Scalable Multiple Sequence Alignment via Genetic Algorithms and localized Deep Reinforcement Learning Agents

Supplementary Materials

I. Experimental results

We carried out several experiments to evaluate the proposed approach, by addressing the following *research questions*:

1) RQ1: can our method produce reliable results?
2) RQ2: can the scalability limitations of the RL base agent be addressed by training on smaller boards and deploying it adaptively for larger tasks?
3) RQ3: can our method outperform state-of-the-art MSA techniques?

*a) **Dataset**:* The experiments were organized in:

- *Phase 1: synthetic data experiments.* The first stage focused on addressing the research questions described above through a controlled evaluation and calibration of the proposed approach using synthetic datasets; this step aimed to optimize key parameters, such as population size, number of generations, mutation rate, and crossover rate; additionally, we investigated the impact of the board size used for training the DRL agent, assessing its effect on alignment accuracy and computational efficiency; to this aim, we developed a controlled and stable environment for generating synthetic datasets designed to emulate real biological conditions while maintaining consistency for comparative analysis; such a controlled environment allowed us to calibrate the method systematically before transitioning to real-world sequences (see Section V for details about the simulation framework).
- *Phase 2: real data validation.* Once the optimal configuration was identified from synthetic experiments, the proposed method was tested on real biological datasets to validate its generalization ability and benchmark its performance against widely used MSA tools.

The synthetic datasets are each organized into 50 sub-datasets (see Section V). Specifically, we produced: *(i)* Dataset1, with 3 sequences of length 30 per sub-dataset (board 3×30); *(ii)* Dataset2, with 6 sequences of length 30 per sub-dataset (board 6×30); *(iii)* Dataset3, with 3 sequences of length 60 per sub-dataset (board 3×60), and *(iv)* Dataset4, with 12 sequences of length 150 per sub-dataset (board 12×150).

The real datasets are generated by extracting DNA-seq sequences from the ENCODE (Encyclopedia of DNA Elements) project [1]. From the experimental search, the *DNase-seq in activated T-cell* experiment [2] is selected together with the associated *fastq* file. From the DNA-seq fastq files, to create optimal and controlled real biological data, the *R1 fastq* is used to extract 40 DNA sequences of 101 base pairs long. Those sequences formed the 10 real biological data used to test the proposed approach and the benchmark tools.

*b) **Baseline methods and configurations**:* For our experiments, we selected the following widely adopted MSA methods for comparative evaluation: ClustalW, ClustalΩ, MSAProbs, PASTA, Muscle, UPP, MAFFT, and DPAMSA. The configuration settings for DPAMSA followed the specifications proposed in Liu et al. (2023), with the optimal *hyperparameters* setting values provided in Section V. In the following, the proposed method, which combines the *GA orchestrator* with the "local" *RL base model*, will be referred to as GA-DPAMSA. Such a method has been tested across the various experiments by adjusting and varying the parameters. For each experiment conducted, we report the results achieved with the best setting (see Section VI).

We remark that, in *Phase 1*, we focused on evaluating the performance of ClustalW, ClustalΩ, MSAProbs, and DPAMSA, as these methods represent a diverse set of well-established alignment strategies, including progressive, iterative, and RL-based approaches. These methods provided a robust baseline for assessing the effectiveness of our proposed GA-driven alignment strategy. However, when transitioning to real biological datasets, additional complexities arise, such as varying evolutionary distances, structural constraints, and the presence of highly divergent sequences. To ensure a comprehensive evaluation under realistic conditions, we extended our comparative analysis by incorporating PASTA, MUSCLE, UPP, and MAFFT, which have been specifically designed to handle large-scale biological alignments with improved scalability and robustness. These methods employ advanced techniques such as profile-based iterative refinement (MUSCLE, MAFFT), phylogeny-aware progressive alignment (PASTA), and statistical profile modeling (UPP), making them particularly well-suited for challenging real-world datasets. Given that synthetic datasets were designed to provide a controlled benchmarking environment, testing these additional tools was not essential for the initial calibration phase. Their inclusion in real-data experiments, however, ensures a more rigorous and biologically

[1] https://www.encodeproject.org/

[2] https://www.encodeproject.org/experiments/ENCSR300NZA/

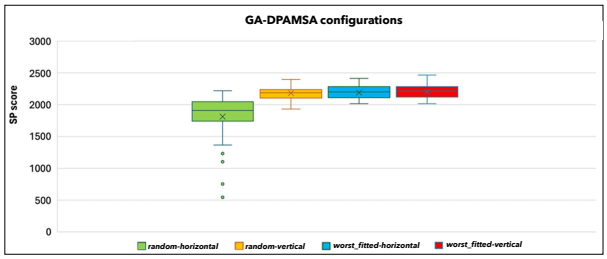

Fig. 1. GA-DPAMSA - *mutation-crossover* configurations

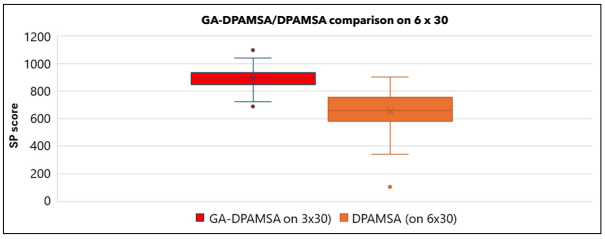

Fig. 2. Scalabity evaluation of DPAMSA on boards *6 x 30*

meaningful evaluation of our method in comparison to state-of-the-art MSA tools tailored for complex genomic datasets.

The following metrics are used to evaluate the methods: *(i) SP score*, *(ii)* CS score, *(iii) NASP*, i.e., the number of optimal alignments based on SP score over one type of dataset, and *(iii) NACS*, i.e., the number of optimal alignments based on CS score over one kind of dataset. The experiments have been conducted on a Intel(R) Xeon(R) Gold, with NVIDIA Quadro GPU, 232 GB, 5218 CPU @ 2.30 GHz, RTX 4000. The source code and further details of the proposed method GA-DPAMSA are available online[3].

II. GAs and Multi-objective optimization

Genetic Algorithms (GAs) emulate the evolutionary dynamics observed in natural selection by simulating the adaptation of a population over successive generations. A set of candidate solutions, referred to as *individuals*, represents abstract encodings known as *chromosomes*. The algorithm initializes a population randomly, which then undergoes iterative refinement through a sequence of evolutionary steps. During each of such steps, or *generation*, individuals are evaluated based on a fitness function that quantifies their quality. A subset of the

[3]https://github.com/FLaTNNBio/GA-DPAMSA

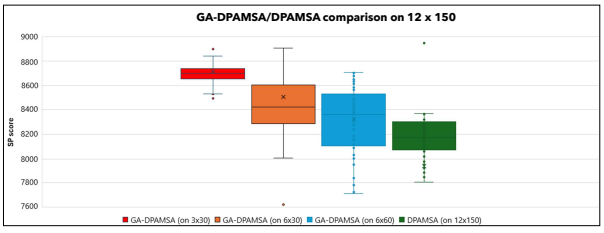

Fig. 3. Scalabity evaluation of DPAMSA on boards *12 x 150*

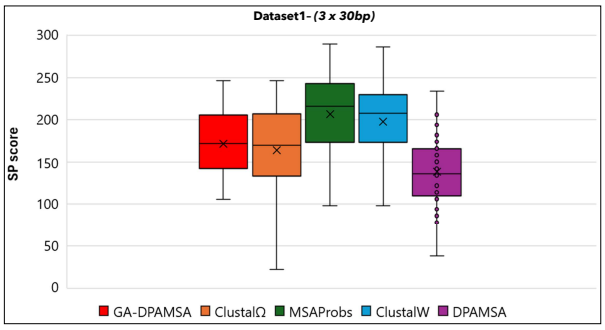

Fig. 4. Average SP score on boards *3x30*

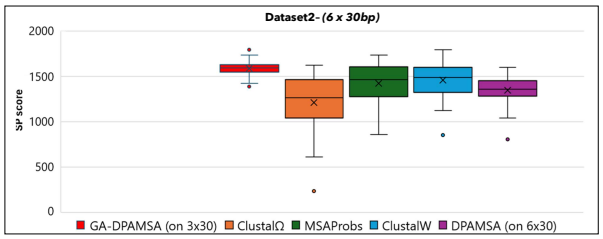

Fig. 5. Average SP score on boards *6x30*

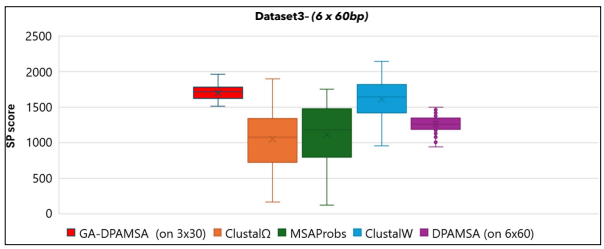

Fig. 6. Average SP score on boards *6x60*

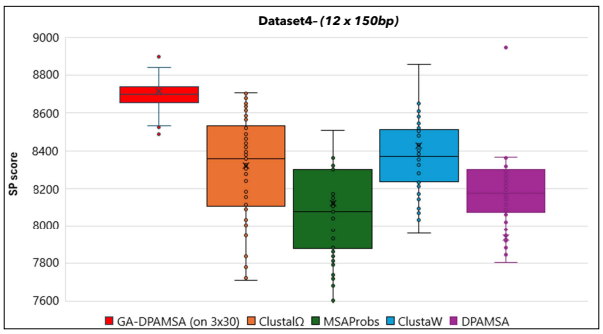

Fig. 7. Average SP score on boards *12x150*

population is then probabilistically selected (*selection*) according to their fitness, undergoing recombination (*crossover*) and stochastic modifications (*mutation*) to generate a new set of individuals. This evolutionary cycle continues across multiple generations until either a predefined number of iterations is reached or an optimal solution emerges.

In many optimization scenarios, the fitness function is designed to optimize a single objective, resulting in a conventional single-objective optimization problem, either maximization or minimization. However, numerous real-world problems require the simultaneous consideration of multiple, often conflicting, objectives, leading to a multi-objective optimization framework.

A *Multi-Objective Genetic Algorithm* (MOGA) evaluates individuals based on k multiple fitness functions in a maximization problem:

$$\max(f_1(x), f_2(x), ..., f_k(x)) \tag{1}$$

where each objective function $f_i : X \to \mathbb{R}$, for $i = 1, ..., k$, maps the solution space X to the real numbers. A fundamental challenge in *multi-objective optimization* is that, in most cases, it is impossible to maximize all objectives simultaneously. The optimization of one criterion may lead to a deterioration in another, necessitating a trade-off between competing goals. Consequently, rather than seeking a single globally optimal solution, the goal is to identify a diverse set of *non-dominated* solutions that represent the best possible compromises among objectives.

A solution y_1 *dominates* another solution y_2 if it outperforms y_2 in all objective functions. It is considered *optimal* if no other solution in the search space dominates it. The set of all non-dominated solutions constitutes the *Pareto front*, representing the best possible trade-offs among conflicting objectives. MOEAs, including MOGAs, aim to approximate the Pareto front by generating solutions that closely reflect the optimal balance between competing criteria. Once this set of high-quality solutions is obtained, a subsequent decision-making process can be employed to select the most appropriate solution based on domain-specific requirements.

Various MOEAs have been proposed across different applications, with numerous comparative studies evaluating their performance. In this work, we employ a multi-objective genetic algorithm built upon a specialized adaptation of NSGA-II. One of the key features of NSGA-II is its *elitism* mechanism, which maintains an external archive of all non-dominated individuals. This external population is crucial, as it preserves high-quality solutions that contribute to the final alignment. By incorporating elitism, we ensure that superior genetic traits are retained throughout the evolutionary process. Individuals from the external population actively participate in recombination operations, further enhancing the optimization process. Our algorithm integrates this technique to refine the alignment while maintaining diversity and convergence toward the Pareto front.

III. MSA: FORMAL PROBLEM DEFINITION

Given a dataset S that contains s sequences, a corresponding alignment is denoted as CA, which is often not unique; in fact, there can be multiple or even countless possible results, indicated as $\{CA_i\}_{i=1}^{\infty}$. To evaluate these alignments, we utilize two scoring systems, the *Sum-of-Pairs* (SP) score and the *Column score* (CS). SP score measures the pairwise similarity between sequences in an alignment, and is calculated using the following formula:

$$SP(CA) = \sum_{k=1}^{\text{len}(CA)} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} p(CA^{i,k}, CA^{j,k}),$$

where $len(CA)$ is the length of a single sequence in the alignment CA, and $CA^{i,k}$ represents the k-th nucleotide in the i-th sequence. The function $p(x, y)$ is used to calculate the matching score between two nucleotides x and y, as follows:

$$p(x, y) = \begin{cases} -4 & \text{if } x = \text{gap} \vee y = \text{gap}, \\ 4 & \text{if } x = y, \\ -4 & \text{if } x \neq y. \end{cases}$$

This scoring system considers three key scenarios: *(i) gap penalty*, assigning a score of -4 when either x or y is a gap, *(ii) matching score*, assigning 4 when x and y are identical, and *(iii) mismatch penalty*, assigning -4 when x and y differ. The objective of the MSA problem is to maximize the SP score:

$$\max(SP(CA)), \quad CA \in \{CA_i\}_{i=1}^{\infty}$$

Another commonly used evaluation indicator is the CS score. For the ith column in the aligned area described above, the score $C_i = 1$ if all residues are the same; otherwise, $C_i = 0$. The CS Score is:

$$CS = \sum_{i_1}^{M} \frac{C_i}{M}$$

IV. THE RL BASE AGENT

Our model is based on a GA that intelligently and locally orchestrates a *RL base model*. In this specific work, we exploit `DPAMSA` as base RL model, which tackles the problem through a structured process consisting of several key steps: *(i)* the *agent*, which interacts with the environment, adjusting its action strategy in order to optimize decision-making over time; *(ii) sequence alignment*, where sequences are aligned column by column, with the sequence state being updated after each column's alignment, *(iii) state update*, where the sequence state transitions from S_i to S_{i+1}, embedding the unaligned sequence in S_i into a vector that is processed by the deep Q-network, which outputs an action value dictating the insertion of gaps; *(iv) state embedding*, where all unaligned sequences are concatenated into a single string and converted into an integer vector; after each column is aligned, the vector length is reduced, and padding is applied to maintain a fixed length of $n \times (L + 1)$, where n is the number of sequences and L is the maximum sequence length; *(v) gap insertion*, where the

action value is binary encoded and reversed (e.g., with action *110*, gaps are inserted into the first two columns); *(vi)* the *episode*, which tracks the sequence of states and actions from the *initial state* s_o to the *terminal state* s_T:

$$episode = s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \ldots \xrightarrow{a_{T-2}} s_{T-1} \xrightarrow{a_{T-1}} s_T$$

where s_t is the *state* at time-step t, and a_t is the action in s_t.

In DPAMSA, the *self-attention mechanism* focus on critical sequence areas by calculating weighted relevance scores between nucleotides, using vectors for *query* Q, *key* K, and *value* V:

$$attention(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{|K|}}\right)V$$

To retain nucleotide order, *positional encoding* adds sine and cosine-based values to each position, preserving positional information crucial for alignment accuracy. Extracted features from self-attention and positional encoding are processed through a *multi-layer perceptron*, which computes *Q-values* to guide alignment actions. This RL process uses the *Bellman equation* to optimize decisions, with Q-values updated to maximize expected rewards:

$$Q(s_t, a_t) = r_t + \gamma \max_{a'} Q(s_{t+1}, a')$$

where r_t is the reward, and γ the discount factor. Training employs both an evaluation and a periodically synchronized target Q-network to ensure stable updates. The model aims to minimize prediction error using a *loss function*:

$$\text{Loss} = (y_t - Q(s_t, a_t))^2$$

This setup enables the gradual improvement in alignment quality by continuously optimizing Q-values across training episodes.

V. Dataset simulation framework

The dataset generation process aims to reproduce a biological scenario in which multiple DNA sequences share *conserved regions* (unchanged nucleotide blocks), while *non-conserved segments* undergo mutations and insertion/deletion events (gaps). The objective is to construct datasets that retain a specific degree of similarity across sequences while introducing biologically realistic variations.

The dataset simulation follows a structured pipeline:

1) *Conserved blocks.* A set of DNA fragments is generated to represent conserved regions that remain identical across all sequences, mimicking conserved genomic segments.
2) *Conserved blocks insertion.* Each sequence is initialized as a fully random DNA strand where the conserved blocks are then inserted at identical positions in all sequences to introduce a biologically plausible alignment structure.
3) *Mutation and gap insertion.* Mutations are inserted at random positions in non-conserved regions to simulate genetic drift and point mutations; gaps (denoted by $-$) are inserted at random positions outside the conserved blocks to mimic indels (insertion/deletion events).
4) *Similarity evaluation.* The dataset's alignment similarity is quantitatively evaluated using the SP score to reward matched nucleotides while penalizing mismatches and gaps.
5) *Dataset Validation and Acceptance.* The dataset is accepted only if its calculated alignment score meets a predefined similarity threshold. Otherwise, the generation process is iterated until a valid dataset is obtained.
6) *FASTA storage.* Once generated, the validated sequences are stored in FASTA format for further processing and testing.

This dataset generation strategy ensures that the evaluation framework effectively balances *alignment complexity* and *computational tractability*, allowing for a systematic comparison between the proposed method and baseline approaches. To establish an optimal experimental setup, the synthetic datasets were used to *fine-tune* the following key hyper-parameters of the proposed approach: *(i) GA hyper-parameters*, such as, *population size*, *number of generations*, *mutation rate*, *crossover rate*; *(ii) Board size*, i.e., the size of the sequence alignment sub-problems used for training the DRL agent. The hyper-parameter optimization process involved evaluating multiple configurations and selecting the one that maximized *alignment accuracy* while minimizing *computational overhead*.

To rigorously evaluate the proposed method, we generated multiple synthetic datasets by systematically varying key parameters that influence sequence similarity and alignment complexity. The dataset generation was controlled by the following factors:

- *Conserved region length.* The length of conserved blocks was adjusted to simulate varying degrees of sequence conservation, ranging from highly conserved segments

(longer blocks) to more divergent scenarios (shorter blocks).

- *Mutation rate.* Different mutation rates were applied to non-conserved regions to represent varying evolutionary pressures, introducing point mutations at rates of 5%, 10%, and 20% per sequence.
- *Gap probability.* The frequency and length of insertion/deletion events (gaps) were modified to reflect real genomic variations, with gap insertion probabilities set at 2%, 5%, and 10%.
- *Sequence length and dataset size.* Experiments were conducted on datasets with sequence lengths of 30, 60, and 150 base pairs, each with different numbers of sequences per alignment, ranging from 3 to 12.

For each parameter configuration, the generated datasets were evaluated in terms of *alignment complexity* (measured via the SP score distribution) and *computational feasibility* (execution time and memory consumption). The performance of `GA-DPAMSA` was tested across these datasets, allowing us to systematically assess the method's robustness under different levels of sequence variability.

After extensive testing, we identified an optimal setting that maximized alignment accuracy while maintaining computational efficiency. The final configuration adopted the following parameter values: conserved block length covering 50% of the sequence, a mutation rate of 10%, a gap probability of 5%, and a sequence length of 60 base pairs with six sequences per dataset. This setting provided a balanced trade-off between sequence variability and alignment stability, ensuring that the experimental framework adequately reflects real-world alignment challenges while remaining computationally tractable.

The source code and further details are available online[4].

VI. SETTINGS

Dataset	# sub-datasets	# sequences	Avg seq. length
Dataset1	*50*	*3*	*30*
Dataset2	*25*	*6*	*30*
Dataset3	*9*	*3*	*60*
Dataset3	*50*	*12*	*150*

TABLE A1
SYNTHETIC DATASETS GENERATED FOR THE EXPERIMENTS.

Hyperparameter	Value
batch_size	*128*
replay_memory_size	*1000*
number_episode	*6000 (max)*
update_iteration_target_Q_network	*128*

TABLE A2
DPAMSA HYPERPARAMETERS.

Parameter	Description	Value
N	*size of the population*	*150*
max_gen	*# evolutionary epochs*	*1000*
tourn	*tournament selection %*	*25*
c_prob	*crossover probability*	*50*
m_prob	*mutation probability*	*85*
e_prob	*elitism probability*	*40*
%_select	*pareto front retain %*	*45*

TABLE A3
BEST SETTING FOR *GA-DPAMSA*.

Name	Input Size	Output Size	Activation Function
Linear 1	$d_{model} \times n \times L$	*1028*	*LeakyReLU*
Linear 2	*1028*	*512*	*LeakyReLU*
Linear 3	*512*	$2^n - 1$	*Tanh*

TABLE A4
THE PARAMETERS OF MULTI-LAYER PERCEPTRON.

[4]https://github.com/FLaTNNBio/GA-DPAMSA

VII. Experimental results tables

Tool	Mean	Median	SD	Min	Max
random-horizontal	*1868.70*	*1881.02*	*208.25*	*1752.10*	*2039.81*
random-vertical	*2236.11*	*2292.78*	*116.81*	*2106.48*	*2243.52*
worst fitted-horizontal	*2305.74*	*2373.70*	*116.67*	*2125.75*	*2272.04*
worst_fitted-vertical	*2309.94*	*2396.21*	*121.61*	*2164.15*	*2291.12*

TABLE A5

GA-DPAMSA *mutation-crossover* CONFIGURATIONS: DETAILED VALUES FOR THE MEAN, MEDIAN, STANDARD DEVIATION, MINIMUM, AND MAXIMUM OF THE SP SCORE.

Dataset	Board	avg train time (h)	avg GA time (s)
dataset1	*3 x 30*	*0.21*	*25.65*
dataset2	*6 x 30*	*0.30*	*26.36*
dataset3	*6 x 60*	*1.26*	*26.70*
dataset4	*12 x 150*	*25.34*	*28.72*

TABLE A6

AVERAGE *training time* OF DPAMSA ON EACH DATASET

Tool	Mean	Median	SD	Min	Max
GA-DPAMSA	*170.25*	*172.00*	*38.88*	*146.00*	*206.00*
ClustalΩ	*174.24*	*177.00*	*44.96*	*141.60*	*213.12*
MSAProbs	*228.48*	*248.10*	*39.17*	*178.10*	*247.31*
ClustalW	*227.76*	*238.06*	*39.24*	*177.65*	*238.11*
DPAMSA	*138.16*	*139.00*	*38.94*	*112.03*	*161.34*

TABLE A7

DETAILS OF THE MEAN, MEDIAN, STANDARD DEVIATION, MAXIMUM, AND MINIMUM OF THE SP SCORE ACROSS THE 50 SUB-DATASETS IN THE DATASET1 OF SIZE *3x30bp*.

Tool	Mean	Median	SD	Min	Max
GA-DPAMSA (on 3x30)	*1572.30*	*1580.24*	*64.47*	*1557.31*	*1604.36*
ClustalW	*1253.34*	*1312.11*	*1072.65*	*272.43*	*1485.14*
MSAProbs	*1493.30*	*1512.42*	*192.75*	*1345.49*	*1602.40*
DPAMSA (on 6x30)	*1376.18*	*1481.22*	*123.21*	*1341.32*	*1432.12*
ClustalΩ	*1487.10*	*1512.00*	*235.12*	*1301.02*	*1612.27*

TABLE A8

DETAILS OF THE MEAN, MEDIAN, STANDARD DEVIATION, MAXIMUM, AND MINIMUM OF THE SP SCORE ACROSS THE 50 SUB-DATASETS IN THE DATASET2 OF SIZE 6X30BP.

Tool	Mean	Median	SD	Min	Max
GA-DPAMSA (on 3x30)	*1640.73*	*1652.66*	*108.42*	*1595.56*	*1809.47*
ClustalW	*1042.32*	*1116.04*	*373.33*	*865.68*	*1391.72*
MSAProbs	*1131.03*	*1195.56*	*328.48*	*916.76*	*1489.94*
DPAMSA (on 6x60)	*1331.03*	*1395.56*	*1278.48*	*1402.10*	*1689.94*
ClustalΩ	*1763.76*	*1789.94*	*1044.20*	*1455.56*	*1843.20*

TABLE A9

DETAILS OF THE MEAN, MEDIAN, STANDARD DEVIATION, MAXIMUM, AND MINIMUM OF THE SP SCORE ACROSS THE 50 SUB-DATASETS IN THE DATASET3 OF SIZE 6X60BP.

Tool	Mean	Median	SD	Min	Max
GA-DPAMSA (on 3x30)	*8719.45*	*8724.71*	*48.63*	*8647.38*	*8714.04*
ClustalΩ	*8362.45*	*8423.05*	*198.75*	*8123.92*	*8587.27*
MSAProbs	*8097.31*	*8112.67*	*218.66*	*7854.33*	*8324.33*
ClustalW	*8394.31*	*8398.67*	*168.66*	*8167.79*	*8504.33*
DPAMSA	*8194.11*	*8208.72*	*143.13*	*8095.71*	*8301.13*

TABLE A10

DETAILS OF THE MEAN, MEDIAN, STANDARD DEVIATION, MAXIMUM, AND MINIMUM OF THE SP SCORE ACROSS THE 50 SUB-DATASETS IN THE DATASET4 OF SIZE 12X150BP.