.

[6] Kazutaka Katoh and Daron M Standley. MAFFT multiple sequence alignment software version 7: improvements in performance and usability. *Molecular biology and evolution*, 30(4):772–780, 2013.

[7] Burkhard Morgenstern. DIALIGN: multiple DNA and protein sequence alignment at BiBiServ. *Nucleic acids research*, 32(suppl_2):W33–W36, 2004.

[8] Burkhard Morgenstern. Multiple sequence alignment with DIALIGN. In *Multiple Sequence Alignment Methods*, pages 191–202. Springer, 2014.

[9] Cédric Notredame and Desmond G Higgins. SAGA: sequence alignment by genetic algorithm. *Nucleic acids research*, 24(8):1515–1524, 1996.

[10] Jimin Pei and Nick V Grishin. MUMMALS: multiple sequence alignment improved by using hidden Markov models with local structural information. *Nucleic acids research*, 34(16):4364–4374, 2006.

[11] GPS Raghava, Stephen MJ Searle, Patrick C Audley, Jonathan D Barber, and Geoffrey J Barton. OXBench: a benchmark for evaluation of protein multiple sequence alignment accuracy. *BMC bioinformatics*, 4(1):47, 2003.

[12] Sara Shehab, Sameh Shohdy, and Arabi E Keshk. PoMSA: An Efficient and Precise Position-based Multiple Sequence Alignment Technique. *Journal of Advanced Research in Computing and Applications*, 9(1):14–20, 2017.

[13] Osamu Gotoh. Optimal alignment between groups of sequences and its application to multiple sequence alignment. *Bioinformatics*, 9(3):361–370, 1993.

[14] Julie D Thompson, Desmond G Higgins, and Toby J Gibson. Clustal w: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic acids research*, 22(22):4673–4680, 1994.

[15] Julie D Thompson, Patrice Koehl, Raymond Ripp, and Olivier Poch. BAliBase 3.0: latest developments of the multiple sequence alignment benchmark. *Proteins: Structure, Function, and Bioinformatics*, 61(1):127–136, 2005.

[16] Xiao-Dan Wang, Jin-Xing Liu, Yong Xu, and Jian Zhang. A survey of multiple sequence alignment techniques. In *International Conference on Intelligent Computing*, pages 529–538. Springer, 2015.

[17] Zheng Zhang, Scott Schwartz, Lukas Wagner, and Webb Miller. A greedy algorithm for aligning dna sequences. *Journal of Computational biology*, 7(1-2):203–214, 2000.

[18] Jaroslaw Zola, Denis Trystram, Andrei Tchernykh, and Carlos Brizuela. Parallel multiple sequence alignment with local phylogeny search by

simulated annealing. In — *Proceedings 20th IEEE International Parallel & Distributed Processing Symposium*, page 279. IEEE, 2006.

[19] MAFFT software, https://mafft.cbrc.jp/alignment/software/, 10 10 2018.

[20] Sievers, Fabian and Wilm, Andreas and Dineen, David and Gibson, Toby J and Karplus, Kevin and Li, Weizhong and Lopez, Rodrigo and McWilliam, Hamish and Remmert, Michael and Söding, Johannes and others. Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega. *Molecular systems biology*, 7(1):539, 2011.

[21] Blackshields, Gordon and Sievers, Fabian and Shi, Weifeng and Wilm, Andreas and Higgins, Desmond G. Sequence embedding for fast construction of guide trees for multiple sequence alignment. *Algorithms for Molecular Biology*, 5(1):21, 2010.

[22] Sokal, Robert R. A statistical method for evaluating systematic relationship. *University of Kansas science bulletin*, 1409–1438, 1958.

[23] Kimura M. A simple method for estimating evolutionary rate of base substitutions through comparative studies of nucleotide sequences *Journal of Molecular Evolution*, 16, 11–120, 1980.

[24] Katoh, Kazutaka and Toh, Hiroyuki. Parallelization of the MAFFT multiple sequence alignment program. *Bioinformatics*, 26(15), 1899–1900, 2010.

[25] Ivica Letunic and Peer Bork 20 years of the SMART protein domain annotation resource. *Oxford University Press*, 46(D1), D493–D496, 2017.

[26] Sara Shehab, Sameh Abdulah, Arabi E. Keshk A Survey of the State-of-the-Art Parallel Multiple Sequence Alignment Algorithms on Multicore Systems. *International Journal of Computer Applications*, 182(12), 0975 - 8887, August, 2018.