An Improved Genetic Algorithm for Multiple Sequence Alignment Project Report of CSE848, Fall 2010

Yuan Zhang
Rujira Achawanantakun

Department of Computer Science and Engineering,
Michigan State University

December 10, 2010

Abstract

In this project we applied Genetic Algorithm (GA) to solve the problem of Multiple Sequence Alignment (MSA). MSA is one of the most useful tools in bioinformatics. It is widely used to identify conservation of protein domains, RNA secondary structure and classification of biological sequences. However, it is recognized as one of the most challenging tasks in bioinformatics. Most formulations of MSA lead to an an NP-hard problem. Thus most existing methods use heuristics approaches due to the exponential time complexity of global optimization. Genetic Algorithms (GAs) have been proposed to solve this problem and achieved some good performance. However, they suffer from problems such as premature convergence. We implemented conventional GA on this problem using ECJ, a research Evolutionary Computation system written in Java. Also we have made use of the idea of reserved area in our GA and compared its performance with conventional GA. We found that GA with reserved area (GARS) [1] successfully prevented premature and brought in improvement in MSA for short sequences. However, for the dataset with long sequences, there is no significant improvement.

1 Introduction

DNA, RNA and protein sequences contain useful information for biologists to understand the evolution of organism as well as the phenomena of life. Comparing these sequences further improves our understanding of evolutionary relationship and distance between different organisms. Multiple Sequence Alignment (MSA) is a sequence alignment of three or more biological sequences which have an evolutionary relationship. MSA is among the most useful tools in bioinformatics. However, it is an NP-complete problem [2], which is costly computational process in terms of both time and space. Most practical algorithms use heuristic methods because finding global optimization is extremely computationally expensive in most cases.

2 Related Works

There are a number of MSA methods in the literature. A direct method is dynamic programming, which is used to find the global optimal solution. It makes use of a substitution matrix and a gap penalty. However, this method is not practical in that the time complexity increases exponentially as the number of sequences increases. An alternative is the progressive approach, which uses a heuristic search. The most popular progressive alignment method is ClustalW [3]. However, progressive methods heavily rely on initial alignment. So sometimes it compromises accuracy at the cost of improved efficiency. Other MSA methods include iterative methods and Hidden Markov models.

The methods mentioned above lack the ability to effectively search the huge solution space. Thus in recent decades genetic algorithms (GAs) have been proposed to solve MSA problems [4, 5]. They can search through the solution space effectively and generate good alignment results. However, most of these techniques suffer from the problem of premature convergence which leads to a local optima [6, 7]. In this project we make use of the method of reserved area in genetic algorithm [1] and do parameter tuning in order to prevent GA's premature convergence and improve its performance on MSA problem.

3 Methodology

In this project we implemented two sets of programs. One is conventional GA. The other is GA with reserved area (GARS). The only difference between GA and GARS lies in that GARS introduce a concept named reserved area which is a subset of the whole population. In this area, individuals are selected to breed offspring based on their uniqueness values which measure the density around each individual. I divide our methodology into these two methods.

3.1 Conventional Genetic Algorithm

3.1.1 Individual Representation

Each individual in the population is a candidate solution to MSA problem. Each alignment is represented as a two-dimensional matrix of binary strings with each row representing an input sequence and each column representing a position in the alignment. The value of 0 in a bit b_i indicates an inserted gap '-' and the value of 1 indicates a character in the original sequence. The number of bit 1 in each row should be the same with the length of the original sequence. For each row in the matrix, we only allow 20% gaps due to the observation in [4]. Figure 1 is an example of an individual.

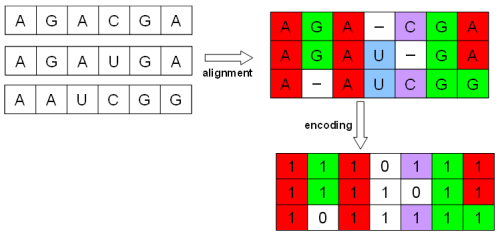

Figure 1: An example of an individual

3.1.2 Fitness Function

The fitness of a multiple sequence alignment is measured by the sum of all of the pairs of characters at each position in the alignment (the so-called sum of pair score). In order to precisely measure the alignment we use the score matrix used in Blastn [8]. The reason is that the score matrix in Blastn depends on the similarity between sequences. Also we use affine gap penalty, which divides gaps into two types: gap opening and gap extension. The total gap penalty for a series of consecutive gaps is the sum of gap opening penalty and all following extensions of gaps. Based on the similarity between sequences in our two datasets, the following score matrix is used (we use A to represent the alphabet of input sequences):

$$M(x,y) = \begin{cases} 6 & \text{if } x,y \in A \text{ and } x = y, \\ 1 & \text{if } x,y \in A \text{ and } x \neq y, \\ 0 & \text{if } x \in A \text{ and y is a gap opening,} \\ 1 & \text{if } x \in A \text{ and y is a gap extension.} \end{cases}$$

3.1.3 Parent Selection

In out experiments, we use tournament selection to choose parents for breeding. The tournament size is 2. In conventional GA, selection is made based on individuals' fitness values. We will see that for GARS, parents are selected based on different metric.

3.1.4 Crossover and Mutation

We use one-point crossover and one-bit mutation as our GA operators. In our individual each sequence is a gene. So one point will be randomly chosen between two sequences of the whole genome. All the sequences above this position will be crossed over. For mutation, first we randomly choose a gene to mutate. For sequences with multiple gaps, we will choose one of them for mutation. Finally we will select a random number which represents the number of positions a gap will shift to the right direction. In implementation,

the actual number of positions for shift will be the mod between the random number and the length of the sequence. Figure 2 is an example of crossover and mutation in our GA.

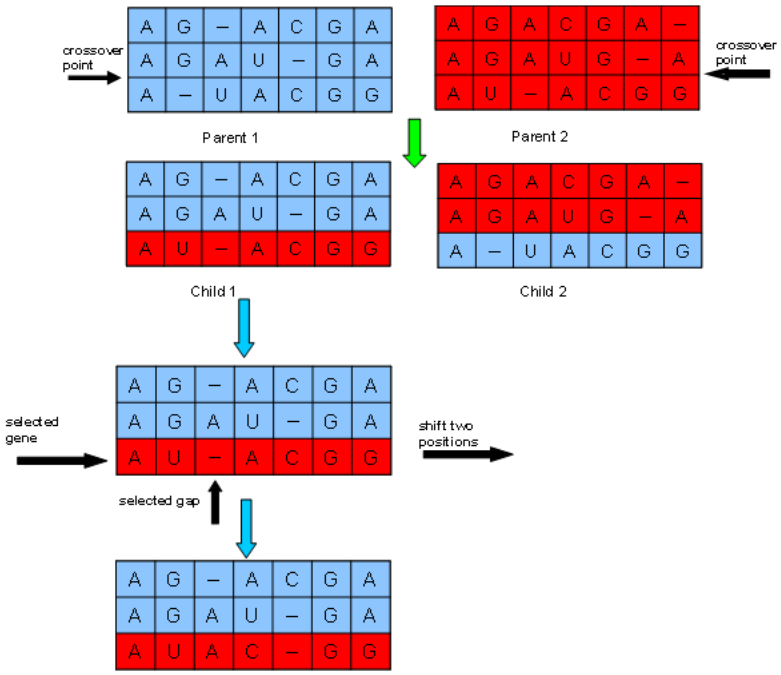

Figure 2: An example of one point crossover and one point mutation

3.2 Survivor Selection

Our replacement policy is to replace the worst k individuals in the current generation individuals using the best k individuals of the offspring. k will be adjusted in multiple runs in order to achieve optimal overall performance.

3.3 Genetic Algorithm with Reserved Area

The reserved area is introduced to prevent premature convergence. The idea is to divide the whole population into a non-reserved area and a reserved area. The size of the reserved area is called the reserve size. The percentage the individuals in the reserved area takes up in the whole population is called reserve rate. In practice we first sort the current population based on their fitness values. Then the individuals which have lowest fitness values will be selected to fill up the reserved area. The idea behind this mechanism is to preserve the diversity of the population and activate the population's capacity to explore the global optimum [1].

In non-reserved area, the selection method is based on the fitness values. However, in reserved area, individuals are measured by their uniqueness which is an indication of the diversity. The uniqueness is calculated using the method proposed in NSGA-II [9]. First we will map individuals to the axis based on their fitness value. For the bordering points, i.e, the individuals with the highest fitness value or lowest fitness value, positive infinity uniqueness will be assigned. For the other points, their uniqueness values will be the absolute value of the subtraction between their two neighbors on the fitness axis. Figure 3 is an example of how we calculate the uniqueness for the four individuals in the reserved area (suppose we only have four individuals in the reserved area).

In parent selection, individuals will be selected based on their uniqueness values. This makes sure that diversity in this area will be maintained. This is the only different between non-reserved area and reserved area.

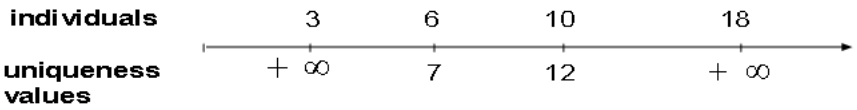

Figure 3: An example of uniqueness values in the reserved area

4 Experiments

4.1 Datasets

We have used two datasets which are RNA sequences from two RNA families, TAR and IRES_PICO from from The Balibase 2.1 database [10]. Balibase is a standardized set of benchmark reference multiple sequence alignments. These datasets are used as input to our multiple sequence alignment program using GA. Each data set contains five sequences with the same length. The datasets were chosen to cover a range of different sequence lengths. The details of the datasets are shown in Table 1.

Dataset	Sequence length	Sequence name
TAR	57	L28864.1_329-385
		M93259.1_9532-9588
		AF443088.1_8897-8953
		AF196710.1_461-517
		AJ286133.1_8742-8798
IRES_PICO	252	AF230973.1_399-650
		D00627.1_394-645
		AF524867.1_393-644
		AY186745.1_373-624
		AJ295195.1_354-605

Table 1: Experimental Datasets

4.2 Implementation

The MSA experiment was conducted using GA parameters shown in Table 2. Two datasets used the same value for all parameters except the population size. The first dataset, TAR, has a smaller population size because the number of possible alignments is smaller than the second dataset, IRES_PICO.

Parameter	Dataset 1	Dataset 2
Population size	50	100
Generations	30000	
Selection strategy	Tournament selection	
Tournament size	2	
Crossover operator	One point	
Crossover probability	0.8	
Mutation operator	One bit	
Mutation probability	0.01	

Table 2: GA parameters

We implement our GA using ECJ version 19, which is a Java-based evolutionary computation research system (http://cs.gmu.edu/ eclab/projects/ecj/). By taking advantage of inheritance from base classes of ECJ, we implemented our species, individuals, genes, fitness function, parent selection, crossover, mutation and replacement. Since ECJ relies on the runtime parameters, we wrote the parameter file to guide the system to make decisions, from classes it loaded to the evolution settings which used during run time. The

parameter file was written using the basic parameter file as parent parameter file. Moreover, we reset some parameters to match our experimental design. Also, new parameters were added to the parameter file based on our need. For example, the reserve size and the FASTA input file, etc.

For GARS, we choose three different reserve rates, 20%, 30% and 40%. For conventional GA, we can regard it as a special case of GARS that has a reserve size of 0. This will be used to compare with the performance of GARS to test the performance of reserve area.

The quality of a alignment is measured by the similarity between the test alignment and the reference alignment. We used the program called COMPALIGN [11] which calculated the fractional sequence-identity between a trusted alignment and a test alignment. In this case, a trusted alignment is an alignment from Balibase and the test alignment is an alignment by our GA or GARS. The reference alignments of two selected datasets do not have any gap. Thus, we allow gaps to align together at the same position during GA process. This alignment may not a pragmatic alignment. However, in the last step we removed gaps at these positions before we compared it with a reference alignment.

4.3 Experimental Results

The experiments were run on HPCC. It took about 5 minutes to complete 30000 generations for each run on the first dataset, which consisted of shorter sequences. For the other dataset which was consisted of longer sequences, it took longer time. At first we tried from 1000 generations to 5000 generations. We found that the second dataset did not converge at this point. We then tried 70000 generations and it took us around 50 minutes. We observed from the fitness curve of the second dataset that and found that around 30000 generations the fitness curve became relative smooth. So we chose 30000 generations as a balance between running time and optimization. The average running time for 30000 generations was approximately 15 minutes. The final results are shown in Table 3. The fitness curves for the two datasets are shown in Figures 4 and 5.

Dataset	Best result	None-reserved area	Reserved area size		
			20%	30%	40%
TAR	alignment score	3122	3099	3206	3166
	generation	4305	4411	5433	6810
	similarity	0.87	0.88	0.93	0.94
IRES_PICO	alignment score	13457	13073	13212	12347
	generation	29674	29489	29684	29474
	similarity	0.86	0.80	0.83	0.66

Table 3: best results within 30000 generations

Two sets of analyses are presented in our work. The first one is a comparison between the best alignment results from the convention GA and the reference alignment from Balibase. The second one is a comparison between the best solutions we get from GA using various reserve rates and the reference alignment from Balibase.

For the first dataset, all GAs with reserved area converged slower than the conventional GA due to the reserve selection mechanism which preserved variety in new population. Also they ended up with greater alignment quality than the conventional GA. The larger the reserve area, the higher the alignment similarity. For the second dataset, which contains longer sequences, the alignment result using 20% reserve rate outperforms all the other GAs with reserve reserve area. However, there is no obviously improvement brougth by reserve reserve areaon this dataset. Conventional GA even has better alignment than GAs with different reserve rates. So reserve area mechanism did not work well for this dataset.

5 Conclusion and Future Work

We have presented a detailed implementation of multiple sequences alignment using GA. Also we have made use of the idea of reserved area [1] to address the premature convergence problem which most GAs suffer from in MSA problems. The results show that for the dataset with short sequences, GA with reserved area successfully maintains population diversity and alleviates premature convergence during evolution process.

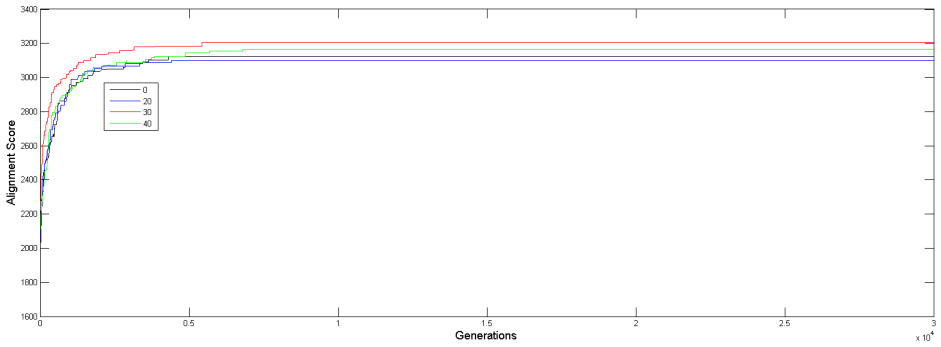

Figure 4: Fitness curves for the first dataset

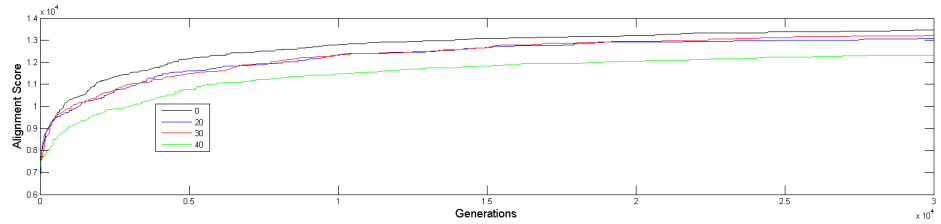

Figure 5: Fitness curves for the second dataset

Thus, the overall performance of GA is improved. However, the reserved area mechanism did not show significant advantage over traditional GAs for long sequences. We think that this happened because uniqueness was calculated based on a single metric, which is the alignment score in our implementation. This cannot always accurately reflect differences between individuals. For long sequences which generally have higher alignment scores, the inaccuracy will be amplified.

In current work, we use the input sequences of the same length, which is not pragmatic in real problems. In future work, we will allow sequences of various length as the input. Also, other methods can be used to preserve diversity for the population. For example, we can use clustering methods to find the outliers in the individuals. These outliers should be preserved during a specific generations to avoid the whole population converge to one cluster too rapidly, which may lead to premature convergence.

References

[1] Yang Chen, Jinglu Hu, Kotaro Hirasawa, and Songnian Yu. GARS:an improved genetic algorithm with reserve selection for global optimization. In *Proceedings of Genetic and Evolutionary Computation Conference (GECCO'07)*, pages 1173–1178, 2007.

[2] Wang L and Jiang T. On the complexity of multiple sequence alignment. *Comput Biol*, 4:337–48, 1994.

[3] Gibson TJ Thompson JD, Higgins DG. Clustal w: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, positions-specific gap penalties and weight matrix choice. *Nucleic Acids Res*, 22:4673–4680, 1994.

[4] K. Chellapilla and G. B. Fogel. Multiple sequence alignment using evolutionary programming. In *Proceedings of the 1999 Congress on Evolutionary Computation (CEC'99)*, pages 445–452, 1999.

[5] E. A. O'Brien C. Notredame and D. G. Higgins. RAGA:RNA sequence alignment by genetic algorithm. *Nucleic Acids Research*, 25:4570–4580, 1997.

[6] L. Davis. *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, New York, 1991.

[7] D. B. Fogel. An introduction to simulated evolutionary optimization. *IEEE Transactions on Neural Networks*, 5:3–14, 1994.

[8] Altschul SF, Gish W, Miller W, Myers EW, and Lipman DJ. Basic local alignment search tool. *J Mol Biol*, 215:403–410, 1990.

[9] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. Fast and elitist multiobjective genetic algorithm:NSGA-II. *IEEE Transactions on Evolutionary*, 6:182–197, 2002.

[10] Julie Thompson, Frederic Plewniak, and Oliver Poch. Balibase. http://bips.u-strasbg.fr/fr/Products/Databases/BAliBASE/.

[11] Andreas W., Indra M., and Gerhard S. An enhanced rna alignment benchmark for sequence alignment programs. *Algorithms Mol Biol*, 2:19, 2006.