Aligning Multiple Sequences with Genetic Algorithm

Segun A. Fatumo, Ibidapo O. Akinyemi and Ezekiel .F. Adebiyi

Abstract— **The alignment of biological sequences is a crucial tool in molecular biology and genome analysis. It helps to build a phylogenetic tree of related DNA sequences and also to predict the function and structure of unknown protein sequences by aligning with other sequences whose function and structure is already known. However, finding an optimal multiple sequence alignment takes time and space exponential with the length or number of sequences increases. Genetic Algorithms (GAs) are strategies of random searching that optimize an objective function which is a measure of alignment quality (distance) and has the ability for exploratory search through the solution space and exploitation of current results.**

In this paper, we view the multiple sequence alignment problems as an optimization problem and present a stochastic approach based on GAs for finding globally optimal multiple alignments in reasonable time, starting from completely unaligned sequences.

Index Terms— **Multiple Sequence, Optimization, Genetic Algorithm**

I. INTRODUCTION

Multiple Sequence Alignments (MSA) has been found to be an important research area of bioinformatics. Numerous results have been obtained and can be used for more complicated genetic research [1]. MSA are used to help predict the secondary or tertiary structure of new sequences; to help demonstrate homology between new sequences and existing families; to help find diagnostic patterns for families; to suggest primers for PCR and as an essential prelude to phylogenetic reconstruction.

The conventional multiple sequence alignment algorithms are classified into two categories: iterative improvement strategies [2] and simulated annealing methods [3]. But majority of the automatic multiple alignments are now carried out using the 'progressive' of [4] or variations of it [5], [6]. This approach has the great advantage of speed and simplicity combined with reasonable sensitivity as judged by the ability to align sets of sequences of known tertiary structure. The main disadvantage of this approach is the 'local minimum' problem which stems from the greedy nature of the algorithm. This means that if any mistakes are made in any intermediate alignments, these cannot be corrected later as more sequences are added to the alignment. Further, there is no objective function (a measure of overall alignment quality) which can be used to say that one alignment is preferable to another or to say that the best possible alignment, given a set of parameters, has been found.

The MSA program [7], [8] used the concept of iterative improvement strategies, and attempted to narrow down the solution space to a relatively small area where the best alignment is likely to be. It then guarantees finding the best alignment in this reduced space. Even with this reduction, it is limited to small examples of around seven or eight sequences at most. Nonetheless, it is the only method we know of that seems capable of finding the globally optimal alignment or close to it, starting with completely unaligned sequences. Stochastic optimisation methods, such as simulated annealing [9], Gibbs sampling [10] have been used on numerous occasions for multiple alignment but can be very slow and usually works well as an alignment improver in most cases i.e. when the method is given an alignment that is already close to optimal and is not trapped in a local minimum [11], [12]. Gibbs sampling has been very successfully applied to the problem of finding the best local multiple alignment block with no gaps but its application to gapped multiple alignment is not trivial.

It is worth saying that multiple sequence alignment belongs to a class of optimization problems with exponential time complexity, called combinatorial problems. In biology, the sequences can have lengths in the order of hundreds (proteins), thousands (RNA), or millions of units (DNA). In a simple GA [13], a solution of a given problem is represented as "chromosomes" which consists of bit strings of 0's and 1's. The genetic operations, such as reproduction, crossover and mutation, are applied to a population of chromosomes to

Manuscript received 28th February, 2009.

Segun. A. Fatumo is a lecturer in the department of Computer and Information Sciences, Covenant University, Ota, Nigeria where he also received his Ph.D. His current research interests are metabolic network modeling and analysis, Drug target identification and validation. He is a member of many internationally recognized professional societies. He was the founder and pioneer president of Region Student Group of International Society of Computational Biology-Student Council (ISCB-SC).

Ibidapo O. Akinyemi is a Ph.D. student in the Department of Computer and Information Sciences, Covenant University, Ota, Nigeria. He holds B. Sc. Mathematical Sciences (Computer Science option) and M. Sc. (Computer Science) . His current research interests are on Computational Intelligence and Software Engineering. He is a member of the Nigeria Computer Society and Computer Professional Registration Council of Nigeria.

Ezekiel F. Adebiyi is a Senior Lecturer and the group leader of the Bioinformatics unit of the Department of Computer and Information Sciences, Covenant University, Ota, Ogun State, Nigeria. He received his Ph.D. in Algorithm and Bioinformatics from the University of Tuebingen, Germany. He is an international scholar and well known scientist in the field of Bioinformatics in Africa.

create a new population of chromosomes. This process is repeated many times as found in [14] so that we can obtain a nearly optimal alignment.

The rest of this paper is succinctly described as follows. Section 2 gives the basic concepts of GA. Section 3 briefly discusses the MSA and section 4 concentrates on the methods and systems of MSA as well as the complexity of its algorithm. Section 5 focuses on the application of GA to the MSA while section 6 concludes the paper. .

II. BASIC CONCEPT OF GENETIC ALGORITHMS

GA is an optimization technique that was formulated during the early years of the 1970's by John Holland [15]. It is a stochastic search algorithm based on the mechanics of natural selection and population genetics. Genetic algorithms are patterned after natural genetic operators that enable biological populations to effectively and robustly adapt to their environment and to changes in their environment. GA, as stated and demonstrated by [15] is theoretically and empirically proven to provide robust search in complex spaces. The GA performs its search, balancing the need to retain population diversity 'exploration', so that potentially important information is not lost, with the need to focus on fit portions of the population 'exploitation'. Reproduction in GA theory, as in biology, is defined as the process of reproducing offspring. However, mating may occur between any two classifiers, as there is no male- female distinction.

Over the years, GA has been used to solve a wide range of search, optimization and machine learning problems. As the name indicates, genetic algorithm attempts to solve problems in a fashion similar to the way in which human genetic processes seem to operate. GAs starts with an initial population of chromosomes chosen at random. In contrast to other search techniques, GAs has no need for auxiliary information about features of search space. They only require the value of an application-dependent objective function to be associated with an individual chromosome. Each member of the initial population must be evaluated using this function. Objective function associates a numerical value (also called "fitness" value) with a chromosome, which serves as some measure of "goodness" of a chromosome, that is, a measure of how well the chromosome fits the search space or solves the problem at hand that is to be maximized. Further steps of GA are repeated iteratively until either all chromosomes have the gene-associated fitness value (convergence conditions) or the desired number of iterations is reached. Each iteration of the algorithm consists of two basic steps: **"Selection"** and **"Recombination"**.

Genetic algorithms are used in solving problems in the areas of cellular automata, fuzzy logic, image registration [16], communications network configuration, simulation modeling and optimization [17], time-tabling [18], multiobjective workforce scheduling , time constraint scheduling of limited resources, and combinatorial optimization. The most widely studied combinatorial task is traveling salesman problem [19]. Bin packing problems are also widely studied [20]. They have been utilized in playing games such as SimCity, SimEarth; in biology, chemistry and medicine; circuitry design and computer engineering; network routing for the telephone company; to detect computer viruses; for military artificial intelligence applications; military guidance and deciphering applications; art and music. GAs has been shown to be able to out-perform conventional optimisation techniques of difficult, discontinuous, multimodal, noisy functions [21].

A. Outline the Basic Genetic Algorithm

1. **[Start]** Generate random population of *n*-chromosomes (suitable for the problem)
2. **[Fitness]** Evaluate the fitness *f(x)* of each chromosome *x* in the population.
3. **[New population]** Create a new population by repeating the following steps until the new population is complete
   i. **[selection]** select two parent chromosomes from a population according to their fitness (the better the fitness, the higher the chances to be selected)
   ii. **[Crossover]** with a crossover probability, crossover the parents to form a new offspring (children). If no crossover was performed, offspring is an exact copy of parents.
   iii. **[Mutation]** with a population probability, mutate new offspring at each locus (positions in chromosomes).
   iv. **[Accepting]** place new offspring in a new population.
4. **[Replace]** Use newly generated population for a further run of the algorithm.
5. **[Test]** If the end condition is satisfied, stop, and returnthebest solution in current population.
6. **[Loop]** Go to step 2.

The main advantage of genetic algorithms over other optimization methods is that there is no need to provide a particular algorithm to solve a given problem. It only needs a fitness function to evaluate the quality of different solutions [22]. Also since it is an implicitly parallel technique, it can be implemented very effectively on powerful parallel computers to solve exceptionally demanding large-scale problems.

III. MULTIPLE SEQUENCE ALIGNMENT

Multiple sequence alignment (MSA) refers to the problem of optimally aligning three or more sequences of symbols with or without inserting gaps between the symbols, [22]. The objective is to maximize the number of matching symbols between the sequences and also use only minimum gap insertion, if gaps are permitted. This problem appears in several fields, such as molecular biology, geology, and computer science. In biology it is especially important for constructing evolutionary trees based on DNA sequences and for analyzing the protein structures to help design new proteins. Multiple sequence alignment belongs to a class of optimization problems with exponential time complexity, called combinatorial problems. It exhibits $O(L^N)$ time complexity where L is the mean length of the sequences to be aligned and N is the number of sequences [23].

Definition 1: Let $S = \{s_1, s_2, \ldots, s_n\}$ be the input sequences and assume that n is at least 2. Let Σ be the input alphabet; we assume that S does not contain the character '–', so that a dash can be used to denote a gap in the alignment. A set $S' = \{s'_1, s'_2, \ldots, s'_n\}$ of strings over the alphabet Σ $\cup\{-\}$, is called an *alignment* of *S* if the following two properties hold;

i. The strings in S¢ have the same length.

ii. Ignoring dashes, string S¢ *i* is identical with string *Si.*

An alignment can be interpreted as an array with n rows, one row for each S¢ i . Two letters of distinct strings are called aligned under S¢ if they are placed into the same column [24].

To compare different alignments, a fitness function is defined based on the number of matching symbols and the number and size of gaps. In biology, this **fitness** function is referred to as cost function and is given biological meaning by using different weights for different types of matching symbols and assigning gap costs when gaps are used [25]. The algorithms that construct multiple sequence alignment require a cost model as a criterion for constructing optimal alignment. In the simplest cost model there is a **cost function** defined by [24] as

sub: Σ' X Σ' → N.

It can be defined such that sub(a, b) is the cost of substituting a, b in the second sequence for an a in the first sequence; also, sub(-, b) is the cost for columns where the first sequence has a gap and the second has a, b; and sub(a, -) is the cost for columns where the first sequence has an a and the second has a – .

IV. METHODS AND SYSTEMS

A. The Problem

Given a finite alphabet set Σ and a family S = $\{s_1, s_2, \ldots, s_n\}$ of *n* sequences of various length i_1 *to* i_n.

$$s_i = s_{i1}, s_{i2}, .., s_{in} (1 A_i .. A_n), \quad s_{ij} \in \sum (1 A_j \quad A_i)$$

where for Deoxyribonucleic acid (DNA) sequences, Σ consists of four characters {A,C,G,T}, and for protein sequences, Σ consists of twenty characters of amino acids, an alignment of S is a matrix $A = (a_{ij})_{1} A_{i} A_{n,1} A_{j} A_{l,}$ $_{max(l1)} A_{l} A \Sigma_{l1}$ satisfying:

i. $a_{ij} \in \Sigma \cup \{-\}$, we denote "-" as gap letters;

ii each row $a_i = a_{i1}a_{i2}...a_{il}$(1 A_i A_n) of A is exactly the corresponding sequences s_i if we eliminate all gap letters;

iii A has no column which only contains gaps.

For example, if given three sequences, a better alignment could be

s_1 = A A T T C C - T - G G A T C G T C G G

s_2 = A A T - C - - T - G G - T C G T C T G

s_3 = A - T T C G A T G G - A T C - - C G G

In real life problem, sequences are normally very long in their hundreds or thousands and they are not necessarily of equal length just as it shown in the above example.

B. The Algorithm Complexity

Assuming the size of the population is N, the number of generations is G, the crossover probability is Pc, the mutation probability is Pm, Then during one cycle, the crossover computation time is N x Pc, the mutation computation time is N x Pm, the selection computation time is N, so the algorithm complexity is O (G x N). It is related to the number of generations and the size of the population.

V. APPLICATION OF GENETIC ALGORITHMS TO MULTIPLE SEQUENCE ALIGNMENT

The use of GA as described by [13] for MSA involves using a population of solution (made of alignments) which evolves by means of natural selection. According to [26] an initial generation of zero (G_0) is randomly created and the size of the population is kept constant. To go from one generation to the next, children are derived from parents that are chosen by some kind of natural selection, based on their fitness (that is, the better the parent the more children it will have). To create a child, an operator is selected that can be a crossover (mixing the contents of two parents) or a mutation (modifying a single parent). Each parent has a probability of being chosen that is dynamically optimized during the run. These steps are repeated iteratively, generation after generation, giving rise to generating new pieces of alignment because of the mutation and are combined by crossover. The selection makes sure that the good pieces survive and the dynamic setting of the operation helps the population to improve by creating the children it needs.

Following the basic GA concepts, the following pseudo code illustrates the overall operation [26];

Initialization

1. Create G_0

Evaluation

2. Evaluate the population of generation *n* (G_n)
3. If the population is stabilized, then End
4. Select the individual to replace
5. Evaluate the expected offspring

Breeding

6. Select the parent(s) from G_n
7. Select the operator
8. Generate new child
9. Keep or discard the new child in G_{n+1}
10. Goto 6 until all the children have been successfully put into G_{n+1}
11. $n = n + 1$

12. Goto Evaluation
13. End

A. CASE ONE: Multiple Sequence Alignment Without Gaps

The Alignment :

Case one example tries to align the sequences without gaps in between any sequence i. Each chromosome is composed of N genes. Each gene stores the translation of its corresponding sequence.

The fitness function can be as simple as counting the total number of matching symbols, or as complicated as considering the type of symbols aligned, their location in the sequences, their neighboring symbols, etc.

In this implementation, we use the simplest possible fitness function, which is to count the total number of matches and then assign 1 point for each match:

Fitness = (total matches) * 1.0

B. CASE TWO: Multiple Sequence Alignment with Gaps

The Alignment:

For Multiple sequence alignment with gaps, the fitness function is modified to include some penalty points for the gaps:

Fitness = (total matches) * 1.0 - (gap penalties)

VI. CONCLUSION

Multiple sequence alignment is very useful in many scientific fields, including biology. However, it belongs to the combinatorial optimization problems with exponential time complexity. GA is a fairly new optimization technique that is effective for this type of problems. In this study we described the genetic algorithms methodology to produce optimal or near-optimal solutions to the MSA problem. Two different types of alignments are considered: alignments with and without gaps. In both cases, it could be deduced that genetic algorithms produce reasonably good solutions.

REFERENCES

[1] C. Shyi-Ming , L. \ Chung-Hui, and C. Shi-Jay. International Journal of Applied Science and Engineering 2005. 3, 2: 89-100.
[2] M. P. Berger,. and P. J. Munson. Comput. Applic. Biosci., Vol. 7, pp. 479-484, 1991.
[3] J. Kim, S. Paramanik, and M.J. Chung, *Comp. Applic. Biosci*., vol.**10**, 1994 pp 419–426.
[4] D. F. Feng, and R.F. Dolittle. *J. Mol. Evol*., **25**, 1987, pp 351–360.
[5] W.R. Taylor. *J. Mol. Evol*., **28**, 1988, pp 161–169.
[6] G.J. Barton, and M.J.E. Sternberg. *J. Mol. Biol*., **198**, 1987, pp 327–337.
[7] D.J. Lipman, S.F. Altschul, and J.D. Kececioglu. *Proc. Natl. Acad. Sci. USA*, **86**, 1989, pp 4412–4415.
[8] S.K. Gupta, J Kececioglu,. and A.A. Schaffer. *J. Comput. Biol.*, **2**, 1996, pp. 459–472.
[9] E.H.L. Aart, and P.J.M. Van Laarhoven. *Simulated Annealing:a Review of Theory and Applications*, Kluwer Academic Publishers, Amsterdam, 1987**.**
[10] C.E. Lawrence, S.F. Altschul, M.S. Boguski, J.S. Liu, , A.F. Neuwald, and J.C. Wootton. *Science*, **262**, 1993, pp. 2–10.
[11] M. Ishikawa, T. Toya, and Y. Totoki. *Artificial Intelligence and Genome Workshop, 13th International Joint Conference on Artificial Intelligence*, Chambery, France, 1993.
[12] M. Hirosawa, M. Hoshida, M. Ishikawa, and T. Toya. *Comp. Applic.Biosci*., **9**, 1993, pp. 161–167.
[13] D.E. Goldberg, (ed.). *Genetic Algorithms in Search, Optimisation and Machine Learning*, Addison-Wesley, New York, 1989.
[14] I. O. Akinyemi, and S. Agholor. A Classification Technique for Credit Risk Assessment Problems using Genetic Algorithm. Journal of Computer Science and its Applications (An International Journal of Computer Society) vol. 12 N0 1, 2006, pp 37 – 46.
[15] J.H. Holland. "Adaptation in Natural and Artificial Systems", Ann Arbor:The University of Michigan Press, 1975.
[16] J. J. Grefenstette, and J. M. Fitzpatrick. *Genetic Search with Approximate Function Evaluations*, Proceedings of the First International Conference on Genetic Algorithms and Their Applications, Hillsdale, New Jersey, Lawrence Erlbaum Associates, 1985.
[17] F . Azadivar, and G. Tompkins. *Simulation optimization with qualitative variables and structural model changes: A genetic algorithm approach*, European Journal of Operational Research, 113, pp 169-182, 1999.
[18] S. J. K. Horman.. *Using Genetic Algorithms to Schedule The University of New South Wales Examination Timetable*, MSc Thesis, The University of New South Wales, 1998.
[19] D. Goldberg, R.and R. Lingle.. *Alleles, loci and the traveling salesman problem.* In J. J. Grefenstette, editor, Proceedings of International Conference on GAs. Lawrence Erlbaum, 1985.
[20] L. Davis. *Applying adaptive algorithms to epistatic domains.* In 9th International Joint Conference on AI, pp 162- 164, 1985.
[21] K. DeJong. *The Analysis and behaviour of a Class of Genetic Adaptive Systems*, PhD thesis, University of Michigan, 1975
[22] K. Kosmas, and H. K. Donald. Genetic Algorithms and the Multiple Sequence Alignment Problem in Biology. Proceedings of the Second Annual Molecular Biology and Biotechnology Conference, February, 1996, Baton Ronge, LA.
[23] H. Carrillo, and D. Lipman The multiple sequence alignment problem in biology", Siam J. Appl.Math., vol. 48, no. 5, pp. 1073-1082, October 1988.
[24] L. A. Anbarasu and V. Sundararajan and Narayanasamy Narayanasamy (2001). Parallel genetic algorithm for performance-driven sequence alignment. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*. San Francisco, California, USA, pp. 1015.
[25] S. F. J. Altschul. "Gap Costs for Multiple Seqence Alignment". . heoretical Biol., Vol 138, 1989, pp 297 – 309.
[26] N. Cedric, and G. H. Desmond. SAGA: Sequence Alignment by Genetic Algorithm. Nucleic Acids Research, Vol. 24 No 8, 1996, pp 1515 – 1524.