Evolving Consensus Sequence for Multiple Sequence Alignment with a Genetic Algorithm

Conrad Shyu and James A. Foster

Initiatives for Bioinformatics and Evolutionary Studies (IBEST)
Department of Computer Science
University of Idaho, Moscow, Idaho 83843, USA
{tsemings,foster}@cs.uidaho.edu

Abstract. In this paper we present an approach that evolves the consensus sequence [25] for multiple sequence alignment (MSA) with genetic algorithm (GA). We have developed an encoding scheme such that the number of generations needed to find the optimal solution is approximately the same regardless the number of sequences. Instead it only depends on the length of the template and similarity between sequences. The objective function gives a sum-of-pairs (SP) score as the fitness values. We conducted some preliminary studies and compared our approach with the commonly used heuristic alignment program Clustal W. Results have shown that the GA can indeed scale and perform well.

1 Introduction

Living things diverge from common ancestors through changes in deoxyribonucleic acid (DNA) and millions of years of evolution [6]. DNA indeed plays a fundamental role in the processes of life in various aspects. It contains the template for the synthesis of proteins, which are crucial molecules for life. Moreover, DNA is essential to life because it functions as a medium to transmit information from one generation to another [10]. Evidently the most important regions in DNA are generally conserved to ensure survival. Sequence alignment is commonly used to detect and quantify similarities in DNA or protein sequences. Alignments of biological sequences generated by computational algorithms are routinely used as a basis for inference about sequences whose structures or functions are not well known [7]. The most common approach is to find the best-scoring algorithm between a pair of sequences where the score records aligning similar residues and penalizes substitutions and gaps. The best-scoring alignment is commonly found by the dynamic programming (DP) algorithms, such as Smith-Waterman and Needleman-Wunsch algorithms [14, 23]. DP algorithms guarantee a mathematically optimal alignment for the given evolutionary model; however, the complexity of DP algorithms grows exponentially as the length and number of sequences increase. Specifically multiple sequence alignments (MSA) with DP have been shown to be NP-hard [19]. Several heuristic approaches, such as Clustal W [20], are frequently used to approximate the optimal alignments. In this paper, we present an approach that utilizes the guided search in GA to evolve the most probable consensus sequence [25] for MSA.

E. Cantú-Paz et al. (Eds.): GECCO 2003, LNCS 2724, pp. 2313–2324, 2003.

The design of GA is derived from the most commonly used DP algorithms for sequence alignments. In addition, we have developed an encoding scheme such that the search complexity does not depend on the number of sequences. The search complexity instead depends on the length of the consensus sequence and similarity between sequences. The scheme encodes each possible matching nucleotide at given column with binary masks. This compact representation greatly reduces the space requirement as well as the search complexity. The GA constructs the final alignment through the backtracking process, which is identical to the one found in DP algorithms [14]. The objective or evaluation function gives the sum-of-pairs (SP) scores to determine the fitness of each chromosome in the population. SP score has been widely used to detect and quantify similarities between sequences; however it does not provide any probabilistic or biological justifications [7]. To further improve the performance of GA, we have devised a sequence profiling formulation that reduces the complexity for calculating the SP scores. We have compared our approach to the most commonly used heuristic alignment program Clustal W and demonstrated that GA can indeed perform and scale well. In most cases, the GA outperformed Clustal W and produced better alignments.

2 Sequence Alignment

There are diverse motivations behind the alignment of biological sequences. Genetic sequences are inherited from common ancestors through millions of years of evolution. Therefore, it is of interest to trace evolutionary history of mutation and other evolutionary changes through sequencing [2, 6]. Alignment of biological sequences in this context is generally understood as comparisons based on the criteria of evolution. For example, the number of mutations, insertions, and deletions of residues necessary to transform one DNA sequence into another is a measure of phylogeny or evolutionary relatedness [3, 13]. On the other hand, a comparison may pinpoint regions of common origin, which may in turn coincide with regions of similar structure or function [10]. A pairwise sequence alignment is a technique of arranging two sequences, so that the residues in certain positions are deemed to have a common evolutionary origin. In other words, if the same residue occurs in both sequences at the same position then it may have been conserved during the course of evolution. On the other hand, if two resides differ then it is generally assumed that they may have derived from a common ancestor. Homologous sequences, those related by common descent, may have different lengths, which is generally explained through insertions or deletions [6, 10].

DP has been commonly used to align two sequences because it guarantees a mathematically optimal alignment. MSA, on the other hand, is simply an extension of pairwise sequence alignment. MSA is the process of aligning three or more sequences simultaneously to bring as many similar residues into register as possible. The resulting alignments are commonly interpreted into two contexts; (a) to find regions that define a conserved pattern or domain; and (b) to derive the possible phylogeny or evolutionary relationships among the sequences [13]. The presence of similar domains in several similar sequences implies a similar biochemical function or structural fold that may be used as the basis for further experimental investigation. Simul-

taneous alignment of three or more sequences with DP, however, poses a difficult algorithmic challenge.

2.1 Dynamic Programming (DP)

Dynamic programming (DP) is a commonly used method for solving sequential or multi-stage decision problems and is recursive in nature. The essence of DP is the principle of optimality [15, 17]. DP has long been used to solve varieties of discrete optimization problems such as scheduling, string-editing, packaging, and inventory management [12]. It views a problem as a set of interdependent sub-problems. DP solves sub-problems and uses the results to solve larger sub-problems. The solution to a sub-problem is expressed as a function of solutions to one or more sub-problems at the preceding levels [7]. In other words, DP expresses the problem in a recurrent formulation. To make optimal decisions for the next and all future states, DP only needs to know the current decision. This is also known as the *Markovian property*. For a process to be Markovian the future must depend only on the present state, and past should not have any effect on the future [7, 12]. The term *programming* in the name actually refers to the mathematical rules that can be easily followed to solve a problem; it has nothing to do with writing a computer program. DP is known to be an efficient programming technique for solving certain combinatorial problems. It is particular important in bioinformatics [17], as it is the basis of sequence alignments for comparing DNA and protein sequences. The following figure shows the recurrent formulation of DP for sequence alignment.

$$F(i, j) = \max \begin{cases} F(i-1, j-1) + s(x_i, y_j), \\ F(i-1, j) - d, \\ F(i, j-1) - d. \end{cases}$$

Fig. 1. This recurrent equation is applied repeatedly to fill the matrix of $F(i, j)$ values. This particular formulation gives the global alignment of two sequences. $F(i, j)$ is the maximum of three previous values, namely $F(i-1, j-1)$, $F(i-1, j)$, and $F(i, j-1)$. The value $s(x_i, y_j)$ is the score for aligning the characters x_i and y_j and d is the penalty for inserting a gap

For pairwise sequence alignments, for example, DP first begins with the construction of an alignment matrix $F(i, j)$ with the indexes (i, j) for the two sequences S_x and S_y. The matrix is initialized with $F(0, 0)=0$. The value of $F(i, j)$ is the score of the best alignment from the first character x_1 to the character x_i of sequence S_x and the first character y_1 to the character y_j of S_y. There are three possible ways that x_i and y_j can be aligned; (a) x_i can align with y_j, which gives a match or mismatch; (b) x_i is aligned with a gap; or (c) y_j is aligned to a gap. Since the matrix is built recursively, in order to calculate $F(i, j)$, the previous states $F(i-1, j-1)$, $F(i-1, j)$, and $F(i, j-1)$ must be known beforehand [7].

2.2 Sum-of-Pairs (SP) Score

Carrillo and Lipman [5] first introduced the sum-of-pairs (SP) score function, which defines the scores of a multiple alignment of *N* sequences as the sum of the scores of the *N(N-1)/2* pairwise alignments [5, 7]. Although SP score function has been widely used to evaluate MSA, it doesn't really provide any biological or probabilistic justification. Each sequence is scored as if it is descended from the *N-1* other sequences instead of a single ancestor. As a result, evolutionary events are often overestimated. The problem worsens as the number of sequences increases [7]. A weighted SP score function has been proposed to partially compensate the problem [1, 8]. Moreover, despite the simplicity of the SP score function, its sheer running time and space consumption makes it impractical even for modestly sized sets of short sequences. Specifically it has been shown that the problem of computing MSA with optimal SP score is NP-hard [22]. Several fast approximations and divide-and-conquer approaches [18] have been proposed to overcome the computational complexity. The following figure shows the mathematical formulation of the weighted SP score function.

$$w(M) = \sum_{1 \le p < q \le k} \left(a_{p,q} \times \sum_{j=1}^{N} s(m_{pj}, m_{qj}) \right)$$

Fig. 2. The SP function, *w(M)*, sums all the pairwise substitution scores in the columns for the sequence pairs *p* and *q*. Each column is evaluated with a scoring matrix. The substitution scoring function, $s(m_{pj}, m_{qj})$, defines all possible alignments for nucleotides p_j and q_j. The function $s(m_{pj}, m_{qj})$ gives the score of the alignment at column *j* for sequence *p* and *q*. The weight, $\alpha_{p,q}$, is intended to balance the overestimation problem in the SP score function [1, 7]

2.3 Clustal W

Clustal W is a commonly used progressive alignment program for biological sequences. It is based on a heuristic algorithm and therefore cannot always find the optimal alignments. Clustal W exploits the fact that homologous sequences are evolutionarily related. It builds up multiple alignments progressively with a series of pairwise alignments moving from the leaves upward in a guide tree that estimates the phylogeny of the sequences [9]. Clustal W first aligns regions of identical or highly conserved residues and gradually adds in more distance ones [21]. This approach is sufficiently fast and allows Clustal W to alignment virtually any number of sequences. Although Clustal W doesn't always find the optimal alignments; however, in most cases those alignments at least give a good starting points for further automatic or manual refinement. This type of alignment is generally useful for the study of identifying regions that are highly conserved. The alignments can be further improved through sequence weighting, positions-specific gap penalties and choice of weight matrix [20]. Clustal W nonetheless suffers two major problems, the local maxima and the choice of alignment parameters.

The local maxima problem stems from the nature of the progressive alignment strategy. As the algorithm follows the guide tree and merges sequences together, the solution is never guaranteed to be globally optimal, as defined by some overall mea-

sure of alignment quality [9, 16, 20]. Any misaligned regions made early in the alignment process cannot be corrected later as new information from other sequences is introduced. This problem is frequently a result of an incorrect branching order in the guide tree. The only way to correct this is to use an iterative or stochastic sampling procedure such as bootstrapping [20]. The choice of alignment parameters is another problem in Clustal W. If parameters are not chosen appropriately, alignments will not converge to a globally optimal solution [7, 20]. For closely related sequences, any reasonable scoring matrices should work fine because matches usually receive the most weights [11]. Therefore, when matches dominate an alignment, almost any weight matrices will find a good solution. However, when aligning more divergent sequences, scores for gaps and mismatches become narrow and critical because they occur more frequently. Moreover for highly conserved sequences, the range of gap penalties that will find the correct or best possible solution can be very broad. As more and more divergent sequences are added, however, the exact values for gap penalties become critical for success [20]. Our observations have confirmed that this is actually a common problem in most MSA algorithms. Statistically as the number of sequences increases, the expected number of matches in each column also increases. For example, the probability of finding a matching nucleotide in the column of ten sequences is much higher than that of three sequences. If the gap penalty is too low, alignments will generally contain excessive amounts of gaps. It is in general difficult to justify why one scoring matrix is better than the others [7].

3 Design of GA

3.1 Consensus Sequence

The *consensus sequence* [25] is a unique as well as the most interesting and important feature of our GA approach. It is essentially a compact formulation to represent all possible alignments for virtually any given numbers of sequences [4]. The consensus sequence borrows the idea from biology that sometimes it is necessary for certain positions in a sequence to be made ambiguous when some residues simply cannot be resolved during laboratory experiments. A sequence with ambiguity codes is actually a mix of sequences, each having one of the nucleotides defined by the ambiguity at that position. For example, if an `R` is encountered in the sequence, then the sequences in the assortment will have either an adenine or a guanine at that position. The ambiguity enables conserved sequences to be condensed into one single representation. The following figure lists the most commonly used ambiguity codes defined by the Nomenclature Committee of the International Union of Biochemistry (IUB) and the next figure shows a hypothetical alignment with five sequences and illustrates how the ambiguity codes are used to derive the consensus sequence. It is assumed that the optimal alignment is already known for a given evolutionary model. The consensus sequence in essence is a condensed sequence with ambiguity codes that shows what nucleotides are allowed in each column.

Symbol	Description	Symbol	Description
A	Adenine	R	Purines (A, G)
C	Cytosine	Y	Pyrimidines (C, T)
T	Thymine	K	Keto (T, G)
G	Guanine	M	Amino (A, C)
W	A or T	B	C, G, or T
S	C or G	D	A, G, or T
H	A, C, or T	V	A, C, or G
N, X	A, G, C, or T	-	Gap symbol

Fig. 3. The most commonly used DNA ambiguity codes are defined by the International Union of Biochemistry (IUB). The presence of ambiguity generally indicates that some residues can not be resolved during the laboratory experiments. Ambiguity codes also enable sequences to be represented in a more condensed form

3.2 Design of Encoding Scheme

To further enhance the design, our chromosomes are broken into four pieces according to the nucleotide they represent. In other words, our GA uses four parallel chromosomes to represent four different nucleotides. Each chromosome encodes the relative occurrences and locations of a nucleotide and is only evolved with the chromosome that encodes the same nucleotide. The fitness of the entire chromosomes is determined by how well they fit together to derive the final alignment. The geometry of the parallel chromosomes is very similar to the four dimensional hyper-plane. Each dimension is evolved and optimized separately and independently.

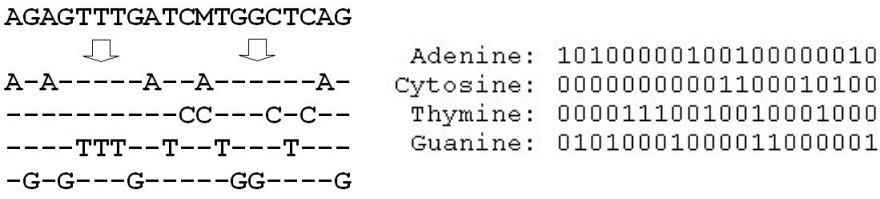

Fig. 4. Sequences are split up into four subsequences according to the nucleotides. Each subsequence only encodes the information where such a nucleotide can be found. Since each nucleotide is individually encoded, any possible ambiguities can be fully represented. Further, chromosomes are represented as binary strings, where `1` signals the existence of such a nucleotide at the given location while `0` signals the absence

The length of chromosomes is difficult to determine precisely. It depends on the evolutionary model as well as the similarity of the sequences. If sequences are highly conserved, chromosomes can be relatively short. Intuitively any two randomly gener-

ated sequences will have at least 25% similarity. For this study, we assumed that sequences have at least 50% similarity to be biologically significant. Therefore, we arbitrarily defined the length of the chromosome to be 1.5 times longer than the longest sequences. For most of our studies, this assumption worked fine. Furthermore, for implementation convenience, each chromosome is further divided into smaller blocks called loci. Biologically, a locus is a block of alleles where genes can be found. The length of a locus is determined by the length of an integer on the hardware platform. The number of loci depends on the length of the chromosome.

3.3 Crossover and Mutation Operations

For this research, we implemented a simple one-point crossover. A point is randomly selected in each locus for each chromosome and alleles are exchanged between two parent chromosomes to form an offspring. An offspring is produced at each generation and then competes with the population. Since each chromosome has separate string for the four nucleotides, the crossover points are chosen separately. Furthermore, we have implemented a bias function for selecting the parent chromosomes from the entire population. The bias function is like an "unfair" randomly number generator. It is essentially a quadratic equation that randomly generates a series of numbers with bias toward the lower indexes. Since the initial population has been sorted in the descending order according to the fitness values, consequently the individuals with higher fitness values are more likely to be selected. Mutation is an important operator that prevents the population from stagnating at local optima [24]. In our implementation, the mutation is only applied to the newly created offspring chromosomes. The GA first calculates the expected number of mutations for each locus in the chromosomes with a random factor. It then iteratively picks random locations on each locus for each of the four parallel chromosomes and changes the alleles. The mutation operator randomly flips the alleles independently on each locus with the binary XOR operator. It inverts the alleles from `0` to `1` or `1` to `0`.

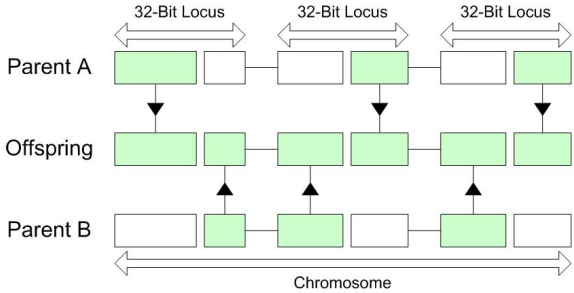

Fig. 5. The crossover operator retrieves the alleles from two parent chromosomes to create an offspring. A point is randomly selected on each locus for each chromosome. The offspring receives approximately a half of alleles from each parent. The length of a locus is 32-bit, which corresponds to the length of an integer on our machines

3.4 Objective Function

The objective function measures the quality of MSA. Therefore, ideally the better the score the more biologically relevant the multiple alignments are. The substitution costs are evaluated using a predefined substitution matrix. The matrix assigns every possible substitution or conservation according to its biological likeliness. We used the nucleic scoring matrix defined by IUB that each match receives 10 points and mismatch 0. The gap penalty is 10.0 for opening and 0.2 extending a gap. The alignment with the highest score is considered a potentially optimal solution. In addition, the objective function subtracts the fitness value with both the mismatch and gap scores multiplied by the numbers of nucleotides that are missing in the alignment for the chromosomes that do not include all nucleotides. Our experiments have confirmed that this strategy worked quite well. The calculation of SP scores for N sequences takes $O(M \times N^2)$ time [4, 26] where M is the average length of the sequences. To further improve the GA performance, we have devised a sequence profiling technique that simplifies the calculations of SP scores.

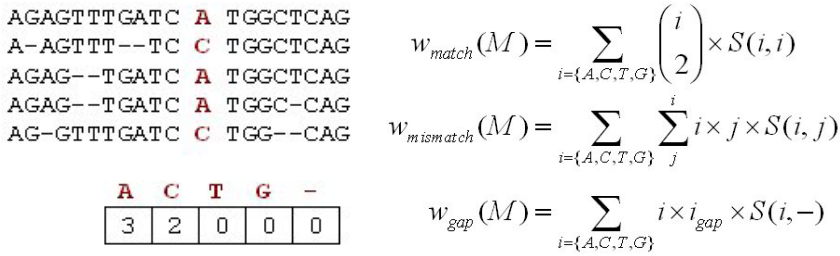

Fig. 6. The figure shows the sequence profile for the column in the red color. The profiling process simply accumulates the frequencies or occurrences of each nucleotide on a given column. The process simplifies the calculations of the SP scores into three smaller tasks and reduces the complexity. Matches are only possible when two identical nucleotides are aligned together. Therefore, the score for matches is the sum of all possible combinations of identical nucleotides multiplied by the matching scores *S(i, i)* from the substitution matrix. The number of mismatches is the sum of all combinations between two different nucleotides. There are only six such combinations. The gap penalties are the sum of all arrangements between each nucleotide and gaps

The objective function first computes the *profile* of the sequences for each column. A profile is simply the occurrences or frequencies of each nucleotide. The profiling process accumulates the occurrences of each nucleotide and reduces the calculations of SP scores into three smaller tasks. Matches are only possible when two identical nucleotides are aligned together. Therefore, the matching score is simply the sum of all possible combinations of the same nucleotides multiplied by the match score in the substitution matrix. The number of mismatches, on the other hand, is the sum of all the combinations of two different nucleotides. There are only six such combinations. The number of gap alignments is derived from the sum of the frequencies of each nucleotide multiplied by the number of gaps. The result is then multiplied by the gap penalty to obtain the overall gap score.

3.5 Alignment Construction

The construction of the final alignment is very similar to that of the DP algorithm [14, 23]. The GA derives the alignment from the last nucleotide to the first and the chromosomes are accordingly decoded backward. If a nucleotide is permitted at a given column, then it is consumed and added in the final alignment. The process moves on to the preceding ones. Otherwise a gap is inserted into the alignment. If no nucleotides are ever used in the column, then the allele is skipped. Alleles that are not used to derive the final alignment are considered the non-coding regions. One of the very interesting and important features of the scheme is that the alleles that are used to derive the alignment do not have to be consecutive. In addition, two different chromosomes can potentially give the same alignments. In other words, the alignment construction process picks the "appropriate" alleles as it moves along. The chromosomes do not have to encode exact bit patterns for the alignments. This makes every allele in the chromosome a potential solution for the alignment. Experiments have confirmed that the GA discovered the optimal alignment, as defined by the substitution matrix, quickly and effectively. The alignments produced by the GA are at least as good as the ones obtained from Clustal W. If the GA is allowed to continue evolving, better alignments are very likely to be found. As the number of sequences increases, the effectiveness of the GA begins to surface. Our experiments have shown that regardless the number of sequences being aligned; the GA performed extremely well and produced alignments with competitive scores.

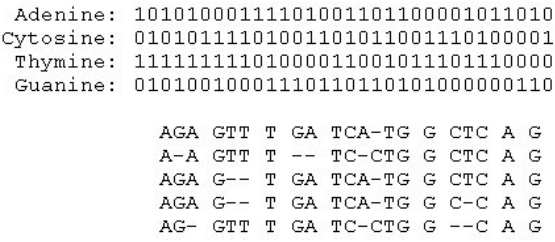

Fig. 7. The figure shows how the final alignment is constructed from the chromosomes. The alignment is derived in the reverse order. The spaces in between are the alleles that did not match up any of the nucleotides in the sequences. If a particular nucleotide does not present in the sequence, a gap is inserted. Chromosomes do not have to encode the exact bit patterns for the alignments. The alignment process simply picks the "appropriate" alleles

4 Experiment Settings

The objective of our experiments was to demonstrate that the GA could scale better as well as produce competitive alignments. For the purpose of this study, we assumed that Clustal W always gave the optimal scores. Sequences were first aligned with Clustal W and the scores were used as the stopping condition for the GA. We applied the standard IUB nucleic scoring matrix and used the gap penalties identical to that of Clustal W. For this study, we have gathered 20 short random DNA sequences of an

average length of 60 base pairs. Sequences were manually verified to have at least 50% similarity. Each chromosome was about 96 bits long and had three loci. The mutation rate was 0.0625 and the average expected number of mutations on each locus was about one. The GA began with a randomly generated population of 64 individuals. The population was first evaluated and sorted in the descending order according to the fitness values. At each generation, the bias function randomly picked two individuals from the population that served as the parent chromosomes. The crossover operator exchanged alleles from two parent chromosomes and created an offspring. Mutation was applied to the offspring repeatedly until the fitness is higher than both parent chromosomes. The offspring then competed with the entire population and removed the individual with the lowest fitness. We gradually increased the number of sequences in each trial. Due to the stochastic nature of GA, all trials were performed at least three times in order to obtain more reliable results.

5 Experiment Results and Discussions

Experiments show promising results for our GA approach. In most cases, the GA outperformed Clustal W and produced better alignments. The number of generations needed to find the optimal solutions remained approximately the same even though the quantity of sequences increased. This is tribute to the fact that the GA was able to utilize the guided search effectively and found the optimal alignments.

During the course of experiments, we have tried various chromosome lengths in order to understand how they affect the performance of the GA. Notably the performance dropped dramatically when the length of sequences reached to 300 base pairs. Further investigation is still needed in order to gain a better understanding. The exact length of the consensus sequence is difficult to determine because it largely depends on the similarity of sequences and the evolutionary model. Our observations revealed that if the consensus sequence is too short, GA frequently failed to converge. On the other hand, if it is too long, the progress becomes extremely slow. In addition, we have confirmed that the SP scoring function was never a good measurement for MSA. If the gap is not heavily penalized, the same score can be easily achieved with more matches but excessive amounts of gaps. The relative difference in score between the correct and incorrect alignments decreases as the number of sequences increases. Clearly this is very counterintuitive and not realistic. The relative difference should increase when more sequences are introduced into the alignment.

Table 1. This table summarizes the numbers of generations needed to find the optimal alignments, at least as good as Clustal W, with various amounts of sequences

Trial	Number of Sequences / Generations					
	10	12	14	16	18	20
1	29,044	36,286	32,225	25,304	35,893	42,805
2	20,835	31,012	22,244	35,447	27,701	46,888
3	26,080	39,906	26,141	32,720	43,989	44,452

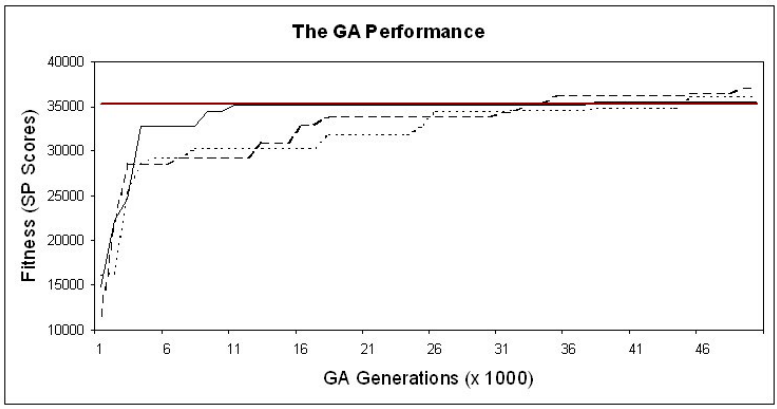

Fig. 8. The figure shows the GA performance on a set of 18 sequences. The GA typically found a good alignment within 50,000 generations. If the GA is allowed to continue evolving, alignments with even higher scores are very likely to be found. The horizontal line indicates the score found by Clustal W, which is about 35,294

6 Future Works

The consensus sequence with GA showed very promising performance and results. For future work, we would like to extend this approach to align protein sequences and implement statistical scoring techniques. In addition, we plan to incorporate the weighting scheme and analyze the impact on the GA performance. We are currently investigating an approach that incorporates several statistical and simulation techniques to try to quantify the significance of alignment scores.

Acknowledgements. Shyu was partially funded by a grant from Proctor and Gamble and Foster was part-ially funded for this research by NIH NCRR 1P20 RR16448.

References

1. Altschul, S.F., Carroll, R.J., and Lipman, D. Weights for data related by a tree. Journal of Molecular Biology, **207**: 647–653 (1989).
2. Altschul, S.F. and Lipman, D. Trees, stars, and multiple sequence alignment. SIAM Journal of Applied Mathematics, **49**: 197–209 (1989).
3. Altschul, S.F., Gish, W., Miller, W., Myers, E. W., and Lipman, D. Basic local alignment search tool. Journal of Molecular Biology, 215 (3):403–410 (1990).
4. Day, W.H. and McMorris, F.R. The computation of consensus patterns in DNA sequence. Mathematical and Computational Model. **17**, 49–52 (1993).
5. Carrillo, H. and Lipman, D. The multiple sequence alignment problem in biology. SIAM Journal of Applied Mathematics, **48**: 1073–1082 (1988).
6. Carroll, S.B., Grenier, J.K., and Weatherbee, S.D. From DNA to diversity: molecular genetics and the evolutionary of animal designs. Malden, MA: Blackwell Science (2001).

7. Durbin, R., Eddy, S., Krogh, A., and Mitchison, G. Biological sequence analysis: probabilistic models of proteins and nucleic acids. Cambridge, UK: Cambridge University (1998).
8. Fogel, D.B. and Corne, D.W. (ed.). Evolutionary Computation in Bioinformatics. San Fran-cisco, CA: Morgan Kaufmann Publishers (2003).
9. Feng, D. and Doolittle, R.F. Progressive sequence alignment as a prerequisite to correct phylogenetic trees. Journal of Molecular Evolutionary, **25**: 351–360 (1987).
10. Graur, D. and Li, W.H. Fundamental of Molecular Evolution, 2nd ed. Sunderland, MA: Sinauer Associates (2000).
11. Wang, L., and Gusfield, D. Improved approximation algorithms for tree alignment. Journal of Algorithms, **25**: 255–273 (1998)
12. Gusfield, D. Algorithms on strings, trees and sequences: computer science and computational biology. New York, NY: Cambridge University Press (1997).
13. Hall, B.G. Phylogenetics trees made easy: a how-to manual for molecular biologists. Sunderland, MA: Sinauer Associates (1997).
14. Needleman, S.B. and Wunsch, C.D. A general method applicable to the search for similarities in the amino acid sequence of two proteins. J. of Mol. Biol. **48**: 443–453 (1970).
15. Sean, R.E. A memory-efficient dynamic programming algorithm for optimal alignment of sequence to an RNA secondary structure. BMC Bioinformatics, **3**: 13 (2002).
16. Sauder, J., Arther, J., and Dunbrack, R. Large-scale of comparison of protein sequence alignment algorithms with structure alignments. Proteins: structures, function, and genetics, **40**: 6–32 (2000).
17. Setubal, J. and Meidanis, J. Introduction to computational molecular biology. Boston, MA: PWS Publishing (1997).
18. Stoye, J., Perry, S.W., and Dress, A.W.M. Improving the divide-and-conquer approach to sum-of-pairs multiple sequence alignment. Applied Mathematical Literature, vol. 10, no. 2, pp. 67–73 (1997).
19. Thomsen, R., Fogel, G.B., and Kirnk, T. A Clustal alignment improver using evolutionary algorithms. Congress on Evolutionary Computation, vol. 1; p. 121–126 (2002).
20. Thompson, J.D., Higgins, D.G., and Gibson, T.J. Clustal W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position specific gap penalties and weight matrix choice. Nucleic Acids Research, **22**: 4673–4680 (1994).
21. Thompson, J. D., Gibson, T.J., Plewniak, F., Jeanmougin, F., and Higgins, D.G. The Clustal X windows interface: flexible strategies for multiple sequence alignment aided by quality analysis tools. Nucleic Acids Research, **24**: 4876–4882 (1997).
22. Wang, L. and Jiang, T. On the complexity of multiple sequence alignment. Journal of Computational Biology, **1**: 337–348 (1994).
23. Waterman, S.M. and Eggert, M. A new algorithm for best subsequence alignments with application to tRNA-rRNA comparisons. J. of Molecular Biology, **197**: 723–725 (1987).
24. Whitley, D. A genetic algorithm tutorial. Statistics and Computing, vol. **4**: 65–85 (1994).
25. Keith, J.M., Adams, P., Bryant, D. Kroese, D.P., Mitchelson, K.R., Cochran, D.A.E., and Lala, G.H. A simulated annealing algorithm for finding consensus sequences. Bioinformaics, vol. 18, no. 11, p. 1494–1499 (2002).
26. Wang, L., Jiang, T. and Gusfield, D. A more efficient approximation scheme for tree align-ment. SIAM Journal of Computational Biology. **30**: 283–299 (2000).