Optimizing Multiple Sequence Alignments using Traveling Salesman Problem and Order-based Evolutionary Algorithms

July Diana Banda Tapia[(1)], Yván Jesús Túpac Valdivia[(1)], Juan Herbert Chuctaya Humari [(2)]

(1) School of Computer Science, San Pablo Catholic University
Arequipa, Peru, {july.banda,ytupac}@ucsp.pe

(2) Cátedra Concytec en TICs, San Agustín National University
Arequipa, Peru, juanherbert@gmail.com

Keywords: Multiple Alignment Sequences; Traveling Salesman Problem; Order-based Evolutionary Algorithms.

Abstract. This paper proposes and implements a method for solving the multiple sequence alignment problem treating it as a traveling salesman problem, where each sequence is viewed as a city and the best order of visits is used as the order of alignment; this best order is found using an order-based evolutionary algorithm. Results were obtained from experiments and compared with *star alignment heuristic* outcomes.

1 Introduction

The Multiple Sequence Alignment (MSA) is an important tool for molecular sequence analysis, provide key information to determine the meaning and function of genetic sequences and proteins, discovering evolutionary relationships and predicting secondary and tertiary structures of new protein sequences.

Due to computational complexity, MSA is identified as one of the challenging tasks in bioinformatics. This problem is proved to be a NP-complete [1] with a complexity $O(n,k) = (n+1)^k(2^k-1)k^2$ when uses dynamic programming to optimize the sum of pairs of k sequences with maximum length n. Even today, MSA is solved through heuristics such as progressive alignment or clustering methods, which offer a partial solution and operate in reasonable time for a large search space [2, 3].
The aim of MSA problem is finding the order of alignment, such that the sum of pairs is optimal. In TSP the objective is similar, since it seeks the cities order visitation such the total distance traveled is minimized. In this sense, the MSA problem can be treated like a TSP problem, where a sequence s_i is viewed as the i^{th} city and the distance $d(s_i, s_j)$ is related to the SP score of the s_i and s_j sequences pair.
Nonetheless, the TSP problem is a NP-hard problem whose complexity, for an exhaustive search is $O(k!)$ for k cities. One of the first applications of dynamic programming is the HeldKarp algorithm that solves the TSP problem in $O(k^2 2^k)$ [4]. Because it, many approaches using heuristics have been developed to solve the TSP problem. So, in this paper the use of order-based evolutionary algorithms is proposed to solve the TSP problem and consequently the MSA problem.

This work is organized as follow: section 2 describes the MSA problem, section 3 describes the MSA as a TSP Problem, section 4 describes the order -based Evolutionary algorithms and the Evolutionary approaching; section 5 shows the performed experiments and obtained results and finally section 6 we draw some conlcusions and future works.

2 The MSA problem

Definition 1 (Multiple Sequences Alignment) *Given a set of k sequences $\mathcal{S} = \{s_1, \ldots, s_k\}$, where $s_i(j)$ denotes the j^{th} element of s_i, and $s_i(j) \in \Sigma$, where Σ is a finite alphabet (nucleotids or aminoacids); a multiple sequence alignment $\mathcal{A} = \{a_1, \ldots, a_k\}$ consists in inserting gaps "-" into the original strings of $\mathcal{S}$ to make them all the same length.*

Definition 2 (score of a pair) *Let a_i and a_j be a pair of sequences from an MSA $\mathcal{A}$. The score of this pair denoted as score$(a_i[m], a_j[m])$ is defined for each column m as:*

$$score(a_j[m], a_i[m]) = \begin{cases} c_{match} & a_j[m] = a_i[m] \\ c_{mismatch} & a_j[m] \neq a_i[m] \\ c_{gapped} & or\ a_j[m] = \text{"-"}\ or\ a_i[m] = \text{"-"} \\ c_{null} & a_j[m] = a_i[m] = \text{"-"} \end{cases} \quad (1)$$

where c_x usually takes the following values: $c_{match} = 1, c_{mismatch} = -1, c_{gapped} = -2, c_{null} = 0$. and the total score of the pair score(a_i, a_j) is the cumulative score for all columns:

$$score(a_j, a_i) = \sum_{l=1}^{k} score(a_j[l], a_i[l]) \quad (2)$$

Definition 3 (Sum of Pairs SP) *The sum of pairs SP$(\mathcal{A})$ is a commom scoring funcion applied in a MSA $\mathcal{A} = \{a_1, \ldots, a_k\}$ and defined by:*

$$SP(\mathcal{A}) = \sum_{i=1}^{k} \sum_{j>i} \sum_{m=1}^{n} score(a_j[m], a_i[m]) \quad (3)$$

Figure 1 shows some examples of alignments:

```
ACG-CGC      ---ACGCGC
ACGACGC      ACGACGC--
```

Figure 1: Global and local alignments

Actually, there have been developed several approaches to solve this problem such as iterative and progressive alignment algorithms, clustering methods, tabu search, genetic algorithms and many others [5]. In a progressive alignment, the algorithm starts aligning a pairwise of sequences and adding each sequence once a time to the previous alingment; Feng and Doolittle proposed [6] this kind of algorithm. Likewise, in a iterative alignment, first are build several alignments and then the sequences are incorporate to those alignments, an example of this method is the tree alignment.

3 The Travelling Salesman Problem

The TSP problem is one of the most intensely studied problems in computational mathematics. TSP is an NP-hard problem whose objective is to find the shortest way of visit all cities, given a collection of cities and the cost of travel between each pair of them. This problem has the following restrictions [7]:

- The salesman must visit each city one time.
- The salesman must return to the starting city.

Literature shows several approachs to this problem, such as: nearest neighbor, local search, tab search, cutting plane, evolutionary algorithms and others [3, 8, 9, 10, 11]; However, until today, there is no general solution to this problem [7].

3.1 MSA as a TSP problem

To model the MSA problem such as a TSP problem, a pair of sequences (s_i, s_j) is seen as two cities (c_i, c_j) and the distance between each pair of cities is associated to the score of the aligned pair (a_i, a_j) defined in (2). Scores for all aligned pairs form a triangular matrix $S = \{s_{ij}\}$ and using S, a distance matrix $D = \{d_{ij}\}$, with $d_{ij} > 0$, $\forall i, j$ is computed by:

$$d_{ij} = s_{\max} - s_{ij} + 1 \tag{4}$$

where $s_{\max}$ is the maximum score in matrix S.

Thus, like a TSP problem, the objective in this approach is finding the order of visit with minimal percorred distance, i.e., the minimal distance or maximum matching between all alignments.

4 Order-based Evolutionary Algorithms

Order-based evolutionary algorithms are a type of evolutionary algorithms whose individuals are permutations of an alphabet. An order-based evolutionary algorithm is defined as:

- Let Φ_t be a population of size m defined as a set $\Phi_t = \{\phi_i\}_{i=1}^{m}$ for the generation t.
- An individual $\phi_i \in \Phi_t$ is a set of elements $\phi_i = \{\phi_i(1), \ldots, \phi_i(n)\}$, with values from an alphabet $N = \{1, 2, \ldots, n\} \subset \mathbb{N}$.
- Let $\phi_i(p)$ denotes the value of the element in a p-th position of ϕ_i. For ϕ_i, the condition $\phi_i(p) \neq \phi_i(q) \Leftrightarrow p \neq q$ must be satisfied.
- It is clear that, for two individuals ϕ_i and ϕ_j, where $i \neq j$, the condition $\phi_i(p) = \phi_j(p)$ is not necessarily true.

An order-based evolutionary algorithm maintains the essential structure of an evolutionary algorithms, since it contains a population Φ, a selection strategy (proportional, tournament), the evaluation funcion $f(\phi_i)$ the fitness funcion $a(f(\phi_i), \Phi)$ and the reproduction process. The main difference is how the solutions are encoded into permutations and the genetic operators must always produce valid solutions. The algorithm 1 illustrates an order-based evolutionary algorithm [12].

Algorithm 1 Order-based Evolutionary Algorithm

```
1: t ← 0
2: Generate a initial random population Φ_0 of m permutations
3: while stopCriterial = false do
4:    assign a fitness f(φ) value to each element φ_i ∈ Φ_t
5:    for j = 1 → m do
6:       select two parent chromosome
7:       crossover the parent
8:       mutate new offspring
9:       place new offspring in a new population Φ_{t+1}
10:   end for
11:   t ← t + 1
12:   assign a fitness to each element of new population
13: end while
```

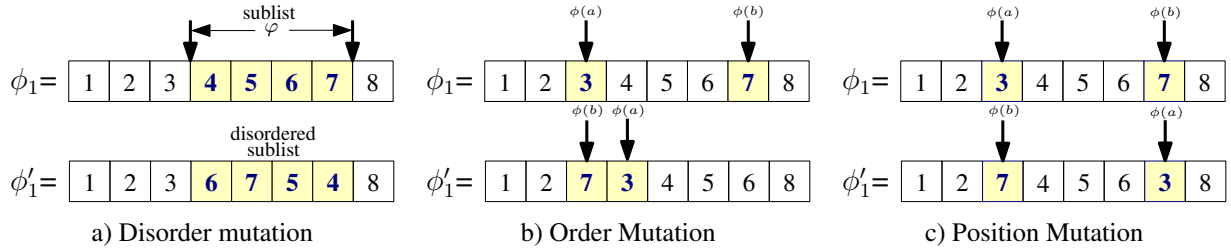
4.1 Order-based Genetic Operators

The most known operators for an order-based evolutionary algorithm are the following:

a) *Disorder Mutation*: Let $\phi_1 = \{\phi_1(1), \ldots, \phi_1(n)\}$ be a selected individual, and $\varphi \subseteq \phi_1$ a sublist, then the elements of φ are disordered. Figure 2.a illustrates this operator.

b) *Order-based Mutation (Insert Mutation)*: Let $\phi_1 = \{\phi_1(1), \ldots, \phi_1(n)\}$ be a selected individual, and $\phi(a), \phi(b), a < b$ two selected elements of ϕ_1. Then, the offspring ϕ'_1 is formed inserting $\phi_1(b)$ before $\phi_1(a)$. Figure 2.b illustrates this operator.

c) *Position-based Mutation (Swap Mutation)*: Let $\phi_1 = \{\phi_1(1), \ldots, \phi_1(n)\}$ be a selected individual, and $\phi(a), \phi(b), a < b$ two selected elements of ϕ_1. The offspring ϕ'_1 is formed swapping the positions of $\phi_1(a)$ and $\phi_1(b)$. Figure 2.c illustrates this operator.



a) Disorder mutation b) Order Mutation c) Position Mutation

Figure 2: Order-Based Mutation Operators

d) *Order-based Uniform Crossover*: Let ϕ_1,ϕ_2 be two selected individual to operate. The offspring ϕ'_1 is obtained of the following form:

- Generate a mask $\mathbf{m}_k$ of n binary bits.
- Fill ϕ'_1 copying only the genes $\phi_1(i), i = 1, \ldots, n$ where $\mathbf{m_k}(i) = 1$.
- Creates a sublist φ using the remaining elements from ϕ_1.
- Sort the sublist φ using the order of the same elements found into ϕ_2.
- Fill the remaining spaces into ϕ_1 using the reordered sublist φ.

The second offpring ϕ'_2 will be composed applying the procedure above using the negated mask $\overline{\mathbf{m}}_k$ and filling the elements of ϕ_2. Figure 3 illustrates this crossover operator.

ϕ_1=	1	2	3	4	5	6	7	8
ϕ_2=	**8**	**6**	**4**	**2**	**7**	**5**	**3**	**1**
$\mathbf{m}_k$=	0	1	1	0	1	1	0	0

ϕ'_1=	–	2	3	–	5	6	–	–
ϕ'_2=	**8**	–	–	**2**	–	–	**3**	**1**

genes ϕ_1 for $\mathbf{m}_k(i) = 0 : \{1, 4, 7, 8\}$, sorted by $\phi_2 = \{8, 4, 7, 1\}$

genes ϕ_2 for $\mathbf{m}_k(i) = 1 : \{6, 4, 7, 5\}$, sorted by $\phi_1 = \{4, 5, 6, 7\}$

ϕ'_1=	**8**	2	3	**4**	5	6	**7**	**1**
ϕ'_2=	**8**	4	5	**2**	6	7	**3**	**1**

Figure 3: Order-based Uniform Crossover

4.2 Solving the TSP problem by an Order-based Evolutionary algorithm

This work models the MSA problem such a TSP problem described as follows:

1. Firts, all pairs (s_i, s_j) are aligned using the Needleman-Wunsch algorithm [13], and build the matrix S using the scores.

2. Build the distance matrix D using Eq. (4) and the scores of S.

3. Launch the order-based evolutionary algorithm to solve the TSP problem finding the tour that offers the minimal total distance.

4. The best found tour is converted into MSA applying the consistency principle: *once a gap, always a gap*, then, the sum-of-pairs is computed.

Figure 4 illustrates this process.

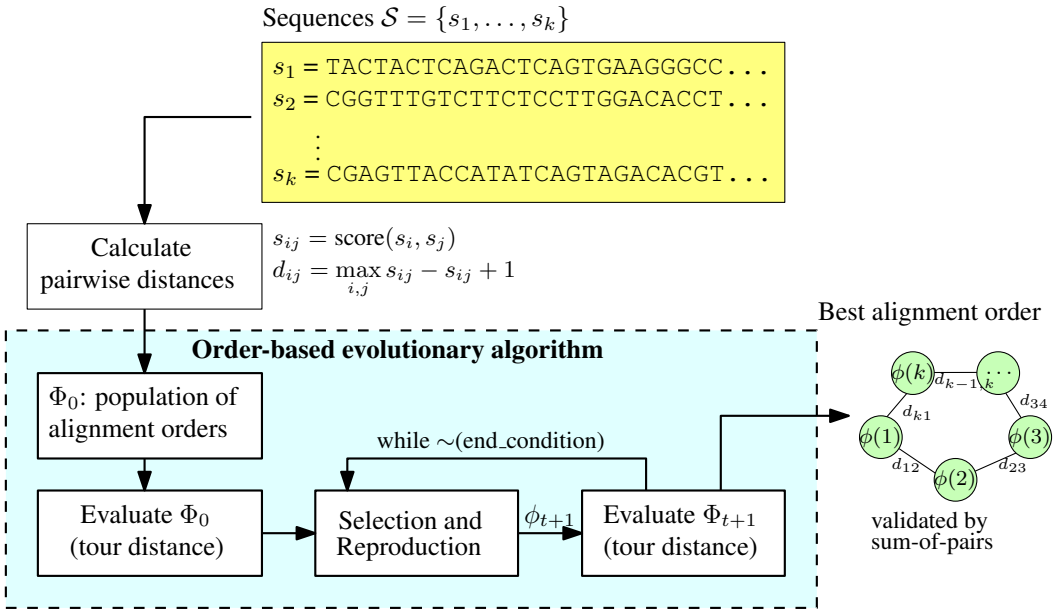


Figure 4: Framework of the MSA as TSP using order-based EA

1. *The representation* A TSP problem is straightforwardly represented by an order-based evolutionary algorithm where the individual $\phi_i = \{\phi_i(1), \ldots, \phi_i(k)\} \in \Phi$ is a random tour.

2. *The evaluation* The evaluation $f(\phi)$ calculates the total distance of the tour defined by individual $\phi = \{\phi(1), \phi(2), \ldots, \phi(n)\}$ using the matrix D as Eq. (5).

$$f(\phi_i) = \sum_{j=2}^{n} d_{\phi_i(j-1)\phi_i(j)} \tag{5}$$

   where it must be noted the absence of distance between the final and departure city $d_{\phi_i(n)\phi_i(1)}$, since the alignment of first and last sequences is not required.

3. *Selection and evolutionary operators* This work uses elitism to retain the best tour found, and the classic order operators described in section 4.1, and two special operators from [14] described as follow:

- *Inverse mutation*: a particular case of disorder mutation where a sublist φ is randomly selected and the elements in φ are not shuffled but inverted.
- *Boundary mutation*: a city is chosen and replaced randomly with either the upper or lower bound for that city. The chromosome is then searched for the upper or lower bound and that city is replaced with the bound.

5 Experiments and Results

The experiments were performed over a local database extracted from the National Center for Biotechnology Information (NCBI)[1]. 500 sequences were selected for analysis of this database, where each sequence contains 80-120 nucleotides. To perform the experiments, the sequences were grouped in groups of 10 sequences each totalizing 50 groups.

The algorithms (pairs alignment, TSP model and order-based evolutionary model) were implemented using C++ language. Experiments were done over a genetic algorithm using the parameters of Table 1.

Table 1: Order-based evolutionary model parameters

Parameter	Value
Generations	100
Population	100
Crossover rate	70%
Mutations rate	30%

The first experiment was performed over the 500 sequences, inproving in 16% the quality of the alignments when compared to the traditional alignment. The table 2, shows the maximum, minimum and averages values of the obtained alignment by star alignment and the proposed TSP-OEA alignment.

Table 2: Star and TSP alignments comparison

Value	Star	TSP-EA
Maximum	-566	-1816
Minimum	-6410	-7505
Average	-3319	-4690

The figures 5 and 6, shows the iteration where was found the best fitness value and the best fitness value for each alignment.

[1]National Center for Biotechnology Information, available in `http://www.ncbi.nlm.nih.gov/`

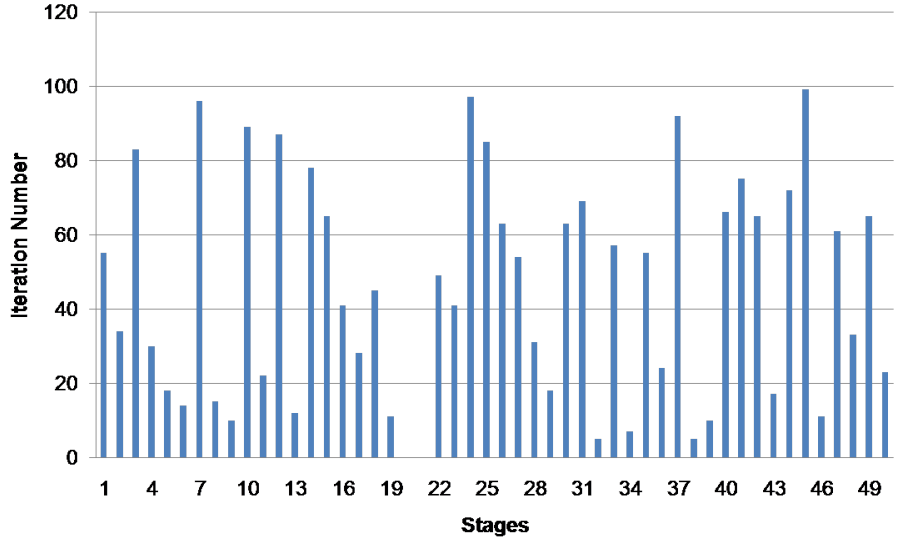


Figure 5: Iteration where was found the best fitness value, per alignment

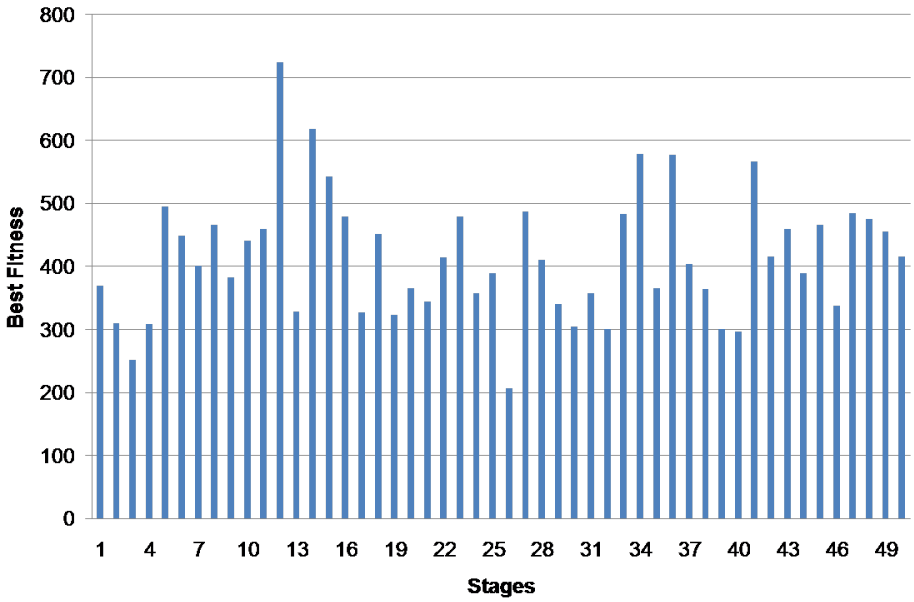


Figure 6: Best fitness value found in each alignment

6 Conclusions and Future Works

The present work have shown an approach that applies the TSP with an order-based evolutionary algorithms to solve the multiple alignment sequences problem. This approach applies a distance metric obtained from the scores of pairwise alignments previously calculated. The outcome results, associated to the best TSP tour, shows that this approach can reach an important improvement respect to the classical star alignment heuristic. Some futures works consist in take in consideration the idea of hierarquical order given by the progressive MSA models (based in a clustering algorithm to construct a tree and use this tree as order of sequences alignment), in this case we pretend to apply Genetic Programming [15] to find the best tree and best order to k sequences alignment.

References

[1] L. Wang and T. Jiang, "On the complexity of multiple sequence alignment," *Journal of Computational Biology*, vol. 1, pp. 337–348, 1994.

[2] C. Korostensky and G. Gonnet, "Near optimal multiple sequence alignments using a traveling salesman problem approach," in *String Processing and Information Retrieval Symposium, 1999 and International Workshop on Groupware*, 1999, pp. 105–114.

[3] O. M. Sallabi and Y. El-Haddad, "An improved genetic algorithm to solve the traveling salesman problem," *World Academy of Science, Engineering and Technology*, vol. 52, pp. 471–474, 2009.

[4] M. Held and R. M. Karp, "A dynamic programming approach to sequencing problems," *Journal of the Society for Industrial and Applied Mathematics*, vol. 10, pp. 196–210, 1962.

[5] N. C., "Recent progress in multiple sequence alignment: A survey," *Pharmacogenomics*, no. 1, 2002.

[6] D. Feng and R. Doolittle, "Progressive sequence alignment as a prerequisite to correct phylogenetic trees," *Journal of Molecular Biology*, pp. 351–360, 1987.

[7] W. Cook. (2007) TSP traveling salesman problem. [Online]. Available: http://www.tsp.gatech.edu//index.html

[8] K. Mehdizadeh, M.A.Nekooui, K. Sabahi, and A. Akbarimajd, "A modified dna-computing algorithm to solve tsp," *ICM 2006-IEEE 3rd International Conference on Mechatronics*, pp. 65–68, June 2006.

[9] Y. Y. Lee, S.-Y. Shin, T. H. Park, and B.-T. Zhang, "A modified dna-computing algorithm to solve tsp," *BioSystems*, pp. 39–47, June 2004.

[10] D. Chao, C. Ye, and H. Miao, "Two-level genetic algorithm for clustered traveling salesman problem with application in large-scale tsps," *Tsinghua Science and Technology*, vol. 12, no. 4, pp. 459–465, August 2007.

[11] S. Patvichaichod, "An improved genetic algorithm for the traveling salesman problem with multi-relations," *Journal of Computer Science*, pp. 70–74, 2011.

[12] B. H. Hasan and M. Saleh, "Comparative study of mutation operators on the behavior of genetic algorithms applied to non-dterministic polynomial (np) problems," *Second International Conference on Intelligent Systems, Modelling and Simulation*, pp. 7–12, 2011.

[13] S. B. Needleman and C. D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins," *Journal of Molecular Biology*, vol. 48, no. 3, pp. 443–453, 1970.

[14] B. H. Hasan and M. S. Mustafa, "Comparative study of mutation operators on the behavior of genetic algorithms applied to non-dterministic polynomial problems," *Second International Conference on Intelligent Systems Modelling and Simulation*, pp. 7–12, 2011.

[15] J. R. Koza, "Hierarchical genetic algorithms operating on populations of computer programs," in *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, N. S. Sridharan, Ed. San Mateo, California: Morgan Kaufmann, 1989, pp. 768–774.