A Hybrid Genetic Search for Multiple Sequence Alignment

Seung-Hyun Moon
School of Computer Science and Engineering,
Seoul National University,
Seoul 151-744, Korea
shmoon@soar.snu.ac.kr

Sung-Soon Choi
School of Computer Science and Engineering,
Seoul National University,
Seoul 151-744, Korea
sschoi@soar.snu.ac.kr

Byung-Ro Moon
School of Computer Science and Engineering,
Seoul National University,
Seoul 151-744, Korea
moon@soar.snu.ac.kr

ABSTRACT

This paper proposes a hybrid genetic algorithm for multiple sequence alignment. The algorithm evolves guide sequences and aligns input sequences based on the guide sequences. It also embeds a local search heuristic to search the problem space effectively. In the experiments for various data sets, the proposed algorithm showed the performance comparable to existing algorithms.

Categories and Subject Descriptors

J.3 [**Life and medical sciences**]: Biology and genetics, Medical information systems; F.2.2 [**Nonnumerical algorithms and problems**]: Sorting and searching

General Terms

Algorithms

Keywords

Bioinformatics, genetic algorithms, local search, multiple sequence alignment

1. INTRODUCTION

Multiple alignment of amino acid or nucleotide sequences is one of the most fundamental tasks in bioinformatics. Formally, the multiple sequence alignment (MSA) problem is to align N sequences, whose lengths are at most L, based on some optimization criterion. The most popular criterion is the sum-of-pairs score (SPS) function. This work introduces a hybrid genetic algorithm for the MSA with high performance and reasonable running time.

2. THE MSA PROBLEM

Let Σ be a set of characters. A pairwise sequence alignment of two sequences S_1 and S_2 is defined as two equal-length sequences S_1' and S_2' over $\Sigma \cup \{-\}$, where the character "$-$" represents a gap and removing all the gaps from S_1' and S_2' gives S_1 and S_2, respectively. To measure the similarity between two equal-length sequences, a scoring matrix and a gap penalty function are used. The scoring matrix $\mathfrak{M}$ defines matching points between all pairs of characters in Σ. The gap penalty function $\mathfrak{P}$ defines penalty for occurrences of the gap characters. Whenever k consecutive gap characters are matched to non-gap characters, the penalty of the form $\mathfrak{p}_{\text{op}} + \mathfrak{p}_{\text{ext}} \cdot (k-1)$ is imposed. Here, $\mathfrak{p}_{\text{op}}$ and $\mathfrak{p}_{\text{ext}}$ represent gap open penalty and gap extension penalty, respectively. The scoring function $\mathfrak{S}(S,T)$ that measures the similarity between two sequences S and T of length L is defined as

$$\mathfrak{S}(S,T) = \sum_{x=1}^{L} 1(S[x] \text{ and } T[x] \in \Sigma) \cdot \mathfrak{M}(S[x], T[x]) - \mathfrak{P}(S,T),$$

where $1(\cdot)$ is the indicator function and $S[x]$ is the x^{th} character of a sequence S. Given two sequences S_1 and S_2, the pairwise sequence alignment problem is to find a pairwise sequence alignment, S_1' and S_2', which maximizes $\mathfrak{S}(S_1', S_2')$.

For a set of N sequences $\mathcal{S} = \{S_1, S_2, \ldots, S_N\}$ over Σ, the multiple sequence alignment is a set of N equal-length sequences $\mathcal{S}' = \{S_1', S_2', \ldots, S_N'\}$ over $\Sigma \cup \{-\}$, where removing all the gaps from S_i' gives S_i. As a measure of evaluating the overall similarity of the multiple sequence alignment, the sum-of-pairs score (SPS) function summates the similarity scores over all pairs of sequences. In other words, the SPS for a multiple sequence alignment $\mathcal{S}'$ is

$$\text{SPS}(\mathcal{S}') = \sum_{1 \le i < j \le N} \mathfrak{S}(S_i', S_j').$$

Given a set of N sequences $\mathcal{S} = \{S_1, S_2, \ldots, S_N\}$ over Σ, the MSA problem for the SPS function is to find a multiple sequence alignment $\mathcal{S}' = \{S_1', S_2', \ldots, S_N'\}$ that maximizes $\text{SPS}(\mathcal{S}')$. In this paper, the multiple sequence alignment problem refers to the MSA problem for the SPS function.

3. THE PROPOSED ALGORITHM

There are MSA algorithms that utilize a pivot sequence to generate multiple alignment by aligning each input sequence to the pivot. Gusfield [1] picked one of the input sequences that has the minimum average distance to the other input sequences as a pivot, and Shyu *et al.* [2] evolved consensus sequences by GA and picked the best one. Our algorithm takes a similar approach to [2] by evolving guide sequences instead of the consensus sequences.

3.1 Guide Sequence

A guide sequence is a sequence consisting of subsets of Σ. For a guide sequence G, $G[x]$ denotes the x^{th} subset, which represents recommended characters in the x^{th} column of the resulting multiple alignment. The scoring function $\mathfrak{S}'(S,G)$ that measures the similarity between a sequence S and a

GECCO'06, July 8–12, 2006, Seattle, Washington, USA.
ACM 1-59593-186-4/06/0007.

guide sequence G of length L is defined as

$$\mathfrak{S}'(S,G) = \sum_{x=1}^{L} \frac{1}{|G[x]|} \cdot 1(S[x] \in G[x]) \cdot \mathfrak{M}(S[x], S[x]) - \mathfrak{P}(S, S'),$$

where S' is any sequence consisting of non-gap characters of length L. Each input sequence S is aligned so that $\mathfrak{S}'(S,G)$ is maximized by dynamic programming. The time complexity of the whole multiple alignment process is $\Theta(L^2 \cdot N)$.

3.2 Local Search Heuristic

Given a guide sequence G, let $\mathcal{S}'$ be a multiple alignment obtained by aligning N input sequences to G and $\mathcal{S}'[x]$ be the x^{th} column of S'. The first step of the local search heuristic is counting characters in each column of $\mathcal{S}'$. The next step is altering the guide sequence based on the counting information of the previous step. If the number of a character c in $\mathcal{S}'[x]$ is less than $\frac{N}{|\Sigma|}$, then c is removed from $G[x]$. On the other hand, if the number of c in $A[x]$ is greater than $\frac{N}{2}$, then c is added to $G[x]$. This induces the guide sequence to strengthen the majority character in each column of multiple alignment and lead high SPS values. The operations are performed over all columns. The final step is aligning input sequences to the altered guide sequence and evaluating its MSA quality by the SPS function. If the altered guide sequence produces better MSA, it replaces the original guide sequence. The time complexity of the local search heuristic is $\Theta(L^2 \cdot N)$.

4. GENETIC ALGORITHM

We use a typical hybrid steady-state genetic algorithm.

- **Chromosome Encoding:** A chromosome represents a guide sequence. It is composed of $|\Sigma|$ binary strings of the same length as the guide sequence. Each binary string corresponds to a character in Σ and the x^{th} value of the string indicates whether the character is included in the x^{th} subset of the guide sequence or not.
- **Initialization:** Binary strings constituting a chromosome are randomly generated. We set the population size to be 20.
- **Parent Selection:** Binary tournament selection.
- **Crossover:** Two-point crossover.
- **Mutation:** We perform 0-1 or 1-0 conversions for each binary character. The mutation rate is set to be 0.1.
- **Local Optimization:** We alter a guide sequence as mentioned in Section 3.
- **Replacement:** We replace the inferior of the two parents if the offspring is not worse than both parents. Otherwise, we replace the worst member of the population.
- **Stopping Condition:** GA stops when there are no improvements over 100 generations.

5. EXPERIMENTAL RESULTS

We generated input sequences in the same way as in [2]. Firstly, a base nucleotide sequence is randomly generated and related sequences are generated by the Jukes-Cantor model with no more than 50% divergence. In this way, five sets of various numbers of sequences, whose lengths are 100 base pairs, were generated.

Table 1: Comparisons of CLUSTAL W and Our GA

	CLUSTAL W		Our GA	
Instance	SPS	CPU	SPS	CPU
N=100	1.6M	4.7M	2.1M	122.2M(52M)
N=200	5.9M	16.8M	8.2M	359.6M(80M)
N=300	10.9M	37.6M	18.5M	505.1M(103M)
N=400	17.5M	65.3M	33.0M	1.1G(125M)

For performance analysis, our GA was compared to well-known MSA software, CLUSTAL W (v1.83). However, we found that CLUSTAL W series (v1.80-v1.83) mistakenly do not produce correct scores in the final evaluation step. Thus, we patched CLUSTAL W (v1.83) so that it works correctly.

Table 1 summarizes the performance of CLUSTAL W and our GA for 20 instances. Each row in the table is average of the five instances each of which has N input sequences. For each instance, quality of the solutions was measured by the SPS function exactly the same as the default settings of CLUSTAL W (v1.83) and computation time was measured by the number of atomic operations. Ours are denoted in the form of $x(y)$, where x and y represent the number of atomic operations to produce its best solution and the number of atomic operations to find a solution at least as good as CLUSTAL W, respectively.

For all the instances, our GA outperformed CLUSTAL W in terms of the scoring functions. The results of our GA were averages of 50 runs. In the experiments, our GA always used more computation time than CLUSTAL W. As N increases, however, our GA has lower increase rate of the computation time than CLUSTAL W. Thus, it seems that our algorithm is more scalable than CLUSTAL W in terms of N.

Also, we found that the number of generations to find the best solution was approximately the same regardless of the number of input sequences. It is consistent with the results in [2]. We think that it is because the size of the search space of guide sequences depends not on the number of input sequences but on the length of input sequences.

Acknowledgment

This work was supported by the Brain Korea 21 Project in 2006. This was also partly supported by grant No. (R01-2003-000-10879-0) from the Basic Research Program of the Korea Science and Engineering Foundation. The ICT at Seoul National University provided research facilities for this study.

6. REFERENCES

[1] D. Gusfield. Efficient methods for multiple sequence alignment with guaranteed error bounds. *Bull. Math. Biol.*, 55(1):141–154, 1993.

[2] C. Shyu, L. Sheneman, and J. A. Foster. Multiple sequence alignment with evolutionary computation. *Genetic Programming and Evolvable Machines*, 5(2):121–144, 2004.