RESEARCH ARTICLE

VOL. 45 | NO. 2 | **MAY-AUGUST 2024** | PP 62-77

Revista Mexicana de Ingeniería Biomédica

dx.doi.org/10.17488/RMIB.45.2.4

E-LOCATION ID: 1419

Enhancing Multiple Sequence Alignment with Genetic Algorithms: A Bioinformatics Approach in Biomedical Engineering

Una Mejora del Alineamiento Múltiple de Secuencias con Algoritmos Genéticos: Un Enfoque de Bioinformática en la Ingeniería Biomédica

Ernesto Rios-Willars [1], *Jennifer Vélez-Segura* [2], *María Magdalena Delabra-Salinas* [3]

[1] Universidad Autónoma de Coahuila, Facultad de Sistemas Unidad Saltillo - México

[2] Universidad Nacional de Colombia - Colombia

[3] Universidad Autónoma de Coahuila, Facultad de Enfermería Unidad Saltillo - México

ABSTRACT

This study aimed to create a genetic information processing technique for the problem of multiple alignment of genetic sequences in bioinformatics. The objective was to take advantage of the computer hardware's capabilities and analyze the results obtained regarding quality, processing time, and the number of evaluated functions. The methodology was based on developing a genetic algorithm in Java, which resulted in four different versions: Gp1,Gp2,Gp3,and Gp4. A set of genetic sequences were processed, and the results were evaluated by analyzing numerical behavior profiles. The research found that algorithms that maintained diversity in the population produced better quality solutions, and parallel processing reduced processing time. It was observed that the time required to perform the process decreased, according to the generated performance profile. The study concluded that conventional computer equipment can produce excellent results when processing genetic information if algorithms are optimized to exploit hardware resources. The computational effort of the hardware used is directly related to the number of evaluated functions. Additionally, the comparison method based on the determination of the performance profile is highlighted as a strategy for comparing the algorithm results in different metrics of interest, which can guide the development of more efficient genetic information processing techniques.

KEYWORDS: bioinformatics, genetic algorithm, multiple sequence alignment, msa

RESUMEN

El objetivo del presente trabajo es desarrollar una estrategia de procesamiento de información genética para el problema del alineamiento múltiple de secuencias en el área de bioinformática con el propósito de explotar la potencia del hardware y analizar los resultados en términos de calidad de las soluciones obtenidas, así como el tiempo requerido por el sistema de cómputo para realizar el proceso y la determinación del número de funciones evaluadas. Los procedimientos y metodología para dicho trabajo fueron basados en la programación de un Algoritmo Genético en lenguaje Java, del que sucesivamente se obtuvieron cuatro diferentes versiones denominadas Gp1, Gp2, Gp3 y Gp4. Con esto se procesó un conjunto de secuencias genéticas y se evaluaron los resultados a través de la determinación de los perfiles de comportamiento numérico. Entre los hallazgos se encuentra que la diversidad en la población y el procesamiento paralelo tienen influencia en los resultados. A partir del perfil de comportamiento numérico se observa una reducción en el tiempo requerido para realizar el proceso. Se concluye que a) el equipo de cómputo convencional puede generar muy buenos resultados al procesar información genética cuando los algoritmos son preparados para aprovechar al máximo los recursos de hardware. Esto puede guiar al desarrollo de estrategias más eficientes de procesamiento de información genética.

PALABRAS CLAVE: bioinformática, algoritmo genético, alineamiento múltiple de secuencias, msa

Corresponding author

TO: **Ernesto Rios-Willars**

INSTITUTION: **Universidad Autónoma de Coahuila, Facultad de Sistemas Unidad Saltillo - México**

ADDRESS: **Ciudad Universitaria, Fundadores Km 13, Zona Centro, 25350 Arteaga, Coahuila, México.**

EMAIL: **riose@uadec.edu.mx**

Received:

12 February 2024

Accepted:

1 May 2024

INTRODUCTION

Deoxyribonucleic acid, commonly known as DNA, is a large and complex molecule located in the nucleus of cells in all living organisms. It takes the shape of a double helix structure, where the genetic information unique to each individual and species is stored. All life is based on a code written in DNA molecule, the common denominator among living beings [1]. DNA can exist in a single-stranded form or as higher-order structures, including the canonical double helix and noncanonical duplex, triplex, and quadruplex species [2]. In a DNA molecule, each sugar in the two strands is attached to one of four nitrogenous bases - Adenine (A), Guanine (G), Cytosine (C), and Thymine (T). These bases can bond through hydrogen bonds, and their bonding possibilities differ. Specifically, DNA's double helix structure is attributed to hydrogen bonds between A-T (two) and C-G (three) base pairs[3].

The number of hydrogen bonds determines the specific pairing between nitrogenous bases, but this structure is prone to errors and can sometimes be repaired [4][5].

According to the NCBI database [6], the number of records has doubled approximately every 18 months since 1982[7]. From a computer science perspective, advanced metaheuristic strategies are required to handle such information[8][9].

We developed a genetic algorithm to perform multiple genetic sequence alignments in this work. The objective is to explore the improvement from different parametrization schemas (Gp1, Gp2, Gp3, and Gp4) for the algorithm`s performance in terms of a) the solution quality (fitness) achieved by the algorithm and b) the computational effort ($N\hat{F}E$) with the count of evaluated functions. Since the parameter adjustment is made in the number of processing cores and mutation level, the secondary objective is to describe the impact of such adjustments on the processing time between the four algorithm schemas.

In the field of genetics, the parallelization strategy of processing information has shown significant improvement in the results, especially in reducing the computational time required to solve various problems[10][11][12]. As a validation method, the numerical performance profile measures the relative efficiency of the algorithm in solving the set of problems[13].

Bioinformatics software testing is often complex due to the difficulty in verifying the correctness of output and effectively generating failure-revealing test cases[14]. In this work, we developed four strategies based on a genetic algorithm for the multiple sequence alignment problem and compared their performance to each other. Multiple sequence alignment (MSA) is essential in bioinformatics, aiding in phylogenetic, protein, and genomic analysis, but faces challenges in increasing alignment accuracy[15].

There are several algorithm strategies in the state of the art, such as the "Progressive alignment," which has been one of the most used methods since the 1980s [16]. In addition to improvements in algorithm design, modern hardware technology has increased processing speed and enabled more simultaneous alignment sequences[17]. Furthermore, integrations based on framework processing like Apache Spark can produce promising results and new perspectives [18].

Regarding the processing algorithms, researchers have developed exact, progressive, and iterative algorithms to

address the Multiple Sequence Alignment (MSA) problem. Exact algorithms exhaustively search in the space of solutions and calculate the global optimum in sequences of relatively small lengths. However, they cannot guarantee finding the best solution in arrangements of hundreds or thousands of bases. Blast is an example of this type of algorithm [19]. On the other hand, progressive algorithms work by iteratively improving a sequence alignment by adding new sequences. The process involves gradually building upon an initial alignment until an optimal solution is found [20][21]. Dynamic programming algorithms are commonly used to implement strategies that explore solution spaces using a substitution matrix [22]. One example is the ClustalW algorithm [23]. However, errors at the beginning of an alignment can propagate to the rest of the operation, which is a drawback of these algorithms. Furthermore, another strategy to tackle bigger MSA problems is to iteratively perform partial alignments [24][25].

On the other hand, Genetic algorithms are iterative metaheuristic strategies based on bioinspiration. A thorough analysis of the algorithms used in this context can be found in [26]. Due to the metaheuristic origin of Genetic Algorithms, which does not guarantee achieving the optimal solution, we assume no comparison between metaheuristics and/or stochastic strategies vs. progressive or deterministic strategies. However, metaheuristics permit working with more extensive problem instances.

This work presents a bioinformatics breakthrough and builds on fundamentals established by previous papers, such as [27], highlighting the importance of hardware acceleration in DNA sequence alignment. The utility of parallel processing was established by [28], which improves the process at the bit level. This work presents an innovative methodology that combines multiple techniques, including genetic algorithms, to enhance multiple sequence alignment. The methodology also employs the algorithms' performance profile as an efficient comparison method.

Due to the increasing scientific research in medicine-based informatics, the genetic algorithm has been tested using a set of genetic sequences from a group of vegetables of particular interest in medicine, such as *Euphorbia pulcherrima*, which has Anti-Inflammatory, Analgesic, Sedative, and Muscle-Relaxant Activities [29]. Preventing health complications is a significant motivation for studying the properties of vegetables.

MATERIALS AND METHODS

The computational processes in this study were meticulously developed on four identical personal computers (PCs) with a Microsoft Windows operating system. These PCs were equipped with 8 GB of RAM and an Intel core *i5* processor, ensuring the study's robustness and reliability. The algorithms were programmed in Java with Netbeans IDE, a widely recognized and trusted platform for software development.

Genetic Algorithm

In this study, a genetic algorithm (GA) is used to simulate the natural adaptation process of living beings. This computer-based representation was initially proposed by [30]. These are some instances of how GA's are used:[31][32][33][34]. While most search algorithms operate on a single solution, a GA operates on a population of solutions. The classical GA codes a given problem, generating a population of potential solutions. This method involves crossing, leading to mutation, and discovering new and improved solutions. This statement is founded on Darwin's evolutionary theory.

Sequence Alignment

When there are two sequences (m,n) that can accommodate a limited number of gap insertions, the number of align-

ments increases as the length of the sequences increases. The number of potential alignments for a pair of sequences m and n, with 'k' insertions, can be calculated using Equation (1) as stated by Waterman in 1995:

$$f = \sum_{k=0}^{\min(m,n)} \frac{(m+n-k)!}{k!\,(m-k)!\,(n-k)!} \tag{1}$$

When attempting to solve the multiple sequence alignment problem using brute force, the problem becomes NP-complete. In contrast, dynamic programming has a complexity of $O(LN)$, where L is sequence length, and N is the number of sequences. Researchers aim to enhance MSA efficiency via heuristic and metaheuristic approaches and parallel implementations to reduce computational costs [35].

Sum-of-pairs multiple alignments: The given strings, represented by $S1$ to Sk, are used to create multiple alignment. This alignment consists of strings $T_1 \ldots T_K$, which are the same length as the original strings. The new strings are made by inserting spacing symbols at specific positions in the original strings. However, only columns full of spacing symbols are not allowed.

Given a multiple alignment $T_1 \ldots T_K$ of length m and a scoring function σ for pairs of characters ($\sigma(a,b)$) and pairs of character and spacing symbol ($\sigma(a,$-) or σ(-,b)), we define its sum-of-pairs score as described in Equation (2):

$$\sigma(\mathrm{T}_1, \ldots, \mathrm{T}_\mathrm{k}) = \sum_{i<j}\sum_{p=1}^{m} \sigma\left(\mathrm{T}_\mathrm{i}(p), \mathrm{T}_\mathrm{j}(p)\right) = \sum_{i<j} \sigma * \left(\mathrm{T}_\mathrm{i}, \mathrm{T}_\mathrm{j}\right). \tag{2}$$

Compute a multiple alignment of strings S1 to Sk with a given scoring function sigma to find the maximum sum-of-pairs score.

Problem Codification: Text or binary sequences are commonly used for multiple sequence alignment. These sequences include array-like data structures, making handling characters and reordering positions easier. As a result, the computational effort required by the processing machine is reduced. In this work, each alignment matrix A represents an individual, and each matrix position contains a single character (a). Equation (3) shows the matrix positions where a nucleotide or a gap can be found.

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \ldots & a_{1n} \\ a_{21} & a_{22} & a_{32} & \ldots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \ldots & a_{3n} \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ a_{m1} & a_{m2} & a_{m3} & \ldots & a_{mn} \end{pmatrix} \tag{3}$$

Each row in the alignment represents a sequence of the set to be aligned.

Proposed genetic algorithm: GAAPd

The algorithm starts with an Initial Population (*Pi*) of 500 individuals, which is later reduced to a fixed number of 100 individuals (*Pt*) between 100 generations. This helps maintain stability in the algorithm and prevents excessive computational resource usage. The individuals in the initial population are generated by random potential solutions to the problem.

Individuals are evaluated using a parallel process and the widely used Blosum evaluation matrix [36]. This matrix helps measure the quality of the in-turn alignment. There are ongoing efforts to parallelize genetic algorithms either partially or entirely, and these efforts have demonstrated meaningful results in their respective fields of application [37][38][39][40].

To speed up the evaluation process, we divide the workload and assign it to each evaluator instance, which is a component of the evaluation class designed for this process. We avoid double evaluations of union regions to ensure synchronicity among evaluators.

After completing the task, evaluators assign each population member a fitness value (*fi*) used on their partial results. Figure 1 illustrates the evaluation process described in this work.

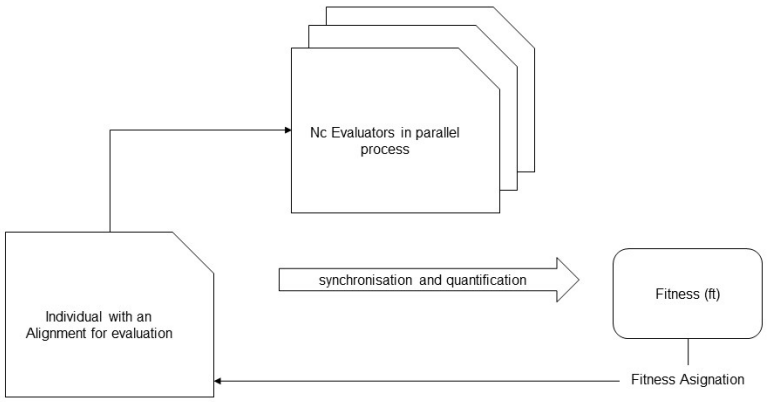

FIGURE 1. The process of evaluation. Nc refers to the number of evaluator instances created to evaluate everyone. Once the fitness value is assigned, the process is repeated for the following individuals in the population.

According to theory, there is a direct correlation between the number of cores and the reduction of processing time ($\hat{T}$), stated in Equation (4). However, other intrinsic processes consume resources, such as synchronization and quantification.

$$\hat{T} = N\hat{F}E/Nc \tag{4}$$

Where $N\hat{F}E$ is the number of evaluated functions.

During the crossover process, individuals are randomly selected from the best performers of the evaluation process in the *Pt* set. In this context, it implies that only the most outstanding individuals proceed to the crossover phase. Their probability of being assigned to another individual is

$Pb = 1/Pt$, equal for all.

A random double-point cut is made (*CPA,CPB*), and each crossover produces two children integrated into the population. The positions of *CPA* and *CPB* in the alignment matrix are also random.

After making the necessary cuts for both individuals, three parts of the father (*P1F,P2F,and P3F*) and three of the mothers (*P1M,P2M,and P3M*) are obtained.

Child 1 and Child 2 incorporate all six elements shown in Figure 2.

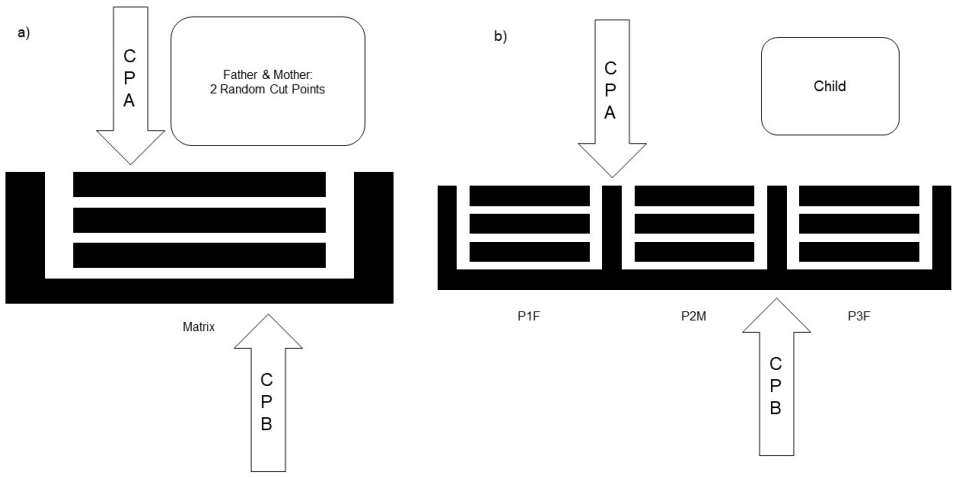

FIGURE 2. The crossover processes. In this process, two individuals produce three parts each. The resulting child has one part from the mother (P2M) and two from the father (P1F,P3F). Another child has one part from the mother and one from the father.

The mutation process occurs for progenies [41], which is achieved by inserting or deleting gaps using the hyphen character "-" in various positions within the matrix. This process helps to improve the alignment further. The probability of an individual moving to the mutation process is Pm = 0.2, in which changes are randomly applied to the alignment matrix based on the Pm value.

Various mutation techniques have been documented in the literature, each relying on a specific mutation probability [41][42][43]. We developed four forms of mutation described as follows:

a) One of the sequences is randomly selected for mutation, where a maximum of 200 gaps can be inserted at the beginning.
b) A random point in the matrix is selected for gap insertion. The number of spaces added is randomly determined, up to 300.
c) Insertion of a gap column at a random point. A gap column is inserted at some position in the alignment matrix.
d) Random gaps are deleted. Up to 100 gaps are removed, and the number of gaps removed is randomly determined within a range of 1 to 100.

Algorithm Pseudocode and GitHub Access

The pseudocode of the proposed algorithm is presented as follows. Let *Nc* be the number of processing cores and Nm the mutation level as a repeating mutation process over a given matrix.

1) Population Initialisation (*P*)
2) Population Evaluation (Blosum Matrix)
    a. Distribute partial work for cores (Nc) and evaluator Instances.
    b. Execute parallel evaluation process.
    c. Once finished, evaluation instances compute the overall fitness value.
3) Worst Individuals Elimination.
4) Crossover.
5) Repeat progeny mutation (*Nm*) Times.
6) Integrate progeny into *P*.

The GitHub project is available for further analysis at https://github.com/riosew/GAAPd-Software.git.

The algorithm is presented as a set of corresponding classes. The different versions designated as Gp1, Gp2, Gp3, and Gp4 can be produced by changing the number of cores (*Nc*) for the parallel process and the mutation level (*Nm*) to change the mutation cycles. Thus, it can be executed with the different *Nc* and *Nm* values in Table 1. It initiates the process with a basic visual interface, where the user can change the indicated values.

TABLE 1. **GAAPd versions.**

Version	Nc	Nm
Gp1	1	1
Gp2	4	4
Gp3	1	4
Gp4	4	1

Time Complexity Analysis Big O

The evaluation process is divided into the following parts.

Creation of evaluators: This loop runs several times equal to the number of available cores. Inside this loop, another loop runs several times equal to the size of the alignment matrix. Therefore, this part of the algorithm has a complexity of $O(n*m)$, where n is the number of cores and m is the size of the matrix.

Waiting for all evaluators: This loop runs until all evaluators have finished. In the worst case, this could take time proportional to the evaluator, which takes the longest to complete the work.

Traversal of evaluators adding results: This loop runs several times equal to the number of evaluators (equal to the number of cores). Therefore, this part of the algorithm has a complexity of $O(n)$, and the total complexity of the evaluation process is $O(n*m)$, where n is the number of cores and m is the size of the matrix. However, the complexity analysis of the nested cycles in the micro evaluator instance (*mE*) is worth noticing. The mE instances are as many instances as the core Nc value. The complexity of the *mE* process is $O(nmp^2)$. Since n and m are the matrix size, and p is the vector size in the Blosum evaluation matrix, which performs a traversing process for n and m. Finally, in big(O) analysis, the worst case of operation establishes the total algorithm complexity.

Experimental Method

This paper's experiment describes a sequence of algorithms and the Performance Profile.

Set of Sequences

The dataset has been expanded by integrating 13 sequence groups, described in Table 2. It is available for future reference on the Galaxy platform [44] at https://usegalaxy.org/u/rioswillars/h/dataset-for-genetic-algorithm. Each group comprises N sequences of varying lengths.

TABLE 2. **Test Sequence Sets.**

Set	Name	*N*
1	*Agave victoriae reginae*	38
2	*Aristida purpurea*	38
3	*Bellucia grossularioides*	42
4	*Bursera simaruba*	35
5	*Cnidoscolus urens*	35
6	*Cordia alliodora*	28
7	*Curatella Americana*	33
8	*Cydista diversifolia*	29
9	*Echinocactus platyacanthus*	38
10	*Ephiphylum hookeri*	31
11	*Erythrina herbacea*	35
12	*Euphorbia pulcherrima*	23
13	*Gossypium arboretum*	29

Algorithms

Four versions of the GAAPd algorithm were created to evaluate the number of evaluated functions ($N\hat{F}E$), and quality of alignment achieved (*Fitness*). The computational processing parameters were adjusted at varying levels and the impact of processing time due to the adjustments was registered for comparison between the four GAAPd versions. The parametters for adjustment are: the number of core decomposition (Nc), and mutation cycles for the children in each generation (Nm). These versions are named Gp1, Gp2, Gp3, and Gp4; their characteristics are listed in Table 1.

Determination of numerical performance profiles

A comparative analysis was conducted to determine the robustness and efficiency of four methods: Gp1, Gp2, Gp3, and Gp4. The study used numerical performance profiles, which define the cumulative distribution function for a numerical performance profile. The metrics of interest were the algorithm's execution time, its ability to reach the global optimum in the objective function, and the number of evaluations performed during the calculation sequence. These factors were compared to analyze the performance and convergence of the optimization method.

This paper aims to compare four different methods based on three metrics. The first metric is the relative distance

between the optimal solution found by the optimization method and the known global optimal solution obtained by the ClustalW algorithm. This metric measures the robustness of the process, i.e., its ability to locate the global optimum of the objective function. The second and third metrics are the number of function evaluations $\hat{NFE}$ required during the processing and the time taken ($\hat{T}$) for process.

To evaluate specific metrics, we considered four optimization methods (ns = 4) and 13 problems or sets of sequences (np = 13). Each case study was solved 30 times, with random initial solutions and different random alignments. The stop condition was stabilized within the 100-generation limit. However, since the stop criterion is based on the generation number, a sensitive tolerance of 1.0E-06 is assumed for the value of the objective function since the optimal solution is uncertain. It was set as the convergence criterion for the four methods for all four optimization methods and sequence sets, the values of $t_{p,s}$ were calculated using the results of the 30 calculations and the following expressions (Equation 5).

$$t_{p,s}^{d} = \frac{\dot{f}_{obj} - \hat{f}_{obj}}{\dot{f}_{obj} - f_{obj}^{w}} \qquad t_{p,s}^{NFE} = \hat{NFE} \quad t_{p,s}^{T} = \hat{T} \tag{5}$$

Being $\hat{f}_{obj}$ the average value of the target function calculated by the optimization method, $\dot{f}_{obj}$ is the global optimum of the objective function, f_{obj} is the maximum value for the objective function found within the calculation sequence, $\hat{NFE}$ and $\hat{T}$ are respectively the average value of the number of functions evaluated and the time to reach the convergence of the optimization method. It is important to note that the values of $\hat{f}_{obj}$, $\hat{NFE}$ y $\hat{T}$ were determined using the 30 numerical experiments performed for the case study. As established in the literature [45], average values are often used for behavioral metrics to describe method performance. Based on the above, this study also employs such an approach. On the other hand, for all three metrics, the numerical performance profile rate $r_{p,s}$ is defined as in Equation (6):

$$r_{p,s} = \frac{t_{p,s}}{min\{t_{p,s}: s \in S\}} \tag{6}$$

Where S corresponds to the set of optimization methods analyzed. This method assigns a value of 1 to the algorithm that performs the best in each problem. Finally, the cumulative probability rate $\rho_s(\tau)$ for the optimization strategy S and the metric in question is defined as in the Equation (7):

$$\rho_s(\tau) = \frac{1}{n_p} cantidad\{p \in P: r_{p,s} \leq \tau\} \tag{7}$$

Where τ is a factor that is defined in $(1, \infty)$. In the graph of the performance profile, for example, the graph of ρ_s *versus* τ, compares the relative performance between optimization methods for the problem group. So far, performance profiles have been used by [46] to compare bioinspired algorithms and *benchmark* functions. However, this concept has not been used to compare the methods described in the MSA problem.

RESULTS AND DISCUSSION

The results of the fitness, $\hat{NFE}$, and time metrics are summarized and described below. As stated, the fitness value represents the algorithm's ability to reach better results, iteratively searching for solutions for a given problem. In

this case, there are 13 problems for the performance evaluation; the problem count is presented in Table 3 as a C value. Furthermore, as described before, each problem C represents a set of sequences to be aligned in the multiple sequence alignment problem for the algorithm, and each algorithm *(Gp1, Gp2, Gp3,* and *Gp4)* ran 30 times for the calculated performance mean values. Also, the table presents the mean values for the fitness metric on each algorithm and each of the 13 multiple alignment problems.

TABLE 3. Mean values for the Fitness metric of the Gp1, Gp2, Gp3 and Gp4 algorithms for the 13 multiple sequence alignment problems.

C	Gp1	Gp2	Gp3	Gp4
1	62723.7	62385.6	63085.2	63105.4
2	59747.1	60072.8	59776.5	59605.0
3	57785.0	58536.2	58413.7	57109.7
4	52653.2	53250.0	53497.3	53526.8
5	16934.0	16463.8	16917.7	17270.7
6	-4932.2	-4724.0	-5529.0	-4903.8
7	-43993.1	-44014.3	-43848.8	-43909.3
8	-33991.1	-33350.3	-33782.7	-33775.6
9	54515.2	54405.9	54311.7	53919.9
10	-99067.2	-98881.0	-98633.5	-98737.2
11	52933.6	53749.8	52805.7	52540.1
12	-4082.1	-4463.7	-4449.5	-4134.2
13	9190.8	8423.8	8997.1	8699.8

The fitness metric shows promising performance results for the Gp2 and Gp3 algorithms since the objective is to increment the most possible fitness value. Also, the negative values describe complicated sequences for the alignment problem for all four algorithms.

For the number of evaluated functions ($N\hat{F}E$) metric, Table 4 shows the better performance of the Gp3 algorithm in problems 1 and 8. Gp1 performed better in sequence set number 2. Furthermore, all four algorithms present similar $N\hat{F}E$ results in sequence set number 12. This shows a significant difference in the algorithms' performance in terms of the different strategies they represent.

TABLE 4. The $N\hat{F}E$ values for each of the four algorithms on the alignment of the 13 sets of sequences. The results show substantial differences between the four algorithms.

C	Gp1	Gp2	Gp3	Gp4
1	690.867	684.067	679.600	688.733
2	680.400	693.033	686.433	685.100
3	684.333	690.700	686.233	686.600
4	686.033	693.167	688.867	687.900
5	685.300	687.167	683.233	681.033
6	684.667	689.867	683.633	684.167
7	692.133	683.167	690.233	693.933
8	682.367	690.400	679.600	689.867
9	686.200	689.400	684.533	681.067
10	693.400	685.667	686.733	686.733
11	685.233	680.767	693.333	690.700
12	682.733	683.367	683.867	683.100
13	685.933	681.500	683.967	683.600

The time it takes for the algorithms to process the stated problems is a commonly used metric for comparing the strategies' performance. Table 5 shows the performance in terms of the required time for each of the four algorithms to process the 13 multiple alignment problems. It is worth noting that $Gp3$ took more time than the other algorithms to process the alignment problems. Furthermore, $Gp4$ showed better performance than the different algorithms. The efficiency of an algorithm is not determined by the time it takes to run. Still, in this case, the time comparison helps show the main difference between the algorithms: the number of nuclei for the hardware process.

TABLE 5. The comparative of the mean values in the metric Time for the four algorithms in the 13 sets of sequences for the multiple alignment problem.

C	Gp1	Gp2	Gp3	Gp4
1	4409.033	3943.900	5065.433	3381.000
2	4118.767	3554.400	4849.733	2964.267
3	4582.700	3944.633	5352.100	3268.467
4	3779.267	3289.667	4397.400	2727.467
5	3542.733	3141.700	4128.300	2598.000
6	2649.367	2543.267	3106.400	2061.000
7	3439.833	3138.667	4025.133	2667.833
8	2929.867	2753.867	3396.033	2300.000
9	4546.700	3874.767	5232.233	3361.800
10	5006.133	4514.933	5748.400	3919.100
11	3866.033	3288.933	4525.900	2901.100
12	2501.500	2321.367	2912.000	2032.300
13	4440.800	3894.400	5086.733	3478.067

We present a Microsoft Excel file named results in the provided github link for a detailed description of the tables above.

Figure 3 displays the performance profile for the metric $t^d_{p,s}$. This metric is associated with the ability of the algorithm to reach a better solution to a given problem. The figure 3 shows the optimization method's ability to approach the global optimum in the problems considered. $Gp1$ and $Gp3$ maintain remarkable performance and approach convergence. This can be attributed to the fact that they perform single-core processing with maximum and minimum mutation levels of 1 and 4, respectively. It is worth noting that this characteristic is in $Gp1$ and $Gp3$ and contributes to the state of the art of genetic programming, particularly regarding the mutation level for the individuals in the Genetic Algorithms population. In other words, this means that the medium-level mutation had no fitness improvement benefits for this case study.

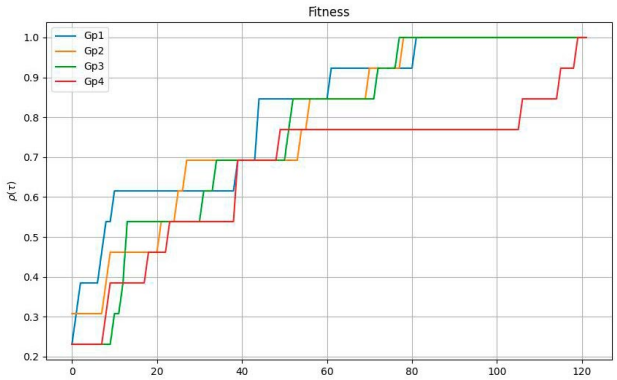

FIGURE 3. Numerical performance profile for fitness metric $t^d_{p,s}$ described as fitness value for each of the four algorithms.

Regarding efficiency, the metric $N\hat{F}E$ is associated with the number of evaluated functions and represents the hardware computational effort that the algorithmic strategy performs. Figure 4 is the performance profile, which indicates that $Gp4$ performs remarkably well. This may be because $Gp4$ employs the fewest mutations in the Genetic Algorithm and the highest number of cores in its algorithmic strategy. Similarly, $Gp1$ also demonstrates remarkable performance. It is worth noting that the previously described fitness results are consistent with the performance of $Gp1$ with its algorithmic strategy by using the fewest number of processing cores and the lowest level of mutation.

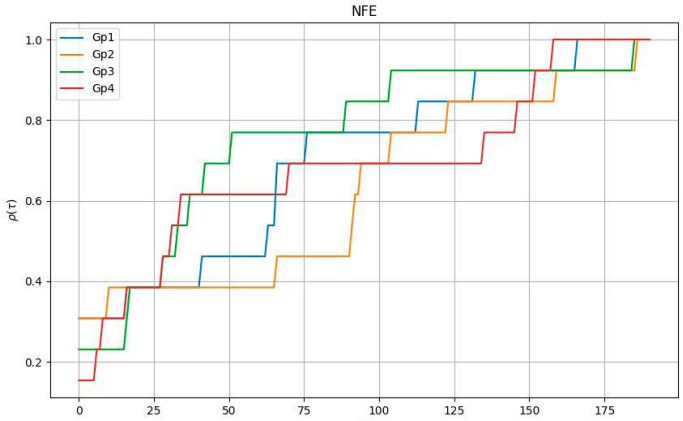

FIGURE 4. **Numerical performance profile for the metric $N\hat{F}E$.**

Figure 5 indicates that the $Gp4$ optimization method outperforms the others in terms of efficiency, as measured by the time spent on optimization $\hat{T}$. This may be because $Gp4$ utilizes greater parallel processing cores and fewer mutation processing than the other methods.

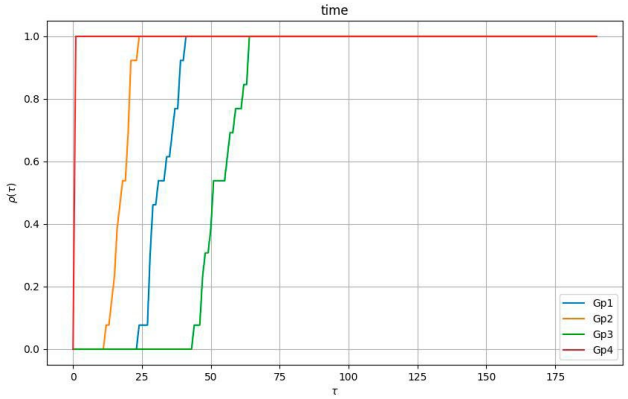

FIGURE 4. **Numerical performance profile for the metric $\hat{T}$.**

However, the Gp4 algorithm did not produce remarkable fitness or $N\hat{F}E$ values results. This shows that ensuring an efficient algorithmic and mutation process is more important than incorporating hardware resources.

CONCLUSIONS

This paper compares and discusses four strategies of the Genetic Algorithm method using the numerical performance profile model [47] in a set of genetic sequences. Based on the findings, the Gp1 approach is the most robust compared to other methods studied in this research. However, the efficiency, measured by the number of func-

tions evaluated and convergence time, varies between the alignment methods. *Gp*4 is the method with the fewest evaluated functions and the highest processing speed. Thus, it can be concluded that the number of cores influences the alignment method's efficiency in parallel processing, and the robustness of the alignment method is related to the number of mutations. However, the *Gp*2 algorithm, with the highest core number and mutation level, did not perform remarkably well compared to the other strategies that varied the mutation level and number of cores.

Increasing the sequences in the data set is recommended to obtain more accurate results. By testing with the data set used in this research, these findings can be compared to other studies on genetic algorithms. Finally, it is essential to note that variations in mutation levels or other parameters can affect the performance of the different strategies.

On the other hand, conventional computer equipment can produce promising results when processing genetic information if the algorithms are specifically designed to make the most of hardware resources. Additionally, the computational effort of the hardware used impacts the number of evaluated functions. The quality of the solution obtained in the case of multiple sequence alignment relies on specific parameters of the genetic algorithm, such as the size of the population, the mutation level, and the crossing method. Moreover, the comparison method based on the determination of the performance profile is recommended as a valuable strategy for contrasting results in different metrics of interest.

ACKNOWLEDGEMENTS

The authors would like to thank the Faculty of Systems Saltillo Unit of the Autonomous University of Coahuila for the facilities in this study.

COMPETING INTERESTS

The authors have no competing interests to declare.

AUTHOR CONTRIBUTION

E.R.W. conceptualization, methodology, software, writing original draft. M. M. D. S. formal analysis, investigation, supervision, validation, writing review and editing. J. V. S. validation.

REFERENCES

[1] N. Mićić and B. Mićić, "God is Dead, Long Live DNA," Agro-knowledge J., vol. 18, no. 2, pp. 143-146, Dec. 2017, doi: https://doi.org/10.7251/AGREN1702143M

[2] D. Ghoshdastidar and M. Bansal, "Dynamics of physiologically relevant noncanonical DNA structures: an overview from experimental and theoretical studies," Brief Funct. Genomics, vol. 18, no. 3, pp. 192-204, Jun. 2018, doi: https://doi.org/10.1093/bfgp/ely026

[3] E. Arunan, "One Hundred Years After the Latimer and Rodebush Paper, Hydrogen Bonding Remains an Elephant!," J. Indian Inst. Sci., vol. 100, no. 1, pp. 249-255, Jan. 2020, doi: https://doi.org/10.1007/s41745-019-00154-4

[4] A. M. Fleming and C. J. Burrows, "Formation and processing of DNA damage substrates for the hNEIL enzymes," Free Radic. Biol. Med., vol. 107, pp. 35-52, Jun. 2017, doi: https://doi.org/10.1016/j.freeradbiomed.2016.11.030

[5] É. Leroux, C. Brosseau, B. Angers, A. Angers, and S. Breton, "Méthylation de l'ADN mitochondrial," Med. Sci., vol. 37, no. 3, pp. 258-264, Mar. 2021, doi: https://doi.org/10.1051/medsci/2021011

[6] E. W. Sayers, E. E. Bolton, J. R. Brister, K. Canese, et al., 'Database resources of the national center for biotechnology information', Nucleic Acids Res., vol. 50, no. D1, pp. D20-D26, Jan. 2022, doi: https://doi.org/10.1093/nar/gkab1112

[7] GenBank and WGS Statistics, GenBank, 2024. [Online]. Available: https://www.ncbi.nlm.nih.gov/genbank/statistics/

[8] M. Abdel-Basset, L. Abdel-Fatah, and A. K. Sangaiah, "Metaheuristic Algorithms: A Comprehensive Review," in Computational Intelligence for Multimedia Big Data on the Cloud with Engineering Applications, A. K. Sangaiah, M. Sheng, Z. Zhang, Eds., Catalunya, Spain: Academic Press, 2018, pp. 185-231. doi: https://doi.org/10.1016/B978-0-12-813314-9.00010-4

[9] S. M. Almufti, A. Ahmad Shaban, Z. Arif Ali, R. Ismael Ali, and J. A. Dela Fuente, "Overview of Metaheuristic Algorithms," PGSRT, vol. 2, no. 2, pp. 10-32, Apr. 2023, doi: https://doi.org/10.58429/pgjsrt.v2n2a144

[10] D. Rodriguez, D. Gomez, D. Alvarez, and S. Rivera, "A Review of Parallel Heterogeneous Computing Algorithms in Power Systems," Algorithms, vol. 14, no. 10, art. no.. 275, Sep. 2021, doi: https://doi.org/10.3390/a14100275

[11] X. Wang and J. Liu, "Multiscale Parallel Algorithm for Early Detection of Tomato Gray Mold in a Complex Natural Environment," Front. Plant. Sci., vol. 12, art. no. 620273, May 2021, doi: https://doi.org/10.3389/fpls.2021.620273

[12] X. Dong, Y. Wu, Z. Wang, L. Dhulipala, Y. Gu, and Y. Sun, "High-Performance and Flexible Parallel Algorithms for Semisort and Related Problems," in Proceedings of the 35th ACM Symposium on Parallelism in Algorithms and Architectures, New York, NY, USA, 2023, pp. 341-353. doi: https://doi.org/10.1145/3558481.3591071

[13] M. Kimiaei, A. Hassan Ibrahim, and S. Ghaderi, "A subspace inertial method for derivative-free nonlinear monotone equations," Optimization, pp. 1-28, Sep. 2023, doi: https://doi.org/10.1080/02331934.2023.2252849

[14] A. H. Kamali, E. Giannoulatou, T. Y. Chen, M. A. Charleston, A. L. McEwan, and J. W. K. Ho, "How to test bioinformatics software?," Biophys Rev., vol. 7, no. 3, pp. 343-352, Sep. 2015, doi: https://doi.org/10.1007/s12551-015-0177-3

[15] Y. Zhang, Q. Zhang, J. Zhou, and Q. Zou, "A survey on the algorithm and development of multiple sequence alignment," Brief Bioinform., vol. 23, no. 3, art. no. bbac069, May 2022, doi: https://doi.org/10.1093/bib/bbac069

[16] M. Maiolo, X. Zhang, M. Gil, and M. Anisimova, "Progressive multiple sequence alignment with indel evolution," BMC Bioinformatics, vol. 19, no. 1, art. no. 331, Dec. 2018, doi: https://doi.org/10.1186/s12859-018-2357-1

[17] B. Schmidt and A. Hildebrandt, "Dedicated Bioinformatics Analysis Hardware," in Encyclopedia of Bioinformatics and Computational Biology, Sydney, Australia: Elsevier, 2019, pp. 1142-1150. doi: https://doi.org/10.1016/B978-0-12-809633-8.20186-6

[18] R. Guo, Y. Zhao, Q. Zou, X. Fang, and S. Peng, "Bioinformatics applications on Apache Spark," Gigascience, vol. 7, no. 8, art. no. giy098, Aug. 2018, doi: https://doi.org/10.1093/gigascience/giy098

[19] N. A. Stover and A. R. O. Cavalcanti, "Using NCBI BLAST," Curr. Protoc. Essent. Lab. Tech, vol. 14, no. 1, May 2017, doi: https://doi.org/10.1002/cpet.8

[20] S. Iantorno, K. Gori, N. Goldman, M. Gil, and C. Dessimoz, "Who Watches the Watchmen? An Appraisal of Benchmarks for Multiple Sequence Alignment," Methods Mol. Biol., vol. 1079, 2014, pp. 59-73. doi: https://doi.org/10.1007/978-1-62703-646-7_4

[21] A. Löytynoja, "Alignment Methods: Strategies, Challenges, Benchmarking, and Comparative Overview," Methods Mol. Biol., vol. 855, 2012, pp. 203-235. doi: https://doi.org/10.1007/978-1-61779-582-4_7

[22] Y. He, "Research on global double sequence alignment optimization algorithm based on dynamic programming," in Third International Conference on Computer Science and Communication Technology (ICCSCT 2022), Beijing, China, 2022, art. no. 125060L, doi: https://doi.org/10.1117/12.2662630

[23] J.-H. Hung and Z. Weng, "Sequence Alignment and Homology Search with BLAST and ClustalW," Cold Spring Harb. Protoc., vol. 2016, no. 11, art. no. pdb. prot093088, Nov. 2016, doi: https://doi.org/10.1101/pdb.prot093088

[24] Q. Zou, X. Shan, and Y. Jiang, "A Novel Center Star Multiple Sequence Alignment Algorithm Based on Affine Gap Penalty and K-Band," Phys. Procedia, vol. 33, pp. 322-327, 2012, doi: https://doi.org/10.1016/j.phpro.2012.05.069

[25] F. Tang, J. Chao, Y. Wei, F. Yang, Y. Zhai, L. Xu, Q. Zou, "HAlign 3: Fast Multiple Alignment of Ultra-Large Numbers of Similar DNA/RNA Sequences," Mol. Biol. Evol., vol. 39, no. 8, Aug. 2022, doi: https://doi.org/10.1093/molbev/msac166

[26] B. Reddy and R. Fields, "Multiple Sequence Alignment Algorithms in Bioinformatics," in Smart Trends in Computing and Communications. Lecture Notes in Networks and Systems, 2022, pp. 89-98. doi: https://doi.org/10.1007/978-981-16-4016-2_9

[27] D. Pacheco Bautista, M. González Pérez, and I. Algredo Badillo, "From sequencing to hardware acceleration of DNA alignment software: A integral review," Rev. Mex. Ing. Biomed., vol. 36, no. 3, pp. 257-275, Sep. 2015, doi: https://doi.org/10.17488/RMIB.36.3.6

[28] D. Pacheco-Bautista, "ABPSE: Alineador de ADN Basado en Paralelismo a Nivel de Bit y la Estrategia Siembra y Extiende," Rev. Mex. Ing. Biomed., vol. 40, no. 1, pp. 1-13, 2019. doi: https://doi.org/10.17488/RMIB.40.1.4

[29] A. S. M. Aljohani, F. A. Alhumaydhi, A. Rauf, E. M. Hamad, and U. Rashid, "In Vivo Anti-Inflammatory, Analgesic, Sedative, Muscle Relaxant Activities and Molecular Docking Analysis of Phytochemicals from Euphorbia pulcherrima," Evid. Based Complement. Alternat. Med., vol. 2022, art. no. 7495867, Apr. 2022, doi: https://doi.org/10.1155/2022/7495867

[30] J. H. Holland, Adaptation in Natural and Artificial Systems. The MIT Press, 1992, doi: https://doi.org/10.7551/mitpress/1090.001.0001

[31] K. Hao, J. Zhao, K. Yu, C. Li, and C. Wang, "Path Planning of Mobile Robots Based on a Multi-Population Migration Genetic Algorithm," Sensors, vol. 20, no. 20, art. no. 5873, Oct. 2020, doi: https://doi.org/10.3390/s20205873

[32] H. Ahrabian, M. Ganjtabesh, A. N. Dalini, and Z. R. M. Kashani, "Genetic algorithm solution for partial digest problem," Int. J. Bioinform. Res. Appl., vol. 9, no. 6, pp. 584-594, 2013, doi: https://doi.org/10.1504/ijbra.2013.056622

[33] M. Fernandez, J. Caballero, L. Fernandez, and A. Sarai, "Genetic algorithm optimization in drug design QSAR: Bayesian-regularized genetic neural networks (BRGNN) and genetic algorithm-optimized support vectors machines (GA-SVM)," Mol. Divers, vol. 15, no. 1, pp. 269-289, Feb. 2011, doi: https://doi.org/10.1007/s11030-010-9234-9

[34] M. Sale and E. A. Sherer, "A genetic algorithm based global search strategy for population pharmacokinetic/pharmacodynamic model selection," Br. J. Clin. Pharmacol., vol. 79, no. 1, pp. 28-39, Jan. 2015, doi: https://doi.org/10.1111/bcp.12179

[35] S. H. Almanza-Ruiz, A. Chavoya, and H. A. Duran-Limon, "Parallel protein multiple sequence alignment approaches: a systematic literature review," J. Supercomput., vol. 79, no. 2, pp. 1201-1234, Feb. 2023, doi: https://doi.org/10.1007/s11227-022-04697-9

[36] D. Song, J. Chen, G. Chen, N. Li, et al., "Parameterized BLOSUM Matrices for Protein Alignment," IEEE/ACM Trans. Comput. Biol. Bioinform., vol. 12, no. 3, pp. 686-694, May 2015, doi: https://doi.org/10.1109/tcbb.2014.2366126

[37] J. S. Piña, S. Orozco-Arias, N. Tobón-Orozco, L. Camargo-Forero, R. Tabares-Soto, and R. Guyot, "G-SAIP: Graphical Sequence Alignment Through Parallel Programming in the Post-Genomic Era," Evol. Bioinform. Online, vol. 19, art. no. 117693432211505, Jan. 2023, doi: https://doi.org/10.1177/11769343221150585

[38] I. R. and A. Chavoya, "PaMSA: A Parallel Algorithm for the Global Alignment of Multiple Protein Sequences," IJACSA, vol. 8, no. 4, pp. 513-522, 2017, doi: https://dx.doi.org/10.14569/IJACSA.2017.080468

[39] T. Harada and E. Alba, "Parallel Genetic Algorithms," ACM Comput. Surv., vol. 53, no. 4, pp. 1-39, Jul. 2021, doi: https://doi.org/10.1145/3400031

[40] J. Zhu, G. Wang, Y. Li, Z. Duo, and C. Sun, "Optimization of hydrogen liquefaction process based on parallel genetic algorithm," Int. J. Hydrogen Energy, vol. 47, no. 63, pp. 27038-27048, Jul. 2022, doi: https://doi.org/10.1016/j.ijhydene.2022.06.062

[41] J. Xu, L. Pei, and R. Zhu, "Application of a Genetic Algorithm with Random Crossover and Dynamic Mutation on the Travelling Salesman Problem," Procedia Comput. Sci., vol. 131, pp. 937-945, 2018, doi: https://doi.org/10.1016/j.procs.2018.04.230

[42] K. R. Anil Kumar and E. R. Dhas, "Opposition based genetic optimization algorithm with Cauchy mutation for job shop scheduling problem," Mater. Today Proc., vol. 72, pp. 3006-3011, 2023, doi: https://doi.org/10.1016/j.matpr.2022.08.263

[43] T. D. Pham and W.-K. Hong, "Genetic algorithm using probabilistic-based natural selections and dynamic mutation ranges in optimizing precast beams," Comput. Struct., vol. 258, art. no. 106681, Jan. 2022, doi: https://doi.org/10.1016/j.compstruc.2021.106681

[44] Galaxy Community, "The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2022 update," Nucleic Acids Res., vol. 50, no. W1, pp. W345-W351, Jul. 2022, doi: https://doi.org/10.1093/nar/gkac247

[45] A. Bonilla-Petriciolet, J. C. Tapia-Picazo, C. Soto-Becerra, and J. G. Zapiain-Salinas, "Perfiles de comportamiento numérico de los métodos estocásticos simulated annealing y very fast simulated annealing en cálculos termodinámicos," Ing. Inv. Tecnolog., vol. 12, no. 1, pp. 51-62, Jan. 2011, doi: https://doi.org/10.22201/fi.25940732e.2011.12n1.006

[46] R.-W. Ernesto, L.-G. Ernesto, B. Rafael, and G.-G. Yolanda, "Perfiles de comportamiento numérico de los métodos de búsqueda immune network algorithm y bacterial foraging optimization algorithm en funciones benchmark," Ing. Inv. Tecnolog., vol. 17, no. 4, pp. 479-490, Oct. 2016, doi: https://doi.org/10.1016/j.riit.2016.11.007

[47] E. D. Dolan and J. J. Moré, "Benchmarking optimization software with performance profiles," Math. Program., vol. 91, no. 2, pp. 201-213, Jan. 2002, doi: https://doi.org/10.1007/s101070100263