This is to certify that the

dissertation entitled

RELATEDNESS OF BIOLOGICAL SEQUENCES USING ALIGNMENT AND RESTRICTION MAP DATABASES

presented by

Jin Kim

has been accepted towards fulfillment
of the requirements for

Ph.D degree in Computer Science

Z. Ramani K

Major professor

Date 1/16/96

RELATEDNESS OF BIOLOGICAL SEQUENCES USING ALIGNMENT AND RESTRICTION MAP DATABASES

By

Jin Kim

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

Department of Computer Science

1996

Abstract

RELATEDNESS OF BIOLOGICAL SEQUENCES USING ALIGNMENT AND RESTRICTION MAP DATABASES

By

Jin Kim

Comparative analysis of DNA and protein macromolecules has been an important component of biological research. Sequence alignment and restriction map comparison, in particular, have been useful in the study of molecular evolution, RNA folding, and protein structure-function relationships. In this thesis, we have investigated multiple alignment problems involving RNA and protein sequences, and the methods to infer relatedness between sequences using restriction patterns and restriction map databases.

We first consider the multiple sequence alignment problem. Multiple sequence alignment is used to discover an optimal alignment based on defined criteria. Variations of dynamic programming provide most commonly used tools for multiple sequence alignment methods. However, the biggest obstacle to using dynamic programming for multiple sequence alignment has been the high computational complexity. Due to this high computational complexity, dynamic programming cannot be effec-

tively used to align more than three sequences and to apply certain types of cost functions. To overcome the limitations of dynamic programming, an algorithm called MSASA, based on simulated annealing, was developed. MSASA can overcome these limitations.

We then consider the multiple RNA sequence alignment problem to identify possible RNA secondary structures. An algorithm called RNASA, based on simulated annealing, is suggested for multiple RNA sequence alignment. In this algorithm, RNA sequences are aligned based on primary sequence similarity and then realigned based on secondary structure information. Dot matrices generated from intra-sequence comparisons are used to obtain possible common secondary structures. Several strategies to reduce the simulated annealing time are suggested.

Sequence database searches have been used for identifying unknown macromolecules. The whole or a part of the primary sequence information is required to do the sequence database search. However, sequencing a macromolecule is expensive and time consuming. A more crude but faster method, without sequencing an unidentified macromolecule, has been developed. The method is based on restriction patterns of an unknown isolate and restriction map databases. We use a three-phase approach. In the first phase, we obtain a restriction pattern of the unknown organism while analytically deriving the corresponding restriction maps of the sequences in the database. In the second phase, we identify a set of sequences, S, which have restriction maps that are most similar to the unknown macromolecule's restriction pattern. Maximum Site Matching Problem (MSMP) is defined and is proved to be in the class of NP-complete problems. In the third phase, we use the set S to infer biological in-

formation about an unknown macromolecule. We demonstrate the usefulness of this approach by applying it to the rRNA sequences of the Ribosomal Database Project (RDP).

To my family

ACKNOWLEDGMENTS

I wish to express my gratitude and appreciation to my thesis advisor Dr. Sakti Pramanik for his consistent guidance and encouragement right from the beginning, and his financial support. I am grateful for many discussions and invaluable comments he provided.

I would like to thank my guidance committee members, Dr. Moon Jung Chung, Dr. Jon Sticklen, and Dr. Zachary Burton, for their help and guidance. Also I would like to appreciate Dr. Eric Torng, Dr. Jame R. Cole their useful advise.

I must express my heartful thanks to my family Myongye Bang, Hyein, Hyejee and my parents, Yong-Beom Kim and my mother, Yong-Sook Shin for their sincere pray during my studying here. Also, I never forget endless love from my parent-in-law, Hwan-moon Bang, Ji-Young Kim.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Basic Concepts

1.1.1 Sequences

A *sequence* is a linear array of elements that are *symbols* or *letters* of an *alphabet*. Let K=$\{a_1, a_2, \cdots, a_l\}$ be an *alphabet* of l symbols. Then $s = b_1 b_2 \cdots b_n$ is a sequence where $b_i \in K$. For *DNA (RNA) sequences*, the alphabet is $\mathsf{K} = \{\mathsf{A}, \mathsf{T(U)}, \mathsf{G}, \mathsf{C}\}$ where 'A' for adenine, 'T' for thymine (or 'U' for uracil), 'G' for guanine, 'C' for cytosine. For *protein sequences*, the alphabet contains 20 symbols which represent 20 types of *amino acids*.

To sequence a long *fragment* of DNA, it must be cut into small fragments, as present techniques do not permit the sequencing of fragments of more than a few hundred bases. At first the DNA is cut into overlapping fragments. These are then sequenced individually, before being reassembled by searching for the overlapped edges,

to reconstruct the whole sequences.

1.1.2 Similarity of Macromolecules

When two or more sequences are compared, it is not so obvious how to assess the similarities between them. This requires some sort of natural biological *metric* which is a measure of the similarity. As an alternative to similarity, one can score the distance between two sequences in a particular alignment based on a metric. Many different alignments can be generated from multiple sequences maps. We seek an alignment with an *optimal* score from these alignments. This optimal alignment gives us the ability to estimate the biological relatedness between sequences. There are two issues to identify the similarity of sequences.

1. Score function: It should be clear that appropriate similarity scores can only be calculated from a specific model of similarity. The user must either develop his own, or more likely use a set which is already developed. The score function should reflect the specific types of similarity between sequences. For example, if we want to identify the primary similarity of sequences, the optimal alignment based on the score function should give us the best picture of the primary similarity.

2. Algorithm to optimize score function: There are several types of algorithms to score alignments. *Exhaustive methods* guarantee an optimal alignment. *Heuristic methods* are used to find, within reasonable time, good alignments that are not necessarily optimal. Users should decide the algorithm to be applied for a

score function based on their application.

1.1.3 Alignment

The term *alignment* is used to refer to a particular arrangement of two or more sequences in sequence comparisons. It is a pattern matching process that finds a correspondence among the elements of the sequences. Any two macromolecular sequences in a homologous group are usually not identical due to *substitutions*, *insertions* or *deletions* of the elements of the set of macromolecular sequences.

Example 2. Let s_1=AATAG and s_2=AATCAG be two DNA sequences. A possible alignment of s_1 and s_2 is

$$s_1\text{=AAT-AG}$$

$$s_2\text{=AATCAG}$$

where '-' represents an insertion. This alignment represents one possible event in evolution.

1.1.4 Restriction Patterns and Restriction Maps

A *restriction enzyme* typically cleaves a fragment of DNA at specific subsequences. A *restriction map* is a set of *ordered* integers which gives the fragment length along the strand of the *restriction sites* at which a restriction enzyme cleaves the DNA.

Example 1. Let s=AGCCCGGCCAAGGCCAA be a DNA sequence. Enzyme *Hae* III identifies the substring GGCC and cleaves in the middle of GGCC. If *Hae* III is applied to the sequence s, s is cleaved to a set of fragments { AGCCCGG, CCAAGG,

CCAA }. Then the restriction map for s is {7, 6, 4}.

Restriction maps are usually constructed prior to sequencing of DNA and many mapped DNAs have never been sequenced. One or more restriction enzymes are applied to cleave the macromolecule into many fragments. At this time only the lengths of the fragments can be identified. This set of *unordered* integers which gives the fragment length is called *restriction pattern*. The lengths of these fragments can be determined by separating them by electrophoresis on a polyacrylamide gel.

1.2 Problem Definitions

1.2.1 Multiple Sequence Alignment

The comparison of more than two sequences of number or letters is common in several fields, such as molecular biology, speech recognition. Sequence comparison is particularly important in molecular biology where it has been critical in the study of evolution and in the analysis of protein structure/function relationships. Multiple sequence alignment is used to find an optimal alignment based on certain criteria. There may be several different sets of optimality. These are definitions for the multiple sequence alignment.

Definitions

A *gap* is the symbol "-".

An *alphabet* $K = a_1, a_2, \cdots, a_l$ is a set of l

symbols containing gaps. it can be nucleic acid or amino acid sequences.

An *element* is a member of the alphabet K.

A *sequences* $= b_1 b_2 \cdots b_n$ is a strings of alphabet if $b_i \in K$ for $i = 1, 2, \cdots, n$.

A ***multiple sequence alignment*** $S = \{s_1, s_2, \cdots, s_k\}$ is a set of sequences of K.

$$s_1 = b_1^1, b_2^1, \cdots, b_n^1$$

$$s_2 = b_1^2, b_2^2, \cdots, b_n^2$$

.

.

.

$$s_1 = b_1^k, b_2^k, \cdots, b_n^k$$

where $b_j^k \in K$ for $j = 1, 2, \cdots, n$ and $i = 1, 2, \cdots, k$. An alignment of the sequences $s_1, \cdots, s_k$ is another set of sequences, $\overline{s_1}, \cdots, \overline{s_k}$ is obtained from s_i by inserting gaps in positions where some of the other sequences have a non-blank character. There exist many different possible alignments. ***The multiple sequence alignment problem is to find out the optimal alignment based on a certain score function for the optimality of an alignment.*** However, different authors may have different sets of score functions for the optimality of an alignment. Sometimes authors have to generate their own score functions for their own purposes. To identify the primary similarity of the sequences, the score function to reflect the primary similarity among sequences should be used. To identify common secondary structures of the sequences, the score function to reflect the common secondary structures among sequences is used.

Authors have been focused in solving the problem by comparing the N sequences simultaneously by dynamic programming using an N dimensional matrix. However,

the complexity of the problem restricts this approach to $N \leq 6$ and $n_i = 200 \sim 300$. Hence various heuristic (possibly sub-optimal) approaches are explored to solve the problem.

1.2.2 Restriction Map Database Searches

Sequence database search is a standard tool in the identification and characterization of new organisms. However, there are many situations in which sequencing of new organisms is not practical. One simple method to substitute sequence database search is to use restriction patterns and restriction map databases. Maximum number of common restriction sites between a new organism and database sequences is calculated and used as a measure of similarity. To compare restriction patterns and restriction maps in the restriction map database, a problem called Maximum Site Matching Problem (MSMP) is defined in this thesis. MSMP is the problem to find maximum common sites between a restriction pattern and a restriction map. In the MSMP, we have as data the restriction pattern from a new organism a a restriction map from a sequence b in a sequence database when a same enzyme is used, say,

Restriction pattern $A = \{a_i : 1 \leq i \leq n\}$ from the digest of sequence a

Restriction map $B = \{b_i : 1 \leq i \leq m\}$ from the nucleotide sequence b

A will be an unordered set, B will be an ordered set. In general A, B will be multi-sets; that is, there may be values of fragment lengths that occur more than once. Also,

$$\sum_{1 \leq i \leq n} a_i = \sum_{1 \leq i \leq m} b_i = L \tag{1.1}$$

Given the above data *the problem is to find orderings for the restriction pattern A such that the number of common sites implied by this ordering is maximum.*

1.3 Previous Studies

1.3.1 Multiple Sequence Alignment Based on Primary Similarity

Existing methods of multiple protein sequence alignment based on primary similarity have time complexity varying from $O(Nn^2)$ to $O(n^N$ where N is the number of sequences and n is the length of the sequences. Most of the methods attempt to find, within a reasonable time, good alignments that are not necessarily optimal. Some methods guarantee optimal alignment in accordance with some pre-specified criteria. The three major multiple sequence alignment methods may be summarized as follows. The main issue in these methods is how to optimize a score function.

Clustering

This approach either constructs an approximate phylogenetic tree to guide the alignment process or arranges the sequences into some sequential order of alignment. Several methods [12, 36, 34] adopt the same hierarchical clustering procedure but ues a different score measure for tree construction. The methods of Feng and Doolittle [20] and Taylor [78, 79] use less rigorous clustering procedures for tree construction. The methods of Barton and Sternberg [7] and Martinez [52] arrange the sequences into a

certain order based on that the sequences are aligned one by one.

Dynamic Programming

This approach refers to the simultaneous comparison of N sequences using an N-dimensional dynamic programming matrix. For example, the algorithm of Needleman and Wunch [59] had been extended directly to the comparison of three sequences using a three-dimensional matrix [43]. However, Murata *et al.* [58] showed that the time complexity of this direct extension would be $O(n^5)$ for three sequences of length n but they were able to reduce it to $O(n^3)$ by using two three-dimensional matrices. A different dynamic programming algorithm of $O(n^3)$ running time for three sequences was also presented by Fredman [26]. Both algorithms use a gap weight (penalty) which is independent of gap length. However, this drawback is removed by Gotoh [28] whose algorithm runs in $O(n^3)$ time and uses a linear gap weighting function.

The algorithm of Murata *et al.*, [58] and the algorithm of Gotoh [28] explicitly specify the weight of simultaneous comparison of three residues. Such a criterion can be extended to the evaluation of an alignment of N sequences, i,e. the cost of aligning the N sequences is taken as the sum of the cost of aligning the $N(N-1)/2$ pairs of sequences. This measure of cost is called the *SP measure (Sum of Pairs measure)* [49].

The algorithm [22] for two sequences looks for the optimal path within a strip of the two-dimensional matrix as defined by an upper bound of the cost of the optimal path. Based on the SP measure, Carrillo and Lipman [10] have proposed a strategy for the alignment of N sequences which is similar to that of Fickett's algorithm. Given the upper bound of the cost of a multiple sequence alignment, Carrillo and Lipman

showed that the upper bound of the alignment cost of each pair of the sequences can be obtained. The upper-bound of the alignment cost of two sequences, say x and y, defines a region on the two-dimensional plane, say (x, y), of the two sequences. Within this region lies the projection of the optimal path of the N sequences onto (x, y). The regions on all two-dimensional planes in turn define the region in which the optimal path in the N-dimensional space would be located. Therefore, the search of the optimal path of N sequences is limited in a certain region of the N-dimensional matrix and the search time is cut down.

The algorithm of Lipman *et al.*, [49] adopts the aforementioned strategy of Carrillo and Lipman. it uses a heuristic procedure to obtain an initial alignment based on which the upper bounds of the pairwise alignment costs can be set. Alternatively, the users may specify any set of bounds. The algorithm has two specific features. One is the use of *quasi-natural gap costs* as proposed by Altschul [1] and the other is the option of either a weighted SP measure or an unweighted SP measure. In the weighted SP measure, different weights are assigned to the pairwise alignment costs following the methods proposed by Altschul *et al.* [1]. The purpose is to discount the dominance of a set of very similar sequences in the multiple sequence alignment. The algorithm of Lipman *et al.* [49] can align six to eight sequences of 200-300 residues which renders it superior to the foregoing three-sequence methods.

The biggest obstacle to using dynamic programming for multiple sequence alignment is its computational requirements of the method. Given these difficulties, heuristics and modified cost function are applied to dynamic programming.

Simulated Annealing

SA is a good heuristic approach to solve combinatorial optimization problems. SA is a version of a successful statistical model of thermodynamic processes for growing crystals that has been transformed into computational terms.

A perfectly homogeneous crystal lattice represents a configuration of solid state material at a global minimum of energy. The solid state material is heated to a high temperature until it reaches an amorphous liquid state. Then, it is cooled very slowly and according to a specific *schedule* of decreasing the temperature. If the cooling is perfect, then the atoms will arrange themselves in a pattern that closely resembles the global energy minimum of the perfect crystal. Thermodynamics teaches that the thermal equilibrium at temperature T is a probability distribution in which a state with energy E is attained with the *Boltzmann* probability

$$e^{-\frac{\Delta E}{T}}.$$

In a theoretical model of what occurs inside the material during the annealing process, states are continually perturbed by the introduction of small random changes of the positions of atoms in the matter. If the result is a state of lower energy, then the state perturbation is unconditionally accepted. If not, then the state perturbation may still be accepted with a probability of $e^{\Delta E/T}$. This probability decreases with the temperature.

Several authors have suggested simulated annealing as an alternative approach to overcome the limitations of dynamic programming. Lukashin *et al.* [50] applied SA to human intro sequences with entropy as a cost function. Ishikawa *et al.* [38] applied

SA to align protein sequences the same cost function as that used in Gotoh [28].

1.3.2 Multiple Sequence Alignment Based on Secondary Structures

It is well-known that secondary structures of RNA are assumed to play an important physiological role. The main issues in multiple RNA sequence alignment is how to generate a score function. Currently, there is no clear global score function for multiple RNA sequence alignment. Most of methods predict local secondary structures and sometimes there is no way to to know the optimality of the alignment obtained.

Manual Method

This approach needs an alignment based on primary similarity. Then, base changes between the compared sequences are noted. And the places where compensating base changes maintain Watson-Crick complementarity between two potential pairing regions is taken as evidence for the existence of a true *stem* at that position. This method is done by hand [11, 32, 54, 63].

Context Free Grammar Method

Stochastic context-free grammars are applied to the problems of multiple alignment of RNA families [18, 70, 73]. This approach is related to use of Hidden Markov Models (HMMs) to model *E.coli* DNA. It incorporates elements of both the thermodynamic and phylogenetic approaches, with emphasis on the latter. The method of Sakakibara *et. al* [70] requires initial knowledge of secondary structures. In contrast, Eddy and

Durbin [18] derive the structure of the grammar directly from unaligned sequences and estimate the probability parameters of the resulting grammar using Expectation Maximization (EM).

1.3.3 Restriction Map Database Search

Some work has been done on constructing restriction maps from restriction patterns [64, 24, 47, 17, 8, 30, 89], but this work typically assumes that there are overlapping fragments.

Several authors [15, 19, 37, 60, 76] studied the restriction site matching probability with given primary similarity of the two sequences. Also simple fragments pattern matching method is used for identification of query sequences. However, this simple pattern matching does not work in some cases. *No* rigorous works for restriction map searches using restriction patterns have been done.

1.4 Our Studies

1.4.1 Multiple Sequence Alignment

In this thesis, we propose two algorithms, MSASA for protein sequences and RNASA for RNA sequences, based on simulated annealing. In chapter 2, multiple sequence alignment for protein sequences is studied. In MSASA, two cost functions, ***natural gap costs*** and ***quasi-natural gap costs***, for protein sequence alignment are discussed as well as the relationships between two methods and two cost functions. We analyzed

the solution sets of multiple sequence alignments and suggest a technique to reduce converging time by confining the solution sets. Another speedup strategy is suggested that involves using fast heuristic algorithm as the high temperature phase. Transition rules to get a new alignment for simulated annealing are generalized, pointingout that any new alignments can be obtained by applying those transition rules. Speedup strategies related to length of the alignment and high temperature phase are discussed. We implement MSASA and show SA can overcome the limitation of dynamic programming by comparing the output alignment from MSASA and dynamic programming

In chapter 3, multiple sequence alignment for RNA sequences is studied. We proposed an algorithm RNASA to align multiple RNA sequences to identify possible secondary structures. Dot matrix generated from *intra*-sequence comparison is expanded to find alignments with maximally overlapped potential secondary structures. When the bases are compared, a certain probability of assigning a hit. This probability is used to make a score function. We use a different transition rule called a ***double shuffle*** in RNASA, which can generate faster convergence to optimal. We show the usefulness of RNASA with experimental results.

1.4.2 Restriction Map Database Searches

In chapter 4, we study the problem of obtaining biological information about a macromolecule isolate using only the restriction patterns of unknown macromolecules and restriction map databases. We propose a three step approach to solve this problem.

In the second step We formulate a *maximum site matching problem* , show this problem is in the class of NP-complete problems, and suggest a heuristic approach to attack this problem. We demonstrate that this three step approach can be used to infer the identification of an unknown macromolecule.

Chapter 2

Inferring Relatedness of Sequences based on Primary Similarity using Multiple Sequence Alignment

Multiple sequence alignment is a useful technique for studying molecular evolution and analyzing structure-sequence relationships. Dynamic programming of multiple sequence alignment has been widely used to find an optimal alignment. However, dynamic programming does not allow for certain types of gap costs, and it limits the number of sequences that can be aligned due to its high computational complexity. The focus of this paper is to use simulated annealing as the basis for developing an efficient multiple sequence alignment. An algorithm called Multiple Sequence Alignment using Simulated Annealing (MSASA). The computational complexity of MSASA is significantly reduced by replacing the high temperature phase of the annealing pro-

cess by a fast heuristic algorithm. This heuristic algorithm facilitates in minimizing the solution set of the low temperature phase of the annealing process. Compared to the dynamic programming approach, MSASA can (a) use natural gap costs which can generate better solution (b) align more sequences (c) take less computation time.

2.1 Introduction

Sequence alignment methods are useful tools to obtain regions of sequence similarities of particular interest. Pairwise alignment has been widely used in sequence alignment, but multiple alignment reveals more information than pairwise alignment. After Needleman and Wunch [59] introduced dynamic programming into pairwise alignment, efforts were made to apply dynamic programming to multiple sequence alignment.

Multiple sequence alignment methods can be divided into two different types of algorithms; heuristic algorithms and exhaustive algorithms. Heuristic algorithms [7, 6, 12, 21, 20, 29, 33, 34, 36, 42, 52, 71, 77, 83] try to find out good but not necessarily optimal alignments within a reasonable time. Most of these heuristic algorithms construct a phylogenetic tree for the alignment of the sequences or assign the sequences to a particular order. The sequences are aligned one by one related to the order.

The exhaustive approach [16, 23, 26, 28, 35, 58, 72, 74] based on dynamic programming tries to compare sequences simultaneously. This approach refers to the simultaneous comparison of N sequences using an N-dimensional dynamic programming matrix [86]. For example, the algorithm of three sequences using a three-dimensional matrix [43].

The algorithm of Fickett [22] for two sequences looks for the optimal path within a strip of the two-dimensional matrix as defined by an upper bound of the cost of the

optimal path., Carrillo and Lipman [10] have proposed a strategy for the alignment of N sequences which is similar to that of Fickett's algorithm. Given the upper bound of the cost of a multiple sequence alignment, Carrillo and Lipman showed that the upper bound of the alignment cost of each pair of the sequences can be obtained. The upper bound of the alignment cost of two sequences, say a and b, defines a region on the two-dimensional plane, say (a, b), of the two sequences. Within this region lies the projection of the optimal path of the N sequences onto (a, b). The regions on all two-dimensional planes in turn define the region in which the optimal path in the N-dimensional space would be located. Therefore, the search of the optimal path of N sequences is limited in a certain region of the N-dimensional matrix and the search time is cut down. This approach guarantees optimal alignment. Although variations of dynamic programming have been widely used to derive optimal alignments, there are certain limitations.

One important problem in multiple sequence alignment is to define substitution costs and gap costs. In pairwise alignment, researchers define substitution costs and gap costs and try to minimize or maximize the total cost. Gap costs and substitution costs in multiple sequence alignment should be defined by using the same rationale as used in pairwise alignment. Altschul [1] analyzed several types of gap cost and substitution cost for multiple alignments. He pointed out that previously defined gap costs in a multiple alignment were not clearly tied to their substitution costs. He suggested a natural gap cost which was clearly related to its substitution cost. Lipman *et al.* [49] implemented the Multiple Sequence Alignment (MSA) program

to align more than three sequences using dynamic programming. In MSA, quasi-natural gap costs were used instead of natural-gap costs because natural gap costs for dynamic programming require impractically long computation time [1] . Due to the type of gap costs used, MSA cannot guarantee producing an optimal multiple alignment in some special cases.

Another problem in expanding dynamic programming to multiple sequence alignment is its high computational complexity. In pairwise alignment, the computational complexity is $O(m \cdot n)$ where m, n are the lengths of the sequences. But when dynamic programming is used for multiple sequence alignment, its computational complexity becomes proportional to the product of the lengths of the sequences to be aligned. Therefore the exponential growth in computational complexity makes dynamic programming impractical for aligning more than three sequences [27, 58]. Lipman [49] has applied dynamic programming to MSA by reducing the solution space using a heuristic algorithm. By confining the solution space, the MSA program can align four to six sequences of length 200-300 residues using rigorous bounds.

Several authors [38, 50] have suggested simulated annealing (SA) as an alternative approach to overcome the limitations of dynamic programming. SA is a good heuristic method to solve combinatorial optimization problems [46]. Ishikawa *et al.* [38] applied SA to align protein sequences with the same cost function as that used in Gotoh [28]. To reduce the long computation time, they utilized a parallel computer for faster convergence to optimal solution, and discussed temperature parallel algorithm which does not require any temperature scheduling. Lukashin *et al.* [50] applied SA to

human intron sequences with entropy as a cost function.

In this study, we developed a method for multiple sequence alignment called Multiple Sequence Alignment using Simulated Annealing (MSASA). Simulated annealing is a good heuristic method to solve combinatorial optimization problems [46]. Because traditional simulated annealing requires long computation time, several speedup strategies have been incorporated into MSASA. The results generated from MSASA were compared to those from MSA. By applying these speedup strategies we show that MSASA can overcome the problems of high computational complexity and the inability to use natural gap costs in MSA.

2.2 Algorithm

2.2.1 Simulated Annealing

Simulated annealing (SA), introduced by Kirkpatrick [46], is a probabilistic approach that can be used to find a global minimum of a function in combinatorial optimization problems. To apply this algorithm to an optimization problem, a state space $X = \{x_1, \cdots, x_n\}$ and a cost function $C : X \rightarrow R$, where R is the set of real number, should be defined. A real value $C(X)$ should be assigned to each state x. The goal of the optimization problem is to find the optimal state x_{opt} whose score is $min(max)\{x_i \mid 1 \leq i \leq n\}$. Simulated annealing continuously generates a new state x_{new} from a current state $x_{current}$ by applying transition rules and acceptance rules

proposed by Metropolis (1953). The criteria of the acceptance rules are:

1. If $\Delta E \leq 0$, accept a new state x_{new}.

2. If $\Delta E > 0$, accept a new state x_{new} with probability $P(\Delta E) = e^{-\Delta E/T}$ where T is a temperature and $\Delta E = C(x_{new}) - C(x_{current})$ is a cost difference.

Probability $P(\Delta E)$ prevents fixation at local minimum. A state $x_{current}$ is called *local minimum* if there is no state x_{new} in X that is generated from the state $x_{current}$ by applying the single transition rule and that has a lower cost than that of the $x_{current}$.

Temperature T controls a probability to accept a new state x_{new}. Initially, T starts from a high temperature and after every iteration, T decreases to become zero by applying an annealing schedule. The probability of accepting a new state with a higher cost than that of the current state also decreases as temperature T decreases. If a careful annealing schedule and number of iterations are given, SA converges to a global minimum state x_{opt}. The main disadvantage of SA is its requirement for a large amount of computation time. Because SA is based on Monte-Carlo methods which allow for a new state with a higher cost than that of a current state. To reduce this computation time, speedup strategies are used in MSASA.

2.2.2 Cost Function

To be used in sequence alignment, a cost function should be explicitly defined as a measure of overall alignment quality. Altschul (1989) classified several global cost

functions for multiple sequence alignment. These cost functions were composed of substitution costs and gap costs.

Substitution Costs

Substitution costs are the costs for aligning n number of sequences which include costs for aligning letters which represent amino acids with nulls. The algorithm of Murata *et al.* [58] and the algorithm of Gotoh [28] explicitly specifiy that the weight of the simultaneous comparison of three residues is the sum of the weights of the pairwise comparisons of the three residues. Such criterion can be extended to the evaluation of an alignment of N sequences, i.e. the cost of aligning of N sequences is taken as the sum of the costs of aligning the $N(N-1)/2$ pairs of sequences. This measure of cost is called the *SP measure (Sum of Pairs measure)* [49]. Substitution costs used in this study were sum of pairs (SP) substitution costs [5, 28, 58]. In MSA and MSASA, SP substitution costs were used.

Gap Costs

Gaps are maximal strings of consecutive nulls ('-') in one sequence aligned with letters in the other sequences. Gap costs are assessed separately from null costs. Null costs are the substitution costs for aligning individual letters with nulls.

The alignment of two sequences, s_i and s_j, derived from a multiple sequence alignment α is referred as the *projection* of α onto s_i and s_j [1, 3]. If the number of gaps in α is defined as the total number of gaps in all the projections of *alpha* and each

gap is assigned as the total number of gaps in all the projections of α and each gap is assigned a constant cost, then the total gap cost of α will be called a *natural gap cost* [1]. Natural gap costs are defined based on the same rationale that is commonly adopted in defining substitution costs. Altschul [1] points out that previously defined gap costs of multiple sequence alignments are proposed independently of the substitution costs. He argues that, since gap costs and substitution costs together determine an optimal alignment, they should be defined based on the same rationale. He suggested the *natural gap cost* as a gap cost which is clearly related to the substitution cost.

But the amount of information to keep in a cell with dynamic programming to apply natural gap costs increases $O([n/ln(2)]^n\sqrt{n})$ where n is the number of sequences [1]. He defined *quasi-natural gap costs* similarly to natural gap costs except for some special cases that massively reduce the amount of information to keep in a cell. Quasi-natural gap costs were used in MSA instead of natural gap costs.

One important advantage of MSASA over the dynamic programming approach used in MSA is its ability to apply any gap costs, including natural gap costs, in multiple sequence alignment. This is because, unlike dynamic programming MSASA does not require space to keep the information. After applying the transition rule to a current alignment, a completely new alignment can be obtained and all the information to be applied to any specific gap costs can be identified. In contrast, any complete alignment can not be obtained in dynamic programming until all of the computations are finished.

2.2.3 Transition Rule

Several operations can be applied to a current alignment to generate a new candidate alignment. Basically, all the operations are related to change the positions of the nulls ('-') in the sequences. The basic operations are as follows.

- *Insertion* $(i, j, k, direction)$: This operation inserts the k number of consecutive nulls from the left/right (direction) of column j in the sequence i.

- *Deletion* $(i, j, k, direction)$: This operation deletes the k consecutive number of nulls from the left/right (direction) of column j where columns $j - \alpha$ through $j + \beta$ $(\alpha, \beta \geq k)$ make a gap where there are consecutive nulls in a sequence.

- *Shuffle* $(i, j, k, direction)$: This operation shuffles the left/right (direction) nulls from the null column j (including null j) in the sequence i and its left/right (direction) k consecutive characters.

Figure 2.1 shows examples of the move sets. By modifying these move sets, effective

```
A1                  A2
MKQIGGAMGSLA--      MKQIGG--AMGSLA
MKKIGGATGALG--      MKKIGGATGALG--
MK---IGGAMGSLA      MK---IGGAMGSLA

A3                  A4
MKQIGG--AMGSLA      MKQIGGA--MGSLA
MKKIGGATGALG--      MKKIGGATGALG--
MK-IGGAMGSLA--      MK-IGGAMGSLA--
```

Figure 2.1: Examples of the move sets. (a) original alignment A1. (b) new alignment A2 after *Insertion*$(1, 6, 2, right)$ to A1. (c) new alignment A3 after *Deletion*$(3, 5, 2, left)$ to A2. (d) new alignment A4 after *Swap*$(1, 7, 1, right)$ to A3.

move sets for a different type of multiple sequence alignment problem can be obtained. Also move sets can be applied to the different sequences simultaneously. The parameters i, j and *direction* in the move set rules may be randomly determined. But k may be determined by a certain distribution function, for example uniform distribution or inverse function related to the size of k. Only experiment can tell which is the best distribution function for k.

Only one basic operation *swap* is used as a transition rule in MSASA. This operation generated a new alignment from the current alignment.

2.2.4 Solution Set

Definition 1 *An alphabet is a finite set of characters and null('-').*

A sequence is a finite string of characters.

A pseudo-sequence is a finite string of characters and nulls.

A pseudo-alignment of the sequence $a_1, a_2, \cdots, a_n$ *is a set of padded pseudo-sequences* $a_1', a_2', \cdots, a_n'$ *of equal length and the removal of the nulls from* a_i' *generates* a_i.

A null column is a column whose elements are all nulls.

A character column is a column such that at least one of its elements is a character.

An alignment is a pseudo-alignment whose columns are only character columns.

Let l_{a_i} be a length of a sequence a_i. The range of the length l of an alignment of the sequences $a_1, a_2, \cdots, a_n$ can be calculated by the definition of an alignment. The

range l is

$$l_{min} = max(l_{a_i}) \leq l \leq l_{max} = \sum_{i=1}^{n}(l_{a_i}) \tag{2.1}$$

In this paper the length of the pseudo-alignment of the sequences $a_1, a_2, \cdots, a_n$ is confined by the above equation.

Definition 2 *Let S_l be the set of all alignments with the same set of sequences, and each alignment in the S_l has its own cost, and the length of each alignment is l. Then the multiple sequence alignment problem is defined as finding the alignment with the smallest cost in the solution set $S_{l_{min}}^{l_{max}} (= \bigcup_{l_{min} \leq l \leq l_{max}} S_l)$.*

In the SA process, the new pseudo-alignment generated from the current pseudo-alignment, using the transition rule, may have null columns. These null columns do not affect the cost of a pseudo-alignment. Figure 2.1 shows how the null columns can be generated during the SA process. These null columns cannot simply be deleted during the SA process. The removal of the null columns during the SA process reduces the length of the pseudo-alignment and it decreases the size of the solution set. Therefore optimal solution may be removed from the solution set. The null columns can be removed only after the SA process.

Lemma 1 *Let P_l be a set of pseudo-alignments with the same set of sequences such that the length of any alignment in P_l is l. Let p_l be a member of the set P_l. Any pseudo-alignment p_l' in P_l can be generated from a pseudo-alignment p_l by applying a finite number of swap operations.*

Proof: The swap operation does not decrease or increase the number of nulls. It only changes the positions of the nulls in the alignment. Any pseudo-alignment p_l' in the

P_l has same finite length l and same finite number of nulls in each sequence. Only the positions of the nulls in each sequence are different from each other. The nulls in p_l can be moved to the positions of the nulls in p'_l within a finite number of swap operations. So the pseudo-alignment p_l can be transformed to any pseudo-alignment p'_l in the set P_l by applying the swap operations. □

Lemma 2 *The set of alignment $S^l_{l_{min}}$ with same set of sequences can be generated from any alignment p_l by applying a finite number of swap operations.*

Proof: From the lemma 1, P_l can be generated from any p_l by applying the swap operations. The set of pseudo-alignments which have no null columns in P_l is equivalent to S_l. And the set of pseudo-alignments which have k null columns in P_l is equivalent to S_{l-k}. The maximum of k is $l - l_{min}$. Therefore eliminating the null columns from the pseudo-alignments in P_l generates $S^l_{l_{min}}$. □

Theorem *Let two pseudo-alignments with same set of sequences be p_{l_1} and p_{l_2} where $l_1 > l_2$. The alignment set, $S^{l_1}_{l_{min}}$, obtained from the alignment p_{l_1} by applying the swap operations is the superset of the set, $S^{l_2}_{l_{min}}$, obtained from the alignment p_{l_2} by applying limited number of swap operations.*

Proof: From the Lemma 2, the set S_{l_1} and S_{l_2} can be generated from the alignment p_{l_1} and p_{l_2} by applying the swap operations. The set $S^{l_1}_{l_{min}}$ can be formed

$$S^{l_1}_{l_{min}} = S^{l_2}_{l_{min}} \bigcup S^{l_1}_{l_2+1} \tag{2.2}$$

Therefore the set $S^{l_1}_{l_{min}}$ is the superset of the set $S^{l_2}_{l_{min}}$. □

In MSASA, the solution set $S^{l_{init}}_{l_{min}}$ is generated from an initial pseudo-alignment

$p_{l_{init}}$ by applying the swap operations. The theorem shows that the longer initial alignment generates the bigger solution set by applying the swap operations. This bigger solution set increases the probability to find an optimal solution.

2.2.5 Speedup Strategies in MSASA

Heuristic Algorithm as The First Phase

Simulated annealing is composed of roughly two phases : a high-temperature phase and a low-temperature phase. In the high temperature phase, SA gives a high probability to all the states with higher costs than that of a current state. This allows any state in the solution set to be a current state. At a lower temperature phase, SA gives a high probability to states with a lower or not much higher temperature than that of a current state. This allows only the states near a current state to be a current state. The high-temperature phase is similar to a random search, and the low-temperature phase is similar to a greedy local search. Several authors [31, 69, 68, 67] suggest good heuristic algorithm as a first phase and a simulated annealing approach as a second phase for fine optimization to the standard-cell-placement algorithm.

In MSASA, the same approach is used. The output alignment generated from the same heuristic algorithm used in MSA is used as the first phase. This heuristic algorithm is similar to progressive pairwise alignment used in the studies of Waterman and Perlwitz [85], Feng and Doolittle [20]. MSASA can eliminate the high-temperature phase and reduce annealing time by using the output alignment gener-

ated from the heuristic algorithm. Figure 2.2 shows the annealing curve and different starting points. SA time P can be saved when the system starts from point B which

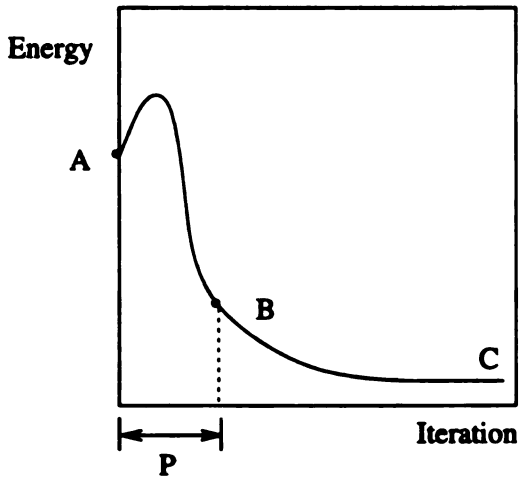


Figure 2.2: Annealing curve (Energy vs. Iteration). A is the starting point in the traditional SA approach. B is the starting point obtained from the fast heuristic approach. C is the minimal point.

is obtained from the fast heuristic approach instead of point A. It is clear that SA time can be reduced if point A is closer to the optimal point C. When the alignment is obtained from the heuristic approach, the initial temperature should be lower than the initial temperature when traditional SA is applied.

Confined Solution Set

To apply dynamic programming for aligning n sequences, a fixed amount of computation is required for each cell of the n-dimensional lattice. The total computational time in dynamic programming is proportional to the product of α and β, $\alpha \cdot \beta$, where α is the fixed number of computations for each cell and β is the size of the n-dimensional lattice. These two factors, α and β, limit the usage of dynamic programming in multiple sequence alignment. Natural gap costs make the factor α too large for aligning

more than five sequences. The factor β becomes too large for aligning more than three sequences with average protein length (about 200-300 residues).

Fickett [22] suggested a way to reduce the solution space in pairwise alignment. He searched the optimal path within a diagonal band of the two-dimensional matrix as defined by an upper bound of the cost of the optimal path. Carrillo and Lipman [10]) expanded the idea to reduce the solution space for aligning n sequences. They calculated the upper bounds of the alignment cost of each pair of the sequences and confined the solution between each pair of the sequences using those upper bounds. Therefore, they could reduce the computation time by applying dynamic programming in only the limited solution space in an n-dimensional lattice.

In MSASA, the solution set is confined to $S_{l_{min}}^{l_{init}}$. Therefore, if the length of the optimal solution is larger than l_{init}, the optimal solution can not be found in the solution set. It is clear that an optimal solution can be found if the solution set $S_{l_{min}}^{l_{max}}$ is used. This solution set can be obtained from a pseudo-alignment $p_{l_{max}}$ by applying the swap operations. Due to its long SA time, it is not practical to use a pseudo-alignment $p_{l_{max}}$ as an initial alignment. Instead of a pseudo-alignment $p_{l_{max}}$ the alignment generated by the progressive pairwise algorithm is used as an initial alignment. The solution set $S_{l_{min}}^{l_{init}}$ can be expanded with longer initial alignment. Once the final alignment which has null columns found in a given solution set is achieved, increasing the solution set further rarely leads to improvement. Such an alignment is optimal for the given solution set.

```
begin
  current_pseudo_alignment ← Output generated from the heuristic algorithm;
  T ← T_initial;
  E_current ← C(current_pseudo_alignment) where C is a cost function;
  final_pseudo_alignment ← current_pseudo_alignment;
  while (T > T_final)
    x ← random(column(1..l)) ;
    for each sequence i in the current_pseudo_alignment
      if column x is a null, then apply the swap operation;
      if column x is a character, then  do nothing;
    end for;
    E_new ← C(new_pseudo_alignment);
    if ((E_new < E_current) then
      current_pseudo_alignment ← new_pseudo_alignment;
      E_current ← E_new;
      if (E_new < E_min) then
        E_min ← E_new;
        final_pseudo_alignment ← current_pseudo_alignment;
      end if;
    end if;
    T ← γ · T, 0 < γ < 1;
  end while;
  Remove null columns from the final_pseudo_alignment;
  Return final_alignment and minimal cost E_min as an optimal alignment;
end
```

Figure 2.3: The MSASA algorithm

2.2.6 The MSASA Algorithm

The simplest data structure for sequences is an array. An alignment is a two dimensional array of symbols in which each row is a one-to-one collinear representation of a molecular sequence. When a swap operation is applied to sequences, only the positions of the nulls are changed in an alignment. MSASA does not require any complicated data structures. The algorithm for MSASA is presented in Figure 2.3.

The space complexity of MSASA is $O(n \cdot l)$, where l is the length of the sequence

and n is the number of sequences. In each iteration the time to apply swap operations to each sequence is $O(n \cdot \Delta l)$ where n is the number of the sequences and Δl is the changed part of the alignment when swap operations are applied.

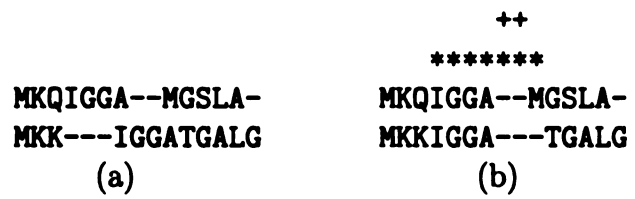


Figure 2.4: Example of the operation *swap*. (a) original alignment (b) alignment after $swap(2, 4, 3, right)$ operation. (*) shows the changed part (Δl) of the alignment and (+) shows the null columns.

Figure 2.4 (b) showed the changed part (Δl) in the alignment. Only the difference of the current costs and new costs of the columns which are affected by the changed part is used to compute new costs. Δl could be referred to as a constant because it is not changed by the number of sequences or length of the sequences to be aligned. The time to calculate the cost of the changed part takes $(n \cdot (n-1) \cdot \Delta l)$. Therefore, the time complexity of MSASA is $O(n^2 \cdot k)$ where k is the number of iterations. The time complexity of MSASA does not depend on the length of the sequences in alignment but to the number of sequences n and the number of iterations k.

2.3 Implementation

The program in this paper was implemented and tested on a Sun SPARC2 running Solaris 2.2 which is a UNIX operating system. The MSASA algorithm was coded in ANSI C (SPARC compiler C 2.0.1). MSASA and MSA were tested and compared in the same environment. MSASA was implemented to produce multiple alignments

and was compared to MSA. The experiments were performed on three serine protease families - chymotrypsin, trypsin and elastase.

Natural gap costs are used as gap costs in MSASA whereas quasi-natural gap costs are used in MSA. The cost of one gap was 8. In MSASA, the PAM-250 matrix [14], derived from a study of amino acid replacements in homologous proteins, was used for substitution costs. The modified substitution costs (17 minus the values in the PAM-250 matrix) were used in both algorithms. The SP substitution costs in MSA have two options, weighted SP substitution costs and unweighted substitution costs. In the weighted SP substitution costs, weights are applied to the pairwise alignments in order to reduce the effect of the dominance of a set of similar sequences in the multiple sequence alignment. In MSASA, both options could be applied. For easy comparison of the two algorithms, only unweighted SP substitution costs were considered.

The alignment from the heuristic procedure of MSA was used as an initial alignment in MSASA. Only the number of nulls of each sequence in the initial alignment are allowed to generate a new alignment. In the swap operation, the maximum value of the parameter k was initially set at 10. The initial temperature T_I was decided by previous experience. At the final temperature, the probability to accept a new state with a higher cost is

$$e^{-\Delta E/T_f} = \epsilon \tag{2.3}$$

where $0 < \epsilon < 1$. This equation is simplified to

$$T_f = -\Delta E / log(\epsilon) \tag{2.4}$$

The minimum cost change, $-\Delta E$, resulting from a swap operation is 1. Empirically, ϵ^{-1} is set to the total number of iterations k. Therefore the final temperature becomes

$$T_f = 1 / log(k) \tag{2.5}$$

The schedule implemented in MSASA is $T_f = T_I \cdot \alpha^k$ where γ is a constant for reducing the temperature. γ can easily be calculated from T_f, T_I and k

$$\gamma = (T_f / T_I)^{1/k} \tag{2.6}$$

2.4 Results

2.4.1 Initial Alignment

A good initial alignment, whose length and cost are as similar as possible to those of an optimal alignment, is crucial to MSASA. In the experiment, the length of the initial alignment generated from the heuristics was longer than that of the optimal alignment. When the final pseudo-alignment did not have null columns, the l_{init} was increased and the SA process was applied again. One way of increasing the length of

an initial alignment is to use a lower gap cost than the gap cost to be applied. If the same final output alignment is obtained, the alignment is the optimal solution within the solution set $S_{l_{min}}^{l_{init}}$. A lower gap cost (3) than the gap cost (8) to be applied in the SA process is used to get a longer initial alignment when aligning the sequences in Figure 2.5. The length of the initial alignment is 277 with gap cost (8) and is 281 with gap cost (8). The longer initial alignment is used to show this method. In the Figure 2.8 thru 2.10, the null columns are not removed from the alignments to show the positions of the final alignments and the lengths of the initial alignments.

Too lower gap costs generates an initial alignment whose length is longer than that of the optimal alignment. MSASA with this longer initial alignment requires longer SA time. Because the solution set becomes bigger and the cost of the initial alignment becomes higher than those of the initial alignment obtained by the gap cost to be applied.

2.4.2 Transition Rules

The operation *swap* in MSASA is used to change the position of the nulls in the alignment. This operation does not change the number of nulls in the alignment in the SA process. The null columns in the alignment gave the effect that the operation could reduce the nulls. Therefore any alignment whose length is between l_{min} and l_{init} could be generated.

The operation *insertion* which could insert the nulls and the operation *deletion*

which could delete the nulls in the alignment were implemented and tested. These operations were possible to change the number of nulls in the alignment. But these operations required too much SA time (order of hours) so that it was not practical to use these operations in MSASA.

2.4.3 Cost Comparison

Quasi-natural gap costs prevent MSA from generating optimal alignment. When a series of nulls with left and right letters completely imposed on the series of nulls in other sequences, quasi-natural costs count one more gap than natural gap costs do (Altschul, 1989). These additional counts prevent generating optimal alignment from MSA. There are no additional counts in MSASA because the natural gap costs were used in MSASA.

MSA and MSASA generate same alignment in some cases. When quasi-natural gap costs are used in both algorithms, both generate the same alignments. And even when natural gap costs are used in MSASA, if there are no completely imposed nulls in the optimal alignment, both algorithms generate the same alignment.

A comparison of the output alignments using the same sequences generated from MSASA and MSA is presented in Figure 2.5.

The alignment A1 was generated from MSA and the alignment A2 was generated from MSASA. The score (9645) of the alignment A2 from MSASA is lower than that (9668) of the alignment A2 from MSA. The marked (*) parts show the difference

A1

```
                                **********          (9663)
IIGGVESIPHSRPYMAHLDIVTEKGLRVICGGFLISRQFVLTAAHC
IVGGTNSSWGEWPWQVSLQVKLTAQR-HLCGGSLIGHQWVLTAAHC
IVNGEEAVPGSWPWQVSLQ--DKTGF-HFCGGSLINENWVVTAAHC
IVGGYTCGANTVPYQVSLN----SGY-HFCGGSLINSQWVVSAAHC
VVGGTEAQRNSWPSQISLQYRSGSSWAHTCGGTLIRQNWVMTAAHC
VVGGTRAAQGEFPFMVRLS--------MGCGGALYAQDIVLTAAHC
```

A2

```
                                **********          (9648)
IIGGVESIPHSRPYMAHLDIVTEKGLRVICGGFLISRQFVLTAAHC
IVGGTNSSWGEWPWQVSLQVKLTAQ-RHLCGGSLIGHQWVLTAAHC
IVNGEEAVPGSWPWQVSLQDKTG---FHFCGGSLINENWVVTAAHC
IVGGYTCGANTVPYQVSLNSG-----YHFCGGSLINSQWVVSAAHC
VVGGTEAQRNSWPSQISLQYRSGSSWAHTCGGTLIRQNWVMTAAHC
VVGGTRAAQGEFPFMVRLSMG--------CGGALYAQDIVLTAAHC
```

Figure 2.5: Alignments of rat mast cell proteinase II, human plasma kallikrein, bovine chymotrypsin, bovine trypsin, pig elastase, and *Streptomyces griseus* trypsin. A1 is the alignment generated from MSA and A2 is generated from MSASA. The score of A1 is 9663 and the score of A2 is 9648. (*) shows different alignments generated from MSA and MSASA.

sequences	*MSA*		*MSASA*		
	score	time	score	time	iteration(million)
4	21244	20 sec	21244	5 min 40 sec	1.3
5	35853	48 min 54 sec	35845	10 min 26 sec	2.0
6	54054	3 hour 18 min 37 sec	54050	16 min 40 sec	2.0

Table 2.1: Comparison of computation time and score in MSA and MSASA

between MSASA and MSA. The difference is due to the different gap costs in MSASA and MSA.

The results of the alignments of the same set of four to six sequences generated from MSA and MSASA are shown in Figure 2.7 through Figure 2.10, and the related scores and running times are in Table 2.1.

The alignment of four sequences (Figure 2.5) does not have the regions creating additional counts for MSA. MSASA and MSA generated the same output in this

MSASA and MSA

```
IVGGTNSSWGEWPWQVSLQVKLTAQR-HLCGGSLIGHQWVLTAAHCFDGLPLQDVWRIYSGILNLSDITKDTPFSQIKEI
IVNGEEAVPGSWPWQVSLQDK--TGF-HFCGGSLINENWVVTAAHCGVT-TSD---VVVAGEFDQGSSSEKIQKLKIAKV
IVGGYTCGANTVPYQVSLN----SGY-HFCGGSLINSQWVVSAAHCYKS-GIQ----VRLGEDNINVVEGNEQFISASKS
VVGGTEAQRNSWPSQISLQYRSGSSWAHTCGGTLIRQNWVMTAAHCVDR-ELT--FRVVVGEHNLNQNNGTEQYVGVQKI

IIHQNYKVSEGNH--DIALIKLQAPLNYTEFQKPICLPSKGDTSTIYTNCWVTGWGFSK-EKGEIQNILQKVNIPLVTNE
FKNSKYNSLTINN--DITLLKISTAASFSQTVSAVCLPSASDDFAAGTTCVTTGWGLTRYTNANTPDRLQQASLPLLSNT
IVHPSYNSNTLNN--DIMLIKLKSAASLNSRVASISLPTSCA--SAGTQCLISGWGNTKSSGTSYPDVLKCLKAPILSDS
VVHPYWNTDDVAAGYDIALLRLAQSVTLNSYVQLGVLPRAGTILANNSPCYITGWGLTR-TNGQLAQTLQQAYLPTVDYA

ECQKR-YQDYKITQRMVCAGYKEGGKDACKGDSGGPLVCKHNGMWRLVGITSWGE--GCARREQPGVYTKVAEYMDWILE
NCKK--YWGTKIKDAMICAG--ASGVSSCMGDSGGPLVCKKNGAWTLVGIVSWGS--STCSTSTPGVYARVTALVNWVQQ
SCKSA-YPG-QITSNMFCAGYLEGGKDSCQGDSGGPVVC--SG--KLQGIVSWGS--GCAQKNKPGVYTKVCNYVSWIKQ
ICSSSSYWGSTVKNSMVCAG-GNGVRSGCQGDSGGPLHCLVNGQYAVHGVTSFVSRLGCNVTRKPTVFTRVSAYISWINN

KTQSS
TLAAN
TIASN
VIASN
```

Figure 2.6: Alignment of human plasma kallikrein, bovine chymotrypsin, bovine trypsin, and pig elastase generated from MSA and MSASA.

case. Because this alignment does not have completely imposed nulls. But in the cases of five sequences and six sequences (Figure 2.6 through Figure 2.10), MSASA can generate the alignment with a lower cost than that of MSA.

2.4.4 Time Comparison

If the lengths of the sequences are short, the number of sequences is small, and the optimal alignment of the sequences does not have regions making additional counts, MSA may generate an optimal alignment faster than MSASA. One of these cases is shown in Figure 2.6. In this case, MSA took a shorter running time than did MSASA.

But in the case of five and six sequences, MSASA required a smaller running time

MSA

```
IVGGTNSSWGEWPWQVSLQVKLT-AQRHLCGGSLIGHQWVLTAAHCFDGLPLQDVWRIYSGILNLSDITKDTPFSQIKEI
IVNGEEAVPGSWPWQVSLQDKTG---FHFCGGSLINENWVVTAAHCGVT----TSDVVVAGEFDQGSSSEKIQKLKIAKV
IVGGYTCGANTVPYQVSL--NSG---YHFCGGSLINSQWVVSAAHCYKS-----GIQVRLGEDNINVVEGNEQFISASKS
VVGGTEAQRNSWPSQISLQYRSGSSWAHTCGGTLIRQNWVMTAAHCVDR---ELTFRVVVGEHNLNQNNGTEQYVGVQKI
VVGGTRAAQGEFPFMVRL--SMG------CGGALYAQDIVLTAAHCVSG----SGNNTSITATGGVVDLQSAVKVRSTKV

IIHQNYKVSEG--NHDIALIKLQAPLNYTEFQKPICLPSKGDTSTIYTNCWVTGWGFSK-EKGEIQNILQKVNIPLVTNE
FKNSKYNSLTI--NNDITLLKISTAASFSQTVSAVCLPSASDDFAAGTTCVTTGWGLTRYTNANTPDRLQQASLPLLSNT
IVHPSYNSNTL--NNDIMLIKLKSAASLNSRVASISLPTSCASAG--TQCLISGWGNTKSSGTSYPDVLKCLKAPILSDS
VVHPYWNTDDVAAGYDIALLRLAQSVTLNSYVQLGVLPRAGTILANNSPCYITGWGLTR-TNGQLAQTLQQAYLPTVDYA
LQAPGYNGT----GKDWALIKLAQPINQPTLKIATTTAYNQGTFT------VAGWGANR-EGGSQQRYLLKANVPFVSDA

ECQKR-YQDYKITQRMVCAGYK-EGGKDACKGDSGGPLVCKHN-GMWRLVGITSWGE--GCARREQPGVYTKVAEYMDWI
NCKK--YWGTKIKDAMICAG---ASGVSSCMGDSGGPLVCKKN-GAWTLVGIVSWGS--STCSTSTPGVYARVTALVNWV
SCKSA-YPG-QITSNMFCAGYL-EGGKDSCQGDSGGPVVCSGK-----LQGIVSWGS--GCAQKNKPGVYTKVCNYVSWI
ICSSSSYWGSTVKNSMVCAGG--NGVRSGCQGDSGGPLHCLVN-GQYAVHGVTSFVSRLGCNVTRKPTVFTRVSAYISWI
ACRSA-YGNELVANEEICAGYPDTGGVDTCQGDSGGPMFRKDNADEWIQVGIVSWGY--GCARPGYPGVYTEVSTFASAI

LEKTQSS
QQTLAAN
KQTIASN
NNVIASN
ASAARTL
```

Figure 2.7: Alignment of human plasma kallikrein, bovine chymotrypsin, bovine trypsin, pig elastase, and *Streptomyces griseus* trypsin generated from MSA.

MSASA

```
                                                    +                                 ++
IVGGTNSSWGEWPWQVSLQVKLTAQR-HLCGGSLIGHQWVLTAAHCFDGL-PLQDVWRIYSGILNLSDITKDTPF--SQI
IVNGEEAVPGSWPWQVSLQDKTGF---HFCGGSLINENWVVTAAHCGVT-----TSDVVVAGEFDQGSSSEKIQK--LKI
IVGGYTCGANTVPYQVSLN--SGY---HFCGGSLINSQWVVSAAHCYKS------GIQVRLGEDNINVVEGNEQF--ISA
VVGGTEAQRNSWPSQISLQYRSGSSWAHTCGGTLIRQNWVMTAAHCVDR----ELTFRVVVGEHNLNQNNGTEQY--VGV
VVGGTRAAQGEFPFMVRLS--MG------CGGALYAQDIVLTAAHCVSG-----SGNNTSITATGGVVDLQSAVK--VRS

KEIIIHQNYKVSEGNH--DIALIKLQAPLNYTEFQKPICLPSKGDTSTIYTNCWVTGWGFSK-EKGEIQNILQKVNIPLV
AKVFKNSKYNSLTINN--DITLLKISTAASFSQTVSAVCLPSASDDFAAGTTCVTTGWGLTRYTNANTPDRLQQASLPLL
SKSIVHPSYNSNTLNN--DIMLIKLKSAASLNSRVASISLPTSCASAG--TQCLISGWGNTKSSGTSYPDVLKCLKAPIL
QKIVVHPYWNTDDVAAGYDIALLRLAQSVTLNSYVQLGVLPRAGTILANNSPCYITGWGLTR-TNGQLAQTLQQAYLPTV
TKVLQAPGYNGTG--K--DWALIKLAQPINQPTLKIATTTAYNQGTFT------VAGWGANR-EGGSQQRYLLKANVPFV

                                                          +
TNEECQ-KRYQDYKITQRMVCAGY-KEGGKDACKGDSGGPLVCKHNG-MWRLVGITSWGE---GCARREQPGVYTKVAEY
SNTNCK--KYWGTKIKDAMICAG---ASGVSSCMGDSGGPLVCKKNG-AWTLVGIVSWGS---STCSTSTPGVYARVTAL
SDSSCK-SAYPG-QITSNMFCAGY-LEGGKDSCQGDSGGPVVCSGK-----LQGIVSWGS---GCAQKNKPGVYTKVCNY
DYAICSSSSYWGSTVKNSMVCAG--GNGVRSGCQGDSGGPLHCLVNG-QYAVHGVTSFVS-RLGCNVTRKPTVFTRVSAY
SDAACR-SAYGNELVANEEICAGYPDTGGVDTCQGDSGGPMFRKDNADEWIQVGIVSWGY---GCARPGYPGVYTEVSTF

MDWILEKTQSS
VNWVQQTLAAN
VSWIKQTIASN
ISWINNVIASN
ASAIASAARTL
```

Figure 2.8: Alignment of human plasma kallikrein, bovine chymotrypsin, bovine trypsin, pig elastase, and *Streptomyces griseus* trypsin generated from MSASA. The null columns (+) are not removed intentionally.

MSA

```
IIGGVESIPHSRPYMAHLDIVTEKGLRVICGGFLISRQFVLTAAHCKGR-EIT----VILGAHDVRKRESTQQKIKVEKQ
IVGGTNSSWGEWPWQVSLQVKLTAQR-HLCGGSLIGHQWVLTAAHCFDGLPLQDVWRIYSGILNLSDITKDTPFSQIKEI
IVNGEEAVPGSWPWQVSLQ--DKTGF-HFCGGSLINENWVVTAAHCGVT-TSD---VVVAGEFDQGSSSEKIQKLKIAKV
IVGGYTCGANTVPYQVSLN----SGY-HFCGGSLINSQWVVSAAHCYKS-GIQ----VRLGEDNINVVEGNEQFISASKS
VVGGTEAQRNSWPSQISLQYRSGSSWAHTCGGTLIRQNWVMTAAHCVDR-ELT--FRVVVGEHNLNQNNGTEQYVGVQKI
VVGGTRAAQGEFPFMVRLS--------MGCGGALYAQDIVLTAAHCVSG-SGN---NTSITATGGVVDLQSAVKVRSTKV

IIHESYNS--VPNLHDIMLLKLEKKVELTPAVNVVPLPSPSDFIHPGAMCWAAGWGKTG-VRDPT-SYTLREVELRIMDE
IIHQNYKV--SEGNHDIALIKLQAPLNYTEFQKPICLPSKGDTSTIYTNCWVTGWGFSK-EKGEI-QNILQKVNIPLVTN
FKNSKYNS--LTINNDITLLKISTAASFSQTVSAVCLPSASDDFAAGTTCVTTGWGLTRYTNANT-PDRLQQASLPLLSN
IVHPSYNS--NTLNNDIMLIKLKSAASLNSRVASISLPTSCA--SAGTQCLISGWGNTK-SSGTSYPDVLKCLKAPILSD
VVHPYWNTDDVAAGYDIALLRLAQSVTLNSYVQLGVLPRAGTILANNSPCYITGWGLTR-TNGQL-AQTLQQAYLPTVDY
LQAPGYNG----TGKDWALIKLAQP------INQPTLKIATTTAYNQGTFTVAGWGANR-EGGSQ-QRYLLKANVPFVSD

KACVD--YRYYEYKFQ-VCVGSP-TTLRAAFMGDSGGPLLCAGV-----AHGIVSYGH----PDAKPPAIFTRVSTYVPT
EECQKR-YQDYKITQRMVCAGYK-EGGKDACKGDSGGPLVCKHN-GMWRLVGITSWGE--GCARREQPGVYTKVAEYMDW
TNCKK--YWGTKIKDAMICAG---ASGVSSCMGDSGGPLVCKKN-GAWTLVGIVSWGS--STCSTSTPGVYARVTALVNW
SSCKSA-YPG-QITSNMFCAGYL-EGGKDSCQGDSGGPVVCSGK-----LQGIVSWGS--GCAQKNKPGVYTKVCNYVSW
AICSSSSYWGSTVKNSMVCAGG--NGVRSGCQGDSGGPLHCLVN-GQYAVHGVTSFVSRLGCNVTRKPTVFTRVSAYISW
AACRSA-YGNELVANEEICAGYPDTGGVDTCQGDSGGPMFRKDNADEWIQVGIVSWGY--GCARPGYPGVYTEVSTFASA

INAVI--N
ILEKTQSS
VQQTLAAN
IKQTIASN
INNVIASN
IASAARTL
```

Figure 2.9: Alignment of rat mast cell proteinase II, human plasma kallikrein, bovine chymotrypsin, bovine trypsin, pig elastase, and *Streptomyces griseus* trypsin generated from MSA.

MSASA

```
                                                                    +
IIGGVESIPHSRPYMAHLDIVTEKGLRVICGGFLISRQFVLTAAHCKGREI-----TVILGAH-DVRKRESTQQKIKVEK
IVGGTNSSWGEWPWQVSLQVKLTAQ-RHLCGGSLIGHQWVLTAAHCFDGLPLQDVWRIYSGIL-NLSDITKDTPFSQIKE
IVNGEEAVPGSWPWQVSLQDKTG---FHFCGGSLINENWVVTAAHCGVTTSD----VVVAGEF-DQGSSSEKIQKLKIAK
IVGGYTCGANTVPYQVSLNSG-----YHFCGGSLINSQWVVSAAHCYKSGI-----QVRLGED-NINVVEGNEQFISASK
VVGGTEAQRNSWPSQISLQYRSGSSWAHTCGGTLIRQNWVMTAAHCVDRELT---FRVVVGEH-NLNQNNGTEQYVGVQK
VVGGTRAAQGEFPFMVRLSMG--------CGGALYAQDIVLTAAHCVSGSGN---NTSITATG-GVVDLQSAVK-VRSTK

                                                                      +
QIIHESYNSVPNL--HDIMLLKLEKKVELTPAVNVVPLPSPSDFIHPGAMCWAAGWGKTGVRDPT--SYTLREVELRIMD
IIIHQNYKVSEGN--HDIALIKLQAPLNYTEFQKPICLPSKGDTSTIYTNCWVTGWGFSKEKGEI--QNILQKVNIPLVT
VFKNSKYNSLTIN--NDITLLKISTAASFSQTVSAVCLPSASDDFAAGTTCVTTGWGLTRYTNAN-TPDRLQQASLPLLS
SIVHPSYNSNTLN--NDIMLIKLKSAASLNSRVASISLPTSCA--SAGTQCLISGWGNTKSSGTS-YPDVLKCLKAPILS
IVVHPYWNTDDVAAGYDIALLRLAQSVTLNSYVQLGVLPRAGTILANNSPCYITGWGLTRTNGQL--AQTLQQAYLPTVD
VLQAPGYNGTG----KDWALIKLAQP------INQPTLKIATTTAYNQGTFTVAGWGANREGGSQ--QRYLLKANVPFVS

EKACVD--YRYYEYKFQ-VCVGS-PTTLRAAFMGDSGGPLLCAG-----VAHGIVSYGH----PDAKPPAIFTRVSTYVP
NEECQK-RYQDYKITQRMVCAGY-KEGGKDACKGDSGGPLVCKHNG-MWRLVGITSWGE--GCARREQPGVYTKVAEYMD
NTNCKK--YWGTKIKDAMICAG---ASGVSSCMGDSGGPLVCKKNG-AWTLVGIVSWGS--STCSTSTPGVYARVTALVN
DSSCKS-AYPG-QITSNMFCAGY-LEGGKDSCQGDSGGPVVCSG-----KLQGIVSWGS--GCAQKNKPGVYTKVCNYVS
YAICSSSSYWGSTVKNSMVCAG--GNGVRSGCQGDSGGPLHCLVNG-QYAVHGVTSFVSRLGCNVTRKPTVFTRVSAYIS
DAACRS-AYGNELVANEEICAGYPDTGGVDTCQGDSGGPMFRKDNADEWIQVGIVSWGY--GCARPGYPGVYTEVSTFAS

TINAVI--N
WILEKTQSS
WVQQTLAAN
WIKQTIASN
WINNVIASN
AIASAARTL
```

Figure 2.10: Alignment of rat mast cell proteinase II, human plasma kallikrein, bovine chymotrypsin, bovine trypsin, pig elastase, and *Streptomyces griseus* trypsin generated from MSASA. The null columns (+) are not removed intentionally.

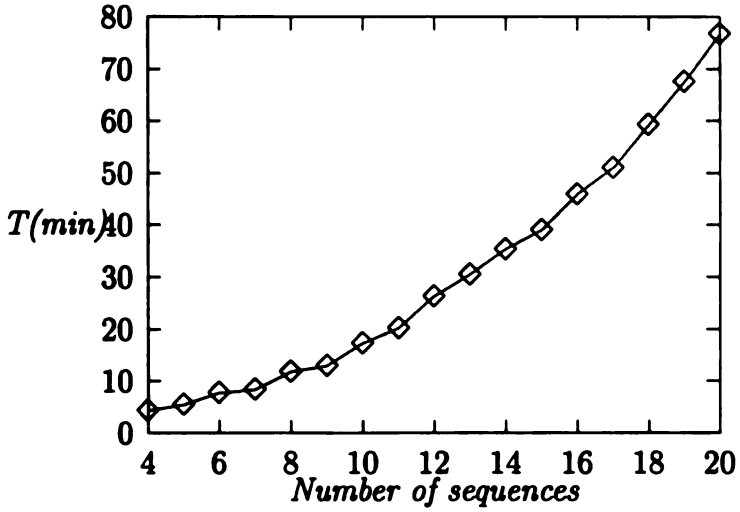


Figure 2.11: Running time for 4 to 20 sequences (1 million iterations)

than that of MSA. The running time to get an optimal alignment in MSASA was not clearly related to the number of sequences. In MSASA, usually 1 to 2 million iterations which took 5 to 16 minutes were enough to get an optimal alignment for four to six sequences.

MSA took an impractically long time to align more than six sequences on a personal workstation. Therefore, it could not be directly compared to MSASA for more than six sequences. In MSASA, 1 to 3 million iterations were enough to get a near-optimal solution when aligning up to twenty sequences. Figure 2.11 illustrates the computation time to run 1 million iterations for aligning several numbers of sequences.

It took about 17 minutes to run 1 million iterations for aligning 10 sequences and about 76 minutes to run 1 million iterations for aligning 20 sequences. Therefore, the running time to align more than six sequences in MSASA is more practical than that in MSA. To reduce the computation time, well-conserved regions may be fixed and

only the regions between the fixed regions can be annealed. By using this divide-and-conquer type approach, the computation time could be greatly reduced.

2.5 Conclusion

It is shown that MSASA based on simulated annealing is more powerful than MSA based on dynamic programming. MSASA could apply natural gap costs, which can generate a better alignment, while MSA could not. MSASA could generate alignments faster and with more sequences than MSA.

Chapter 3

Inferring Relatedness of Sequences based on Secondary Structure using Multiple Sequence Alignment

Multiple sequence alignment has been a useful technique for identifying RNA secondary structures. In this paper, an algorithm for aligning multiple RNA sequences to identify possible secondary structures is presented. In this algorithm, dot matrices generated from *intra*-sequence comparisons are used to obtain possible common secondary structures. A hit probability for dot matrices is calculated and a score function based on this hit probability is defined. Simulated annealing is applied to optimize the score. A solution set for a multiple sequence alignment is introduced and how the number of nulls and length of an alignment affect the solution set is analyzed. A method to reduce the computation time is applied to multiple sequence

alignment based on this solution set. Also several strategies to reduce the simulated annealing time is suggested. *Double shuffle* is used as a transition rule instead of single shuffle. This move set generates faster convergence to optimal. This algorithm was used to find possible common secondary structures in RNA sequences.

3.1 Introduction

Just as the evolutionary relationship in proteins is often seen more in tertiary structure than primary sequence, RNA molecule relatedness is often seen in preserved secondary structures. Much effort has been devoted to finding structural features of RNA. Predictions of RNA secondary structure give useful information about the mechanisms of gene expression, gene evolution and the functions of ribosomes. Several approaches have been used for finding secondary structural features. Phylogenetic analysis of homologous RNA sequences identifies secondary structures which are conserved during evolution [25, 40, 87]. Another approach is to apply thermodynamics to compare the free energy of alternative structures [39, 62, 80, 90]. Context-Free Grammars have been applied to the problems of statistical modeling, multiple alignment, discrimination and prediction of the secondary structures of RNA families [18, 70, 73]. However, RNA secondary structure prediction is not at a stage where perfect prediction is possible. Zuker and Stiegler [90] pointed out that 'A program based solely on conformational rules and thermodynamics will not yield a biologically meaningful folding of a molecule on its own... More and different kinds of additional information must be incorporated into the algorithm as well'. Users need to choose between several methods, each with its own advantages and disadvantages, the one that best fits their particular problem.

Multiple sequence alignment has been a useful tool for identifying sites important in enzyme activity and in gene regulation and in phylogenetic comparisons [72, 84]. Dot-matrix methods have been simple and useful tools for examining sequence simi-

larities. Several multiple sequence alignment methods [4, 81, 82] are dot-matrix based. The dot matrix derived in these methods is from *inter*-sequence comparison based on primary sequence similarity and additional properties of molecular similarity. However, when RNA sequences are distantly related, sequences can not be aligned by just primary sequence similarity. Secondary structure similarity needs to be considered. A dot-matrix generated by *intra*-sequence comparison is one way to identify possible RNA secondary structures in a single molecule [66].

In this paper, we have extended dot-matrix methods based on intra-sequence comparison to find alignments with maximally overlapped potential secondary structures. Individual dot matrices for each sequence are generated and overlapped. Correctly aligned structures appear as diagonal regions with hits in most matrices. When the bases are compared, there is a certain probability of assigning a hit even if the base pairs do not exist as an actual secondary structure. This noise can hide real RNA secondary structures. A sliding diagonal window is applied to reduce noise in the dot matrix. The probability of obtaining an observed number of hits in a cell in a completely unaligned set of sequences can be calculated from the RNA base-pairing relationships. The probability of finding an observed pattern for a window in a completely unaligned set of sequences can be obtained from the hit probability of the cells in the window. This probability was used to design a score function.

Multiple RNA Sequence Alignment using Simulated Annealing (RNASA) is proposed to optimize the score of RNA sequence alignments. Lukashin *et al.* [50] introduced the simulated annealing (SA) technique to multiple sequence alignment for

nucleotide sequences. He discussed the cost function and several other important aspects of SA for multiple sequence alignment. Several authors [38, 45, 44]) extended SA to multiple sequence alignment of amino acid sequence. The SA method was introduced by Kirkpatrick *et al.* [46]. It is a probabilistic approach that can be used to find an optimal state of a function in combinatorial optimization problems. SA starts from an initial state with high temperature. By applying the transition rules and acceptance rules proposed by Metropolis [53], simulated annealing continuously generates a new state from a current state. The criteria of the acceptance rules are:

1. If $\Delta C \leq 0$, accept a new state s_{new}.

2. If $\Delta C > 0$, accept a new state s_{new} with probability $P(\Delta C) = e^{-\Delta C/T}$ where T is a temperature and score difference $\Delta C = C(s_{new}) - C(s_{current})$.

Probability $P(\Delta C)$ prevents a system from fixing at a local minimum. Temperature T affects the probability of accepting a new state s_{new}. In the beginning of the SA process, T starts at a high temperature and T gradually decreases iteration by iteration to become zero by applying an annealing schedule. The probability of accepting a new state with a higher score also gradually decreases as temperature T goes down.

The SA process converges to a global minimum state when a careful annealing schedule and number of iterations are given. The main disadvantage of SA is its need for a large amount of computational time because SA is based on Monte-Carlo methods. To reduce this huge computational time, several speedup strategies including an efficient transition rule, are applied in RNASA. Finally, experimental results obtained

from RNASA are presented.

3.2 Algorithm

3.2.1 Determination of Score Function

Base Pair

The following algorithm aligns conserved regions of possible secondary structure. It does not attempt to determine the lowest energy secondary structure, or chose among conflicting secondary structure possibilities in a specific alignment. Instead it is based on finding an alignment where the possible secondary structure motifs are conserved between sequences. For the purposes of this work, we define a base pair possibility as two positions capable of forming any of the canonical Watson-Crick base pairs (`A-U`, `C-G`, `G-C`, or `U-A`) along with the common non-canonical `G-U` and `U-G` base pairs. In addition to these common base pairs, RNA secondary structures also contain less common base pairs (e.g. `A-G`, `A-A` etc.). Current secondary structure prediction methods do not effectively cover these rare base pairs. Their positions are best established either from the X-ray crystallographic data or from analysis of compensatory changes in unambiguously aligned homologous sequences. In the present alignment method, these rare base pairs are ignored. Since these base pairs are rare by definition, this simplification seems a reasonable compromise.

Dot Matrix

Traditionally a dot matrix for a given RNA sequence is obtained by a square matrix with a dot at the intersection of row i and j $(1 \leq i < j \leq n)$ for the bases i and j of the sequence. A dot matrix from intra-sequence comparison is used to depict the common RNA secondary structures in the sequences to be aligned. This is a rectangular array whose rows and columns are labeled with the bases of a sequence.

A dot matrix with l_{init} x l_{init} is used in RNASA. Length l_{init} is the length of the input alignment, obtained from a progressive pairwise alignment method based on primary sequence similarity. We define a *hit* (1) if the base pair i and j in a RNA sequence in an alignment is one of these base pairs (`A-U`, `C-G`, `G-C`, `U-A`, `G-U`, and `U-G`); otherwise the pair is defined as a miss (0).

Individual dot matrices from each aligned sequence in an alignment are generated and the cells in each individual matrix are calculated. The total number of hits, h_s, in an aligned sequence, s, of an alignment is

$$h_s = (|A| \times |U|) + (|G| \times |C|) + (|G| \times |U|) \tag{3.1}$$

where $|A|, |U|, |G|$ and $|C|$ are numbers of each bases in a sequence. The values of each cell (i, j) in the individual matrices are summed and the sum of each cell (i, j) is recorded in cell (i, j) of a matrix called the *sum* matrix. Clearly the value of cell (i, j) in the sum matrix cannot be larger than the number of sequences n. In the sum matrix, the number of hits, m, in a cell represents the degree of potential common

secondary structure. The larger the number of hits, the higher the possibility of common secondary structure. Figure 3.1 shows an alignment and its sum matrix.

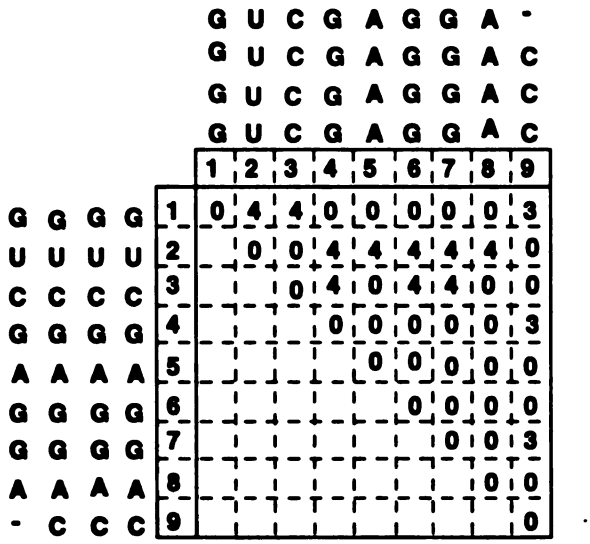


Figure 3.1: Example of an alignment and its sum matrix.

Average Hit Probability

The total number of hits in an aligned sequence is given by equation (1). These hits can be regarded as distributed at random except in regions of real secondary structure. The probability p_s of a hit occurring by chance at any cell (i, j) in an individual matrix of the aligned sequence s is approximately

$$0 \leq p_s = \frac{2 \times h_s}{l^2} < 1 \tag{3.2}$$

where l is the length of the sequence s. For the present study, we use the average probability

$$p = (\sum_{s=1}^{s=n} p_s)/n \tag{3.3}$$

where n is the number of sequences in an alignment as the base pair probability in all the aligned sequences in an alignment. The value of p is less than 1/2 for most RNA sequences.

When individual matrices are overlapped to form a sum matrix, the number of hits at position (i, j) in the sum matrix is between 0 and the total number of bases, n_i, in column i of the alignment. This n_i, may be less than the total number of sequences, N , since some of the sequences may have nulls at position i. We can then approximate the probability of randomly finding that m of n_i bases at a specific position, i, have base pair possibilities (hits) at some other position as:

$$P(i,m) = \begin{pmatrix} n_i \\ m \end{pmatrix} \cdot p^m \cdot (1-p)^{n_i - m} \tag{3.4}$$

The value of $P(i,m)$ becomes smaller as m goes to n_i or m goes to 0. This means the probability that a position will have no potential base pairs is relatively low, as is the probability that a position will have all base pairs.

Sliding Windows on a Sum Matrix

Now the most basic secondary structure motif consists of one or more contiguous base pairs with non-pairing bases at each end. The probability of randomly finding such a conserved motif in the alignment at positions i through $i + w - 1$ with w positions

at another location is simply:

$$\Phi = \prod_{k=1}^{k=w} P(x+k-1, m_k) \tag{3.5}$$

where M = $\langle m_1, \ldots, m_w \rangle$. The number of hits will be equal to n_i for absolutely conserved base pairs, and equal to 0 for positions with no base pair possibility, such as the first non paring position defining the end of a stem region. For incompletely conserved structures, the observed values will be between 0 and n_i and thus the probability of observing such an incompletely conserved motif will be higher. The method we use to optimize the alignment on secondary structure is to find an alignment with a high number of low probability windows.

A window with size w is slid on the diagonals of the sum matrix for this purpose. For example, if the diagonally adjacent cells $(i,j), (i+1,j-1), \cdots, (i+w-1, j-w+1)$ make a window, then the diagonally adjacent cells $(i+1, j-1), (i+2, j-2), \cdots, (i+w, j-w)$ make the next window.

Determination of Score Function

To be used in sequence alignment, a score function should be explicitly defined as a measure of overall alignment quality. A score function must include important secondary structure information so that if the score function is minimized/maximized, the complete secondary structure in the alignment will be identified.

In RNASA the sum of the reciprocal of the probability Φ for all possible windows

on a sum matrix is calculated as a scoring function for simulated annealing as follows.

$$C = \sum_{\textit{all windows}} \frac{1}{\Phi(x, w, M)} \tag{3.6}$$

The $\frac{1}{7}\Phi(x, w, M)$ values for regions of conserved potential secondary structure are so much larger than the expected (random) $\frac{1}{7}\Phi(x, w, M)$ value that they dominate such a sum. A good alignment will have relatively more of these conserved potential secondary structures and will thus have a relatively large C value. The sum C is used as a *score function* in RNASA. In RNASA, at each iteration, a new alignment is generated and C is calculated to find an alignment with maximum score. The goal for RNASA is

$$maximize(C) \tag{3.7}$$

3.2.2 Simulated Annealing to Optimize the Score Function

Heuristic Algorithm as the High-temperature Phase

SA progresses from a *high temperature* phase that approximates a random search to a lower temperature phase approximating a greedy local search. Several authors [38, 45, 44] suggested a good heuristic algorithm could replace the high temperature phase of simulated annealing and provide a substantial speed increase. In the present algorithm, a heuristic primary sequence alignment method similar to the progressive pairwise method [20, 77, 84] is used to provide a preliminary alignment and to partially substitute for the high temperature phase. This preliminary align-

ment can also be used to manually select regions of long RNA sequences that are not well aligned by primary sequence similarity.

Double Shuffle as a Transition Rule

In a RNASA process, new alignments are created from an existing, possibly suboptimal, alignment by introducing random changes according to an optimized transition rule (Figure 3.2).

```
begin double_shuffle(i)
  for j = 1,2
    x ← random(1...l); randomly select a column
      while column(x) ≠ null
        scan left -or- right; (ie x ← x − 1 or x + 1)
        wrap around if necessary;
        return if no nulls are found in sequence i;
      end while;
    y ← random(x + mov_max ··· x − mov_max);
    shift right columns x − 1 thru y
      -or- shift left columns x + 1 thru y;
    column(y) ← null;
  end for;
end double_shuffle
```

Figure 3.2: The transition rule double shuffle

The rule used in the current algorithm allows changing one randomly chosen sequence in the alignment with each iteration. A gap in the sequence is selected and moved to a new position. This process is repeated on the chosen sequence a second time. After both *shuffles*, the score function is recalculated. The reason for applying two shuffles to a sequence in each iteration is to allow both sides of a stem structure to adjust in tandem. This effectively lowers the energy barrier between local points

and allows the system to converge in fewer iterations. If the chosen sequence in the alignment has no null characters, the iteration is skipped. Occasionally, the same gap will be chosen for both shuffle operations, producing the equivalent of a single shuffle move.

Solution Set

The SA process starts from an alignment with length l_{init} produced by the heuristic algorithm. The length l of new alignments produced from the current alignment does not change during the SA process due to the properties of the double shuffle transition rule. Therefore RNASA only generates alignments with length l_{init}.

If l_{a_i} is the length of sequence a_i, the range of potential lengths of an alignment of the sequences $a_1, a_2, \cdots, a_n$ can be calculated. The range l is confined as below.

$$l_{min} = max(l_{a_i}) \leq l \leq l_{max} = \sum_{i=1}^{n}(l_{a_i}) \tag{3.8}$$

Let S_l be the set of all alignments of length l with same set of sequences. Each alignment in the S_l has its own score and the length of each alignment is identically l. Then the multiple sequence alignment problem can be defined as finding the alignment with an optimal score in the solution set $S_{l_{min}}^{l_{max}} (= \bigcup_{l_{min} \leq l \leq l_{max}} S_l)$.

There will be alignments which contain columns whose elements are all nulls (*null columns*). If these null columns are removed from an alignment, the length,l', of the alignment is reduced ($l' < l$). Also, this alignment is a member of the solution set

$S_{l'}$. Thus, the set S_l is a superset of the set $S_{l'}$ where $l' < l$ (Kim *et al.*, 1994).

The number of alignments in S_l can be calculated. Let N be the number of sequences. Let n_i be the number of nulls placed into the sequence a_i to bring it up to the alignment length, l. That is $n_i = l - l_{a_i}$. The total number of alignments β in S_l is

$$\beta_{l,N} = \begin{pmatrix} l \\ n_1 \end{pmatrix} \cdot \begin{pmatrix} l \\ n_2 \end{pmatrix} \cdots \begin{pmatrix} l \\ n_N \end{pmatrix} \tag{3.9}$$

From equations (9) the number of alignments β increases rapidly with each of n_i, l, and N.

To find the optimal alignment, all the alignments in $S_{l_{max}}$ should be investigated. However, the sheer size of $S_{l_{max}}$ prohibits checking all possible alignments. Therefore, if the length of the optimal alignment is larger than l_{init}, the optimal alignment can not be found in $S_{l_{init}}$. In that case, the final alignment from RNASA is near optimal. Solution set can be expanded with longer initial alignment. Increasing the solution set further rarely leads to improvement, so such an alignment is generally optimal for the given solution set. $A2$ and $A4$ in Figure 3.7 are the examples.

The SA time can be reduced by this approach.

Schedule for Temperature.

The schedule implemented in RNASA is $T = T_I \cdot \epsilon^i$ where ϵ is a constant defining the rate of annealing, i the iteration number, T_I the initial temperature, and T the

current temperature. The value of ϵ can easily be calculated from the total number of iterations, k, the final temperature,T_f, and T_I:

$$\epsilon = (T_f/T_I)^{1/k} \tag{3.10}$$

3.2.3 The Complexities of RNASA

Figure 3.3 shows the block diagram of RNASA.

The sum matrix in RNASA is of size l^2. Therefore the space complexity of RNASA is $O(l^2)$. In each iteration, the sum matrix must be updated in the region corresponding to the columns modified by the double shuffle operation. The values of all sliding windows covering these columns must be recalculated. The number of these windows is approximately l times the number of affected columns. In the experiments presented here, the maximum number of columns affected by each shuffle is limited to 10 (Figure 3.2). Since the number of columns affected is essentially constant, the time complexity per iteration is $O(l)$. It should be noted, however, that the number of iterations necessary to assure convergence is dependent on several factors, including l.

3.3 Results

RNASA was implemented and tested on a DEC alpha workstation 3000/400 with 32MB main memory size and DEC OSF/1 V2.0 which is a UNIX operating system.

```
begin RNASA
  current_alignment ← Output generated by the heuristic algorithm;
  /* Generate initial matrix and sum matrix */
  For i=sequence 1 to n
    For each nucleotide pair (j,k) in the sequence i
      matrix(i,j,k) = 1 if pair(i,j) is in complementary set
      matrix(i,j,k) = 0 if pair(i,j) is in not complementary set
      sum_matrix(j,k) = sum_matrix(j,k) + matrix(i,j,k)
    end for;
  end for;
  /* Calculate the score of the sum matrix*/
  C_init = Σ score(window)
  /* Initialize parameters */
  final_alignment ← initial_alignment;
  C_max ← C_init; C_current ← C_init; T ← T_start;
  /* Annealing process */
  while (T > T_end)
    i ← random(sequence(1..n));
    double_shuffle(i) in Figure 3.2;
    Calculate individual matrix i;
    Calculate sum matrix;
    C_new ← C(sum matrix);
    if Metropolice conditions are satisfied then
      current_alignment ← new_alignment;
      if (C_new > C_max) then
        C_max ← C_new;
        final_alignment ← current_alignment;
      end if;
    end if;
    T ← γ · T, 0 < γ < 1;
  end while;
  Return final_alignment, maximum score C_max and sum matrix ;
end RNASA
```

Figure 3.3: The RNASA algorithm

RNASA was written in programming language ANSI C (DEC OSF/1 C compiler). The program will be available on request.

RNASA was implemented to produce multiple sequence alignments for RNA sequences. Experiments were performed on segments of bacterial 16S RNA sequences. rRNA sequences consist of highly conserved regions separated by variable regions. In these variable regions, secondary structure is often more conserved than primary sequence similarity. Several such variable regions were chosen for testing RNASA. A window size of four was applied in all experiments except $B_1, B_2, B_3,$ and B_4 in Figure 3.7.

3.3.1 Alignment and Secondary Structure.

Figure 3.4 shows an alignment of segment of 16S RNA sequences and their possible secondary structures. Three stem regions (`A-A'`, `B-B'`, `C-C'`) could be identified. This alignment is close to the hand alignment in the RDP [51]. From the final sum matrix, we could identify the common secondary structure (Fig 3.5 (b)) and could overlay each aligned sequence onto this common secondary structures. Figure 3.4 (c) and (d) are the possible secondary structures of the sequence `Ehr.bovis` and `Hir.baltic` in the alignment. These possible secondary structures are compared to the proposed secondary structure of `Ag.tumefa7` [32] which is close to the aligned sequences. These structures are matched well.

Figure 3.5 (a) shows an alignment of a segment of ten 16S RNA sequences. The

```
                    A          B          B'     C       C'    A'
                   (((((((( [[[[[[[  ]]]]]]] {{{{   }}}}))))))) ))
Ehr.bovis          aUGAAGAUUagaUCCUCCUuaacAGGAGGGcgCAGUuc-gGCUGGGUCUUgCA
Sym.Mscfu          aUGGAAAUUauaCCUAUUCgaagGGAUAGGguCGGUuc-gGCCGGGUUUCaCA
Sym.Trgr1          aUGGAAAUCauaCCUAUUCgaagGGAUAGGgu-CGgcucggCGCGAUUUUaCA
Sym.Trgr5          aUGGAAAUCauaCCUAUUCgaagGGAUAGGguCGGUuc-gGCCGGAUUUUaCA
Sym.Trgr6          aUGGAAAUCauaCCUAUUCgaagGGAUAGGguCGGUuc-gGCCGGAUUUUaCA
Par.denitr         aUCCCAGG-acc-GGCCCGgagaCGGGUC-uuUCACuucgGUGA-CCUGGaGA
Ros.denitr         aUCCUGUG-cua-ACCCNAgagaUCGGGN-guUCUCgcaaGAGA-CGCAGuGA
Hir.baltic         aUCCUAUG-cu-NACUCCAgagaUGGAGUA--U-C-uucg-G-A-CAUAGuGA
Erb.longus         aUCCUAGGAcgguUUCUG-gaga-CAGACuccUUCCcuucGGGGaCCUAGuGA
R.salexige         aUCCCGGGAcgacUUCCA-gaga-UGGAUuuuUUCAcuucGGUGACCCGGnGA
Ps.diminut         aUGCCUGG-a-cNGCCACGgagaCGUGGCUu-UCCCuucgGGGA-CUAGGaCA
Caw.FWC2           aUGCCUGG-a-cCGCCACAgagaUGUGGNNu-UCCCuucgGGGA-CCAGGaCA
Afp.felis          aUGUCCAGGaccGGUCGCAgagaUGUGACC--UUCUcuucGGAGCCUGGAgCA
Afp.cleve          aUGUCCAGGaccGGUCGCAgagaUGUGACC--UUCUcuucGGAGCCUGGAgCA
```

(a)

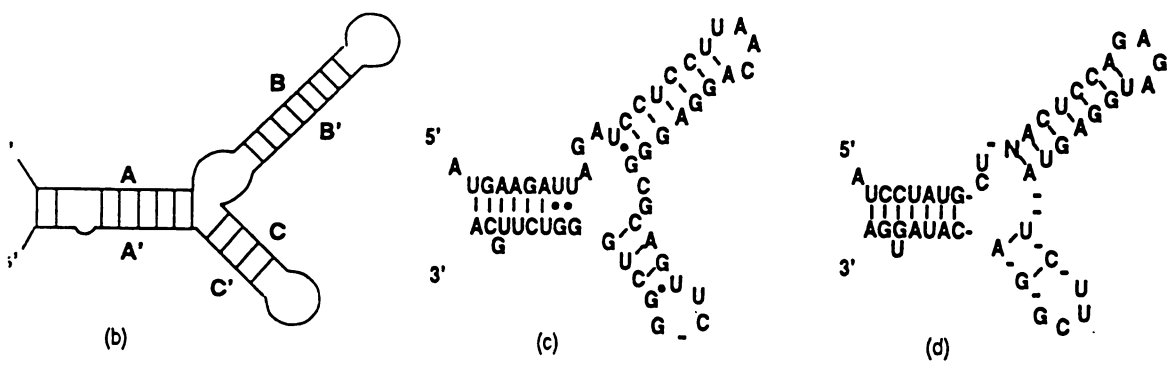
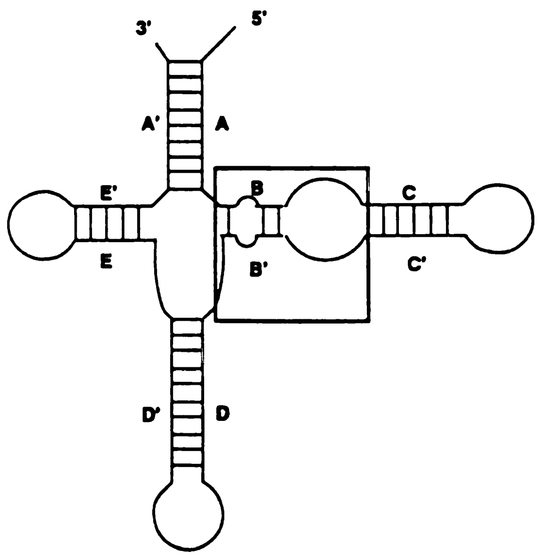


Figure 3.4: (a) Alignment of segments of 15 16S RNA sequences and possible secondary structures. This region corresponds to nucleotides 996 to 1044 in *E.coli* 16S RNA. Left ((, [, {) and right (),], }) parts of the alignment show the possible stem regions (A-A', B-B', C-C'). (b) Possible common secondary structure from the alignment (a). (c) Secondary structure of Ehr.bovis based on (b). (d) Secondary structure of Hir.baltic based on (b). The symbol (•) indictes a G-U base pair and (|) indicates A-U and G-C base pairs. Note that the length of the loop and base pair regions are variable for each sequence.

```
  A         B     C         C'        B'          D              D'          E       E'   A'
 {{{{{{{{< <<    [[[[[[    ]]]]]]     >> >      <<<<<<<<<      >>>>>>>>>     (((((   ))))) }}}}}}}}
 {{{{{{{{BBBBBBBBB[[[[[[    ]]]]]]B'B'B'B'B'B'B'B'<<<<<<<<<      >>>>>>>>>     (((((   ))))) }}}}}}}}
Spi.haloph
uacCUUGAGGUUgGGgauagCCCAUGgaaaCAUGGGguaauaCCgAaugagCUCUAUGUAC-ugug-GUGUAUAGAG-gaaaAGGGCuuugGCCCU-GCCUUGAGau
Spi.bajaca
uguCUUGGGAUGgGGgauagCCCAUGgaaaCAUGGGguaauaCCgCauuagCUNUNUUAUUCauaaGAGUGAUAGAG-gaaaGGCGCuuagGCGUCga-CCUAAGag
Spi.spAntc
uacCCCCUUGUUUGGgauagCUUGUGgaaaCAUAAGguaauaCCUAaugagaUUCUUUUAUCauaaGAUAAUAGAAggaaaGGCGCUuuacAGCGUCGCAAGGGGau
Trp.saccha
uacCCUUUGGaUgGGgauagCCGGUAgaaaUACCGGguaauaCCgAauaaggUCGGUNGUcucggaa-GCCACCGAgg-aaaGCGGCuucgGCCGCg-CCGAAGGaa
Trp.spH1
uacCCUUAGGaUgGGgauagCCACUAgaaaUAGUGGguaauaCCgAaugugg UUACCGGGC-ugug-GCCUNGUAAag-aaaGGAGCu-uuGCUCCg-CCUGAGGau
Spi.stenos
uacCUUUAGGUUgGGgauagCCAUUAgaaaUAGUGGguaauaCCgAauguggUUGCGGGGC-uuug-GCCUUGUAAag-aaaCGAG-cuag-CUCC-GCCUGAAGau
Trp.phaged
uacCCUUAAGaUgGGgauagCUCCUAgaaaUAGGAGguaauaCCgAauac-gCUUAUACGGauaaagCCGUAUAAGga-aaaUAGGCugcgGCCNNg-CUUGAGGau
Lps.sp183
uacCUAGAAGUUgGGgauaaCACAGAgaaaUUUGUGguaauaCCgAaugugaCGGUUCCUG-guag-CAGGGAUUGguaaaa-GCA-gcaa-UGC--GCUUUUAGau
Lpm.illini
ugcCCACGGAUGgGGgauaaCCUUUCgaaaGGAAGGguaauaCCgCauaacACCAUGGUUACcacgGUAAUUAUGGU-caaa-GCA-gcaa-UGC--GUCUGUGGag
Spi.aurant
ugcCUGaCGGaUgGGgauagCUUGUGgaaaCACAGGguaauaCCgAauga-ACUGACGAGACuuugGUUUUGUUAGU--aaaGGUGCuacgGCACCg-CCGaCAGag
```

(a)



(b)

Figure 3.5: (a) Alignment of segments of ten 16S RNA sequences and possible secondary structures. This region corresponds to nucleotides 133 to 229 in *E.coli* 16S RNA. Left ((, <, [, {) and right ( ), >,], }) parts of the alignment show the possible stem regions. (b) Real common secondary structure from the RDP. There are different possible secondary structures in the Region B (in the Box). Note that the length of the loop and base paired regions are variable for each sequence.

possible secondary structure was compared to the proposed secondary structure of `Spi.aurant` [32] which is closely related to the aligned sequences. (second line in the Figure 3.5 (a)). The third line shows the possible secondary structure from the alignment. A secondary structure diagram based on `Spi.aurant` is shown in Figure 3.5 (b). From the alignment and dot matrix, stem regions (`A-A', C-C', D-D', E-E'`) were correctly aligned. However, in the region (`B-B'`), there are different possible stem regions with length=2.

3.3.2 Identification of Secondary Structures

An RNASA generated alignment of one variable region from eight 16S RNA sequences is shown in Figure 3.6. This region corresponds to nucleotides 61 to 106 in *E.coli* 16S RNA [32, 61, 88, 87].

Alignment A_1 in Figure 3.6 was generated by a progressive pairwise alignment method and used as an initial alignment for RNASA. This method aligns the sequences by primary sequence similarity. Therefore when the sequences are distantly related, an alignment by this method does not show the secondary structure similarity of the sequences. A_2 in Figure 3.6 shows the final alignment obtained from RNASA from initial alignment A_1. The primary similarity of the final alignment obtained from RNASA was much worse, but the secondary structure similarity of the final alignment was better than that of the input alignment. The final sum matrix from the output alignment could be used to identify the secondary structures by inspection. This possible secondary structure was compared to the one of `Myx.xanthu`

```
A1
Organism
Dsb.postga   GUCGaacgagaaaGGGAUUGcuugCAAUCCCgaguagagUGGC
Dss.variab   GUCGUacgagaacGCUCUAGcuugCUAGAGCaaguaaaGUGGC
Dsm.acetox   GUCGaacgagaaaGUU---UCUucgGGAAAUgaguagagUGGC
Dmn.tiedje   GUCGUacgagaaaCAUAUCNuucgGGGUAUGgaguaaaGUGGC
Myx.xanthu   GUCGagcgcgaaUAGG------GGcaaCCCUUAguagagCGGC
Cys.fuscus   GUCGagcgcgaauGGA--------gcaaUCCuaguagagCGGC
Con.crocat   GUCGUgcgagaaaGGG------CuucgGCCCcgguaaaGCGGC
Nan.exeden   GUCGaacgGGCUAgca--------aUAGUC-------agUGGC

A2
Organism     [[[[[        [[[[[[[    ]]]]]]]        ]]]]]
Dsb.postga   GUCGaacgagaaaGGGAUUGcuugCAAUCCCgaguagagUGGC
Dss.variab   GUCGUacgagaacGCUCUAGcuugCUAGAGCaaguaaaGUGGC
Dsm.acetox   GUCG-aacgagaaaGUUUCUuc-gGGAAAUgaguagag-UGGC
Dmn.tiedje   GUCGUacgagaaaCAUAUCNuucgGGGUAUGgaguaaaGUGGC
Myx.xanthu   GUCG-agcgcgaa-UAGGGGc-aaCCCUUA-g-uagag-CGGC
Cys.fuscus   GUCG-agcgcgaau-GGA--gcaa--UCC-uaguagag-CGGC
Con.crocat   GUCGUgcgagaaa--GGGC-uucg-GCCC--cgguaaaGCGGC
Nan.exeden   GUCG-a--ac--g-GGCUA-gcaa-UAGUC-a-----g-UGGC
A3
Dsb.postga   GUCGaacgagaaa-G-GGA-UUGcu-ugC-AAUCCCgaguagagUGGC
Dss.variab   GUCGUacgag-aacG-CUC-UAGcu-ugCUAGAGC-aaguaaaGUGGC
Dsm.acetox   GUCGaacgagaaa-GUUUC-UugcG--GA-AAU---GaguagagUGGC
Dmn.tiedje   GUCGUacgagaaaCAUAUCNuucgG--GG-UAU-G-gaguaaaGUGGC
Myx.xanthu   GUCGagcgcg-aa-------UAGGG--Gc-aaCCCUUAguagagCGGC
Cys.fuscus   GUCGagcgcg-aa-------u-GGA--gc-aaUCC-uaguagagCGGC
Con.crocat   GUCGUgcgag-aa-------a-GGGCuuc-gGCCC-cgguaaaGCGGC
Nan.exeden   GUCGaacg-G-GC-------U---A--gc-aaU----AGU-CagUGGC

A4
             [[[[[          [[[[[[[      ]]]]]]]        ]]]]]
Dsb.postga   GUCG-aacg-agaaaGGGAUUGcu--ugCAAUCCCgaguagag-UGGC
Dss.variab   GUCGUa-cga-gaacGCUCUAGcu--ugCUAGAGCaagu-aaaGUGGC
Dsm.acetox   GUCG-aacg-agaaa-GUUUCUuc---gGGAAAUGaguag-ag-UGGC
Dmn.tiedje   GUCGUa-cga-gaaaCAUAUCNuu--cgGGGUAUGga-guaaaGUGGC
Myx.xanthu   GUCG-agcg-cgaa--UAGGGGc---aaCCCUUA-gu--agag-CGGC
Cys.fuscus   GUCG-agcg-cgaau---GGA-gca--a-UCC---uaguagag-CGGC
Con.crocat   GUCGUg-c-gagaaa--GGGC-uuc--g-GCCC--c-gguaaaGCGGC
Nan.exeden   GUCG-aac---g-----GGCUAgc-a-aUAGUC--------ag-UGGC
```

Figure 3.6: Alignment of segments of 16S RNA sequences. The names of the used 16S RNA sequences are from RDP. This region corresponds to nucleotides 61 to 106 in *E.coli* 16S RNA. A_1 is the alignment generated by the progressive pairwise algorithm. A_2 is the final alignment generated by initial alignment A_1 using RNASA with window size=4. A_3 is a longer alignment generated by the progressive pairwise algorithm. A_4 is the final alignment generated by the initial alignment A_3 using RNASA with window size=4. Left ([) and right (]) parts of possible stem regions (A, A', B, B') are symmetric. Upper case characters indicate possible secondary structure regions.

whose secondary structure is known already and matched well.

3.3.3 Initial Alignment

Alignment A_3 in Figure 3.6 was obtained by progressive pairwise alignment. A_3 is longer than A_1 because a lower penalty for nulls was applied in the progressive pairwise alignment. Alignments A_4 is the final alignment using RNASA with input alignment A_3. Alignment A_4 has more nulls than A_2 and these additional nulls give clearer resolution of stem and loop regions. When an initial alignment is not long enough, certain columns in a final alignment may contain bases in loop regions and stem regions. Thus, it is important to ensure that the length of the initial alignment is appropriate, based on the output alignment. Ishikawa *et al.* [38] and Kim *et al.* [45, 44] have shown the advantages of using an initial alignment obtained by heuristics in SA.

3.3.4 Window Sizes

Figure 3.7 shows the alignments obtained with different window sizes.

The alignments B_1, B_2, B_3 and B_4 in Figure 3.7 and A_2 in Figure 3.6 were all obtained from the same initial alignment (A_1 in Figure 3.6) by applying different window sizes. Clearer separation of paired and unpaired regions was obtained at window sizes of 4 to 5. The alignment A_2 is the best match to the known alignment. By increasing the window size, more weight can be given to windows having diagonally consecutive hits. Therefore the effect of noise can be reduced.

```
B1
GUCGaacgagaaaGGGAUUGcuugCAAUCCCgaguagagUGGC
GUCGUacgagaacGCUCUAGcuugCUAGAGCaaguaaaGUGGC
GUCGaacgagaaa-GUUUC-UucgGGAA-AUgaguagagUGGC
GUCGUacgagaaaCAUAUCNuucgGGGUAUGgaguaaaGUGGC
GUCGagcgcgaa-----UAGGG-GcaaCCCUUAguagagCGGC
GUCGagcgcgaa-----uG--GAgcaaUCC-uaguagagCGGC
GUCGUgcgagaaaGGGCuu-c--g--GCC-CcgguaaaGCGGC
GUCGaacg-G-GC----U-A---gc-aaU---AGU-CagUGGC

B2
GUCGaacgagaaaGGGAUUGcuugCAAUCCCgaguagagUGGC
GUCGUacgagaacGCUCUAGcuugCUAGAGCaaguaaaGUGGC
GUCGaacgagaaaGUUUC--Uucg-GGAAAUgaguagagUGGC
GUCGUacgagaaaCAUAUCNuucgGGGUAUGgaguaaaGUGGC
GUCGagcgcga-aUAGGG-Gc--aaCCCUUA--guagagCGGC
GUCGagcgcgaau-GGA--gc---aa-UCCua-guagagCGGC
GUCGUgcgagaaa--GG--GCuucgG-CC-CcgguaaaGCGGC
GUCGaacg-GGC---UA--gc--aa--UA----GUCa-gUGGC
B3
GUCGaacgagaaaGGGAUUGcuugCAAUCCCgaguagagUGGC
GUCGUacgagaacGCUCUAGcuugCUAGAGCaaguaaaGUGGC
GUCGaacgagaaa-GUUUC-UucgGGAAAU-gaguagagUGGC
GUCGUacgagaaaCAUAUCNuucgGGGUAUGgaguaaaGUGGC
GUCGagc-gcgaa-UAGGG-GcaaCCCUUA--gu-agagCGGC
GUCGagc-gcgaau-GGA--gc-aa-UCC-uagu-agagCGGC
GUCGUgcgagaaa--GGGC-uucg-GCCC-c-gguaaaGCGGC
GUCGaa---cg---GGCUA-g-caaUAGUC------a-gUGGC

B4
GUCGaacgagaaaGGGAUUGcuugCAAUCCCgaguagagUGGC
GUCGUacgagaacGCUCUAGcuugCUAGAGCaaguaaaGUGGC
GUCG-aacgagaaaGUUUCU-ucgGGAAAUgaguagag-UGGC
GUCGUacgagaaaCAUAUCNuucgGGGUAUGgaguaaaGUGGC
GUCG-agcgcgaa-UAGGGG-caaCCCUUA--guagag-CGGC
GUCG-agcgcgaa-u-GGA-gcaa-UCCua--guagag-CGGC
GUCGUgcgagaaa--GGGC-uucg-GCCC--cgguaaaGCGGC
GUCG----aacg--GGCUA-gcaa-UAGUC-ag------UGGC
```

Figure 3.7: Alignment of segments of 16S RNA sequences with different window sizes. Each alignment was obtained after 300000 iterations. B_1 is the alignment with window size=1. B_2 is the alignment with window size=2. B_3 is the alignment with window size=3. A_2 in Figure 3.6 is the alignment with window size=4. B_4 is the alignment with window size=5. A_2 in Figure 3.6 gives the best match to a known structure.

3.3.5 Effect of Double Shuffle

To compare the efficiency of the double shuffle and single shuffle, RNASA with double shuffle and RNASA with single shuffle were tested with initial alignment A_1 in Figure 3.6. Figure 3.8 shows the annealing curves with single shuffle and double shuffle.

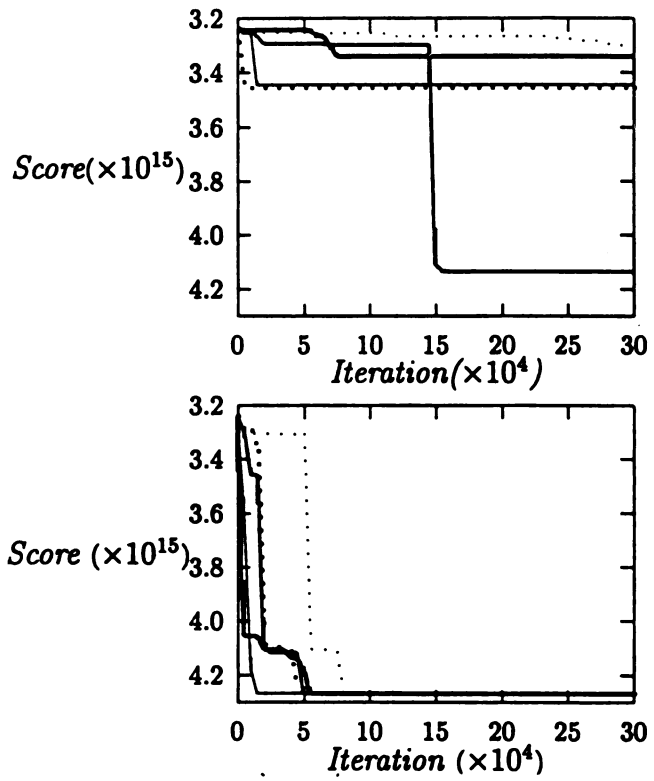


Figure 3.8: (a) Annealing curve with single shuffle. (b) Annealing curve with double shuffle. Initial temperature $T_i = 10^4$ and final temperature $T_f = 1$. Unit of score is 10^{15} and Unit of iteration is 10,000 iterations.

RNASA with single shuffle took longer to converge to the global maxima score and had a tendency to stick to local maximum, even with a very large number of iterations.

3.3.6 Speed of Convergence

The converging time necessary to get satisfactory alignment in RNASA is generally tied to the number of sequences, length of sequences and heuristic algorithm.

- The length of alignment. It is clear that the converging time for longer alignments is longer than that of shorter alignments from equation (9). For example, it took 12 minutes (0.3 million iterations) to get A_4 and 1 minute (10,000 iterations) to get the alignments in Figure 3.7.

  It took less than 10 minutes (.3 million iterations) to align first 10 sequences in Figure 3.4 (a) (length=53). However, it took 2hours (2 million iterations) to align in Figure 3.5 (a) (length=107).

- The quality of initial alignment. The quality of initial alignment affects the converging time. If the initial alignment is closer to the optimal, it will require less converging time. Figure 3.6 shows the importance of a good heuristic algorithm.

- The number of sequences. It is clear that longer SA time is required for a larger number of sequences from equation (9). To align 14 sequences in Figure 3.4 (a), it took approximately 20 minutes (.5 million iterations).

To reduce the computational time, well-conserved regions may be fixed and only the regions between the fixed regions annealed. Also, by eliminating well aligned regions the length of long RNA sequences can be shortened and can then be aligned.

3.4 Conclusion

In this paper, we describe a method to align RNA sequences and identify possible secondary structures using simulated annealing. In our approach, sequences were first aligned based on primary sequence similarity and then realigned based on secondary structure information. Dot matrices from intra-sequence comparison are generated and overlapped to form a sum matrix. We built a clearly defined score function for this sum matrix based on the probability of finding the observed pattern in completely unaligned sequences. We were able to reduce SA time by reducing the search space as well as using the double shuffle move set. We showed that RNASA can generate alignments with clear secondary structure identification.

The main advantage of RNASA is its ability to align conserved secondary structures in distantly related sequences. One potential disadvantage of this method is that is is not based on finding the lowest energy secondary structure (via thermodynamic energy rules). However, RNASA is able to handle pseudoknotted structures and conserved alternative foldings, both of which are not modeled well by most energy based methods.

Chapter 4

Inferring Relatedness of a Macromolecule to a Sequence Database Without Sequencing

Sequence database searches with unidentified sequence data have been a source of useful information of identifying the biological information of a macromolecule isolate. To do the sequence database searches, the whole or part of the primary sequence information is required. However, in some situations, sequencing a macromolecule is not practical. We need more crude but faster methods without sequencing an unidentified macromolecule to identify the relatedness between database sequences and an unidentified macromolecule. To achieve this goal, we study the problem of obtaining biological information about a macromolecule isolate using only the restriction pattern of that isolate obtained from digestion with enzymes and a database D of

sequences. We investigate a three step approach to solving this problem. (1) we obtain a restriction pattern of the isolate while analytically deriving the corresponding restriction maps of the sequences in the database. (2) We identify a set $S \subseteq D$ of sequences which have restriction maps that are most similar to the unknown isolate's restriction pattern. This task is complicated by the fact that we have only approximate fragment lengths for the unknown isolate and that we do not know the actual ordering of the unknown isolate's fragments. Despite these difficulties, we derive experimental results which indicate maximum matching techniques are effective in identifying the correct set most of the time. In this step we introduce Maximum Site Matching Problem (MSMP) and show MSMP is in the class of NP-complete problems which are conjectured to have no polynomial time solution. (3) We use the set S to infer biological information (such as sequence information or hierarchical classification information) about the unknown isolate. We demonstrate experimentally that the closeness of the sequences in the set S to each other can be used to infer the relatedness of the unknown isolate to the sequences of the set S.

4.1 Introduction

Sequence database searches have become a normal first step in the identification and characterization of new macromolecule isolates. The information obtained from such database searches and comparisons often yields novel insights into isolate origin, function, and evolution. A number of very good tools exist for searching sequence databases, for example BLAST [2] and FASTA [65]. These tools can be very powerful in rapidly discovering even weak similarities to a query. However, all of these database search tools require that the molecular sequence of the query be known.

Even though improvements in methods for nucleic acid and protein sequencing have substantially reduced the cost of obtaining sequences, there are many situations in which sequencing costs are still too high or the required specialized equipment is just not available. Often, a researcher is able to obtain more macromolecule isolates than it is practical to sequence. Many of these isolates may be identical, and some rapid method is needed to obtain a set of unique isolates.

One simple method of categorizing multiple isolates is to use the restriction pattern obtained from digestion with certain enzymes (proteases and restriction endonucleases). These enzymes typically cleave the macromolecular substrate at specific subsequences. By comparing the pattern of fragment lengths produced by the isolates, sets of non-identical patterns can be selected for further study.

At a further level of sophistication, the number of matching (same size) fragments between patterns may be used to calculate a measure of (phylogenetic) relatedness

between isolates [60, 76, 48]. If the positions of cleavage sites along the molecule (cleavage site map) are known as well as the fragment sizes, then several more accurate methods exist for estimate the relatedness between isolates [60, 15, 19, 37]. Some work has been done on constructing cleavage site maps from digestion patterns [64, 24, 47, 17, 8, 30, 89], but this work typically assumes that there are overlapping fragments. In our case, there are no overlapping fragments to guide us in constructing a cleavage site map.

Cleavage site maps can be computed easily from molecular sequences. We are interested in how such maps generated from a sequence database might be used to help characterize a molecular isolate. Assume a cleavage pattern for a query molecular isolate (measured experimentally to within some expected limit of accuracy) and a database containing sequences with varying degree of sequence similarity to the query isolate. We would like to be able to extract the set of sequences with cleavage site maps most similar to the (unknown) map of the isolate. A maximum number of common sites between cleavage site maps is used as a measure of similarity. We formalize this problem as Maximum Site Mapping Problem (MSMP) and demonstrate the MSMP to lie in the class of NP-complete problems conjectured to have no polynomial time solution. To overcome this difficulty, we suggest a heuristic approach to solve a confined version of the problem. In addition, we would like some estimate of how closely related these sequences are to our query molecule in terms of sequence similarity or other biological measures. Related work has also been done in identifying the location of restriction maps of short query probes in longer restriction

maps [56, 55]. However, this problem differs from ours in that the ordering of the query fragments is known.

Grouping isolates by cleavage site maps may not be useful in itself, unless this grouping implies some more standard measure of relationship, such as primary sequence similarity. Although primary similarity between two sequences can be used to directly estimate the expected fraction of shared enzyme sites, sequence similarity can not be accurately estimated from the fraction of matching sites alone. Any attempt at performing the reverse calculation requires some knowledge of the underlying distribution of similarity in the pool from which the two sequences were drawn. This leads us to use experimental results for evaluating the effectiveness of different methods of inferring information about a query isolate.

Similar work has been done in classifying unknown peptides by using mass profiles where the unknown peptide is digested by enzymatic or chemical means and the masses of the resulting fragments are determined by mass spectrometry [41, 75, 13]. However, when dealing with peptides, several problem parameters are quite different than when dealing with nucleotides. First, the masses of the resulting fragments can be obtained with essentially no error. Second, the masses reveal significant information about the amino acid composition of each fragment.

We perform our experiments using the Ribosomal Database Project (RDP) database of bacterial 16S rRNA gene sequences [51]. This database is a member of a class of databases containing sequences derived from that of an unknown common evolutionary ancestor. In addition, the phylogenetic relationships of the sequences in

the database have been estimated and are available from the RDP, as is a biological classification scheme based on these relationships. Also, comparison of restriction fragment patterns of rRNA genes (rDNA) is an accepted method of discerning relatedness between isolates in micro-biological studies [57].

In this work, we present a method for finding the set of database sequences with the most sites in common with a query restriction fragment pattern. This method uses sequences in the database as templates to assemble the fragments into putative restriction site maps. The results of this method are shown to be similar to results obtained when the exact fragment order is known for both the query isolate and the database sequences. Also, a method to estimate the sequence similarity between the query and database result set is presented. In addition, we demonstrate that the results can be used to place the query in an existing biological classification scheme.

4.2 Methods

4.2.1 Overview

Our basic problem is to determine biological information such as primary sequence information about an unknown query isolate q using a database D sequences. Furthermore, we do not allow biological sequencing of the unknown query isolate q.

An overview of the approach to solve this problem is given in Figure 4.1.

1. We use enzymes to obtain a ***restriction pattern*** of the query isolate q. Simulta-

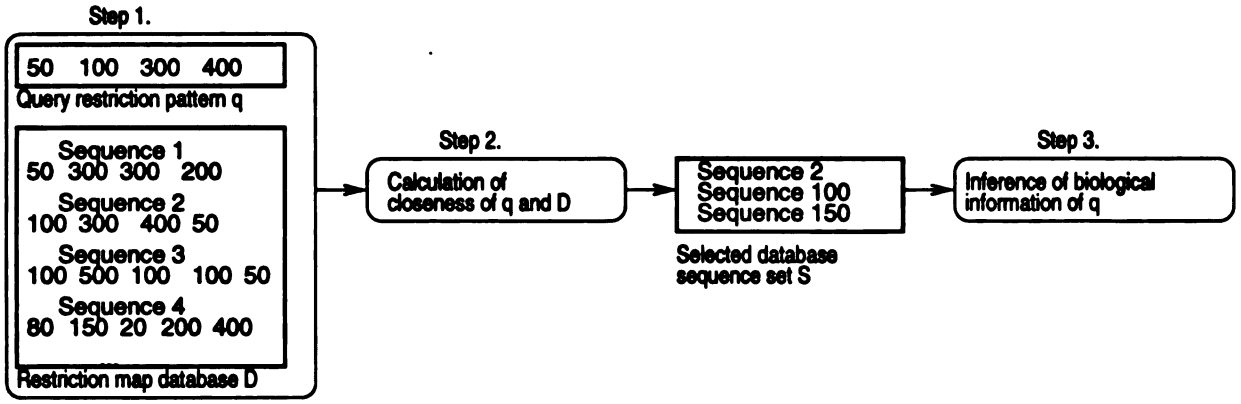


Figure 4.1: The overall process of solving the problem.

neously, we analytically compute *restriction maps* of all the sequences $s \in D$.

2. We then compute the *closeness* of the restriction map of each sequence $s \in D$ to the restriction pattern of q.

3. This closeness information is then used to infer biological information about q.

We describe each of these steps in more detail in the following sections.

We note here that our methods for completing each step, particularly the third step, are dependent on the characteristics of the database D. We first show that analytical methods to infer the primary sequence similarity between the query isolate q and any sequence $s \in D$ require a priori knowledge of the relationship of the underlying probability distribution of the primary sequence similarity of any sequence $s \in D$ to the sequence of isolate q. Straightforward analytic methods are further handicapped by the assumption that for closely related sequences $s, s' \in D$, the relationship of the restriction map of s to the restriction pattern of q is independent of the relationship of the restriction map of s' to the restriction pattern of q. As a result, we experimentally evaluate different methods for inferring information about

the query isolate q. Our experimental results demonstrate that effective inference techniques do exist which utilize the set $S \subseteq D$ of sequences which have restriction maps closest to the restriction pattern of q. This result has good consequences for step two of our approach where we show that computing how close the restriction map of any sequence $s \in D$ to the restriction pattern of q may be computationally difficult but that determining whether the restriction map of a sequence $s \in D$ is extremely close to the restriction pattern of q is computationally tractable.

4.2.2 Obtaining Restriction Patterns and Restriction Maps

The basic biological processing we perform on the isolate q is digestion by enzymes which produces a restriction pattern of q. At the same time, we analytically compute the restriction maps of all sequences $s \in D$ that would have been formed if we had digested these sequences with the same enzymes. To facilitate later comparison of the restriction pattern of q to the restriction maps of $s \in D$, we assume that the database sequences represent molecules with ends at positions homologous to the ends of the query molecule. In practice, the end points of query molecules may be known if, for example, they are produced by the PCR reaction using primers to conserved regions. The primers define the query endpoints. For the specific RDP database, PCR is the method of choice for isolating rRNA genes.

The two processes of obtaining a restriction pattern of q and computing restriction maps for $s \in D$ differ in their precision. Because computing the restriction maps is analytical, we can compute the restriction maps exactly. However, while it

is technically possible to measure the exact length of the restriction fragments of q, the methods required are not suitable for a rapid, inexpensive screen. A more realistic assumption is that fragment sizes would be estimated by simple gel or capillary electrophoresis; this leads us to assume that fragments f which have actual length $|f|$ will be measured to have length between $(1-\epsilon)|f|$ and $(1+\epsilon)|f|$. In our experiments, we estimate ϵ to be 5%.

Definitions.

We now formally define the terms restriction map, restriction pattern, the closeness of two restriction maps, and the closeness of a restriction map to a restriction pattern. We begin by defining restriction maps, restriction patterns, and the equivalence relation that restriction patterns induce on restriction maps.

In the following definitions, s is a nucleotide sequence and z is an enzyme.

Definition 1 The *restriction map* $M_z(s)$ formed by digesting sequence s by enzyme z is the *ordered* multiset of fragment lengths where $l_i > 0$ for $1 \leq i \leq n$. We can obtain the set of locations of cut sites

$$S = \{s : s = \textstyle\sum_{1 \leq j \leq r} l_j; 1 \leq r \leq n\}.$$

Definition 2 Let $M_z(s) = \{l_1, l_2, \ldots, l_n\}$. We define the *restriction pattern* $P_z(s)$ to be the *unordered* multiset of fragment lengths $\{l_1, l_2, \ldots, l_n\}$. We say that $M_z(s)$ *yields* $P_z(s)$.

Note many different restriction maps yield the same restriction pattern. For exam-

ple, if there are n different fragment lengths in $P_z(s)$, $n!$ number of different restriction maps can be generated from $P_z(s)$.

Definition 3 We define two restriction patterns to be *isomorphic* if they yield the same restriction pattern. We denote the set of isomorphic restriction maps which yield the restriction pattern $P_z(s)$ with the notation $[P_z(s)]$.

For example, the restriction map $\{100, 200, 300\}$ and $\{300, 200, 100\}$ are isomorphic but neither is isomorphic to the restriction map $\{100, 200, 100, 200\}$ It is not hard to see that this isomorphic relation defines an equivalence relation on the set of restriction maps.

We now define closeness metrics for restriction maps and restriction patterns. First, we need some notation. Let $sites(M_z(s))$ denote the number of cleavage sites in the restriction map $M_z(s)$. Let $sites([P_z(s)])$ denote the number of cleavage sites in any of the restriction maps in $P_z(s)$. Note $sites(M_z(s))$ and $sites([P_z(s)])$ is one less than the number of fragments in $M_z(s)$ and $P_z(s)$.

Definition 4 We define a *common* site in sequences a and b to be a cleavage site that appears in the same nucleotide position in both a and b. Let $common(M_z(a), M_z(b))$ denote the number of common cleavage sites in the restriction maps $M_z(a)$ and $M_z(b)$. Let $maxcommon(M_z(a), P_z(b))$ denote the maximum number of common cleavage sites in any restriction map in $[P_z(b)]$ and $M_z(a)$.

Definition 5

We define the closeness of the two restriction maps, denoted $closeness(M_z(s), M_z(s'))$, to be $\max(sites(M_z(s)), sites(M_z(s'))) - common(M_z(s), M_z(s'))$. We define the close-

ness of a restriction map to a restriction pattern, denoted $closeness(M_z(s), P_z(s'))$, to be $\max(sites(M_z(s), sites(P_z(s'))) -$

$maxcommon(M_z(s), P_z(s'))$.

Definition 6 We define the set of sequences $C(q, D, j) \subseteq D$ to be the sequences $s \in D$ such that $closeness(M_z(s), P_z(q)) \leq j$.

Example Figure 4.2 explains the above definitions. (a) shows the restriction map $M_z(a) = \{100, 500, 400\}$ for sequence a. (b) shows the restriction pattern $P_z(b) = \{100, 200, 300, 400\}$ for b. $\max(sites(M_z(a)), sites(M_z(b)))$ is 3. (c) shows the $maxcommon(M_z(a), P_z(b)) = 2$. Clearly, the $closeness(M_z(a), P_z(b))$ is 1.

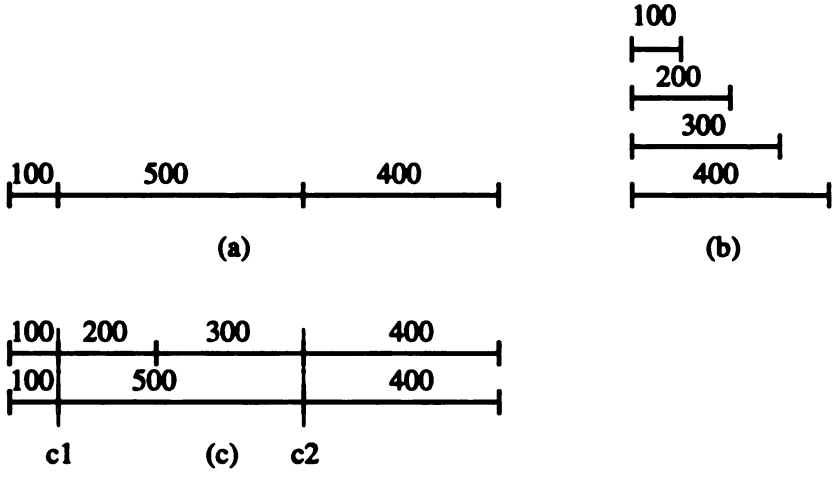


Figure 4.2: (a) Restriction map for a. $(M_z(s) = \{100, 500, 400\}$. (b) Restriction pattern for b. $(P_z(b) = \{100, 200, 300, 400\}$. (c) $maxcommon(M_z(a), P_z(b)) = 2$ with common sites c_1, c_2.

4.2.3 Computing Closeness

Step two of our approach is computing *closeness* $(M_z(s), P_z(q))$ for each sequence $s \in D$. While we show that this problem appears to be a computationally intractable,

we describe how we can efficiently determine if $closeness(M_z(s), P_z(q)) = k$ where k=0,1 or 2. This allows us to compute the set $C(q, D, k)$ which is sufficient for step three of our approach.

We first observe that the problem of computing $closeness(M_z(s_1), M_z(s_2))$ reduces to the problem of computing $common(M_z(s_1), M_z(s_2))$ and the problem of computing $closeness(M_z(s_1), P_z(s_2))$ reduces to that of computing $maxcommon(M_z(s_1), [P_z(s_2)])$. We next note that the second problem, computing $maxcommon(M_z(s_1), [P_z(s_2)])$, is NP-complete via a reduction from the 3-Partition problem. The formal proof is as follows:

Maximum site matching problem: We have as data the restriction pattern from a sequence a and restriction map from a sequence b when a same enzyme is used, say,

Restriction pattern $M(a) = \{a_i : 1 \leq i \leq n\}$ from the digest of sequence a

Restriction map $P(a) = \{b_i : 1 \leq i \leq m\}$ from the nucleotide sequence b

In general $M(a), P(b)$ will be multisets; that is, there may be values of fragment lengths that occur more than once. The lengths of sequences a and b is

$$L = \sum_{1 \leq i \leq n} a_i = \sum_{1 \leq i \leq m} b_i \tag{4.1}$$

Given the above data the problem is to find orderings for the set $M(a)$ such that the number of common sites implied by this ordering is maximum.

Let S_n the set of all permutations of $M(a)$. For $\sigma \in S_n$ calls a configuration. By

ordering $M(a)$ according to σ, we obtain the set of locations of *common* cut sites

$$S = \{s : s = \sum_{1 \le j \le r} a_{\sigma(j)} \text{ and } s = \sum_{1 \le j \le m} b_j; 0 \le r \le n, 0 \le t \le m\}.$$

Since we want to record only the location of cut sites, the set S is not allowed repetitions, that is, S is not a multi-set. Now label the elements of S such that

$$S = \{s_j : 0 \le j \le k\} \tag{4.2}$$

with $s_i \le s_j$ and $0 \le k \le min(m, n)$. The problem is to find a configuration with maximum k common sites.

Theorem: Maximum site matching problem is NP-complete.

Proof: Assume k be the the number of maximum common sites between $M(a)$ and $P(b)$. It is clear that the problem as described above is in NP, as a nondeterministic algorithm need only guess a configuration σ and check in polynomial time if k common sites exist. The number of steps to check this is in fact linear. To show this problem is NP complete we transform the 3-Partition problem to this problem.

In the 3-Partition problem, known to be NP-complete, we are given a finite set $M(a)$ of $3m$ elements, a positive bound C, and a positive integer $s(a)$ for each $a \in M(a)$ such that $C/4 < s(a) < C/2$ and such that $\sum_{a \in M(a)} s(a) = mC$. We wish to determine whether $M(a)$ can be partitioned into m disjoint sets $M_1(a), M_2(a), \ldots, M_m(a)$ such that, for $1 \le i \le m$, $\sum_{a \in M_i(a)} s(a) = C$. If $\sum_{a \in M(a)} s(a) = L$ is not divisible by m, or $|M(a)|$ is not divisible by three, there can be no disjoint sets $M_1(a), M_2(a), \ldots, M_m(a)$; else, consider as input to this problem

the data

$$M(a) = \{s(a_i) : 1 \leq i \leq 3m\}$$

$$P(b) = \{C, C, \cdots, C\}$$

This is simply this problem in the case when the set $M(b)$ has m fragments of equal length and maximum common sites $k = m - 1$.

Heuristic Solution for MSMP.

As a result, we have focused on a restricted version of this problem which can be solved in polynomial time. In particular, we give a simple polynomial time algorithm which solves the decision problem, "Is $closeness(M_z(s), [P_z(q)]) = k$?" where k=0,1, or 2. We can apply this algorithm to all sequences $s \in D$ to efficiently compute the set $C(q, D, k)$.

We first simplify the problem by observing the answer is no unless $sites([P_z(q)]) - 2 \leq sites(M_z(s)) \leq sites([P_z(q)]) + 2$. In other words, we try to find the answer for $closeness(M_z(s), [P_z(q)]) = k$ where $k = 0, 1, 2$. Thus, we need only consider sequences $s \in D$ which satisfy the constraint $sites([P_z(q)]) - 2 \leq sites(M_z(s)) \leq sites([P_z(q)]) + 2$. Next, if s and q have the same number of sites, say n, we observe that for these sequences s, the answer is yes if and only if $P_z(s)$ is identical to $P_z(q)$ assuming that the fragment lengths are exact. This is easily solved in $O(n \log n)$ time by sorting both multi-sets of fragment lengths and verifying that the i^{th} shortest

database fragment length matches the i^{th} shortest query fragment length for $1 \leq i \leq n$. If the answer is yes, which means $closeness(M_z(s), [P_z(q)]) = 0$, we stop. If the answer is no, any two fragments from $P_z(s)$ and $P_z(q)$ are selected and merged into a longer fragment. Let $P_z(s)'$ and $P_z(q)'$ be new restriction patterns. Again we observe the answer is yes if and only if $P_z(s)'$ and $P_z(q)'$. If the answer is yes, which means $closeness(M_z(s), [P_z(q)]) = 1$, we stop. If the answer is no, any two fragments from $P_z(s)'$ and $P_z(q)'$ are selected and merged into a longer fragment. Let $P_z(s)''$ and $P_z(q)''$ be new restriction patterns from $P_z(s)'$ and $P_z(q)'$. We observe the answer is yes if and only if $P_z(s)''$ and $P_z(q)''$. If the answer is yes, which means $closeness(M_z(s), [P_z(q)]) = 2$, we stop. If $sites([P_z(q)])$ and $sites(M_z(s))$ are not same, fragments in a restriction pattern with a larger number of sites are merged to be the same number of fragments for both patterns and the answer is checked. Assume $sites([P_z(q)]) = n$ and $sites(M_z(s)) = m$. The number of ways of merging the fragments into $n-1$ is $\binom{n}{2}$. The number of ways of merging the fragments into $n-2$ is approximately $\binom{n}{2}\binom{n-1}{2}$. However, only consecutive fragments in s can be merged into a single fragment. The number of ways of merging the fragments into $m-1$ is $(m-1)$. The number of ways of merging the fragments into $m-2$ is approximately $(m-1)(m-2)$. Therefore in worst case, we have to examine $closeness(M_z(s), [P_z(q)])$ approximately $1+\binom{n}{2}\times(n-1)+\binom{n}{2}\binom{n-1}{2}\times(n-1)(n-2)$ cases. These merging process and examining $closeness(M_z(s), [P_z(q)]) = k$ where $k = 0, 1, 2$ require polynomial time.

While we observe $closeness(M_z(s), [P_z(q)]) = k$, we assume the fragment length in q are exact. However, while the set of fragment lengths for $P_z(s)$ where $s \in D$ may be computed exactly, the fragment lengths for $P_z(q)$ are only approximate. We only know that the actual length of fragment f ranges between $(1-\epsilon)|f|$ and $(1+\epsilon)|f|$. Thus, we must relax our "yes" condition. In particular, we must consider the length of a query fragment f_q to be identical to the length of any database fragment f_s if $(1-\epsilon)|f_q| \leq f_s \leq (1+\epsilon)|f_q|$.

Ideally, we would like to find the sequences $s \in D$ such that $closeness(M_z(s), M_z(q)) = k$. However, our data is imprecise in two ways which complicates this task. First, we have only $P_z(q)$, not $M_z(q)$. Second, we have only the approximate lengths, not the exact lengths, of each query fragment. The natural question to ask is, how much do these data imprecisions affect the correctness of our algorithm? We perform some experiments with the RDP database to show that these imprecisions do not seriously affect the accuracy of our algorithm. These experiments and results are presented in section 1.5.

4.2.4 Inferring Information from Closeness

We now consider the problem of inferring information about the isolate q from the closeness information we have computed. We first describe analytic methods for inferring information about q given $closeness(M_z(s), P_z(q))$ for any $s \in D$. Unfortunately, these techniques have three drawbacks which limit their applicability to biological databases such as the RDP database. First, the analytical methods require

some a priori knowledge on the relationship between q and D such as the distribution of similarities between elements of D and q. Second, the analytic methods typically assume sequences evolve and change only through mutation or substitution; that is, insertions and deletions of nucleotides are typically not modeled. Third, the analytic methods typically assume that two sequences $s, s' \in D$ vary from $s(q)$ independently, even if we know s and s' are biologically similar. For these reasons, we are forced to evaluate our methods of inference experimentally. We describe a basic heuristic approach and experimentally verify the applicability of this approach.

Theoretical Models.

In this section, we describe two methods for using $closeness(M_z(s), P_z(q))$ to infer primary sequence information about q. In both methods, we assume that we actually have $closeness(M_z(s), M_z(q))$ instead of $closeness(M_z(s), P_z(q))$.

1. **Nei and Li's approach.** The first method is based upon the work of Nei and Li [60]. The basic assumption in this model is that all the sequences in D and the underlying sequence $s(q)$ of isolate q are derived from a common ancestor. Sequences diverge over time via a Poisson process as nucleotides mutate. As time goes on, the number of shared sites between q and s typically decreases as more and more nucleotides mutate. Nei and Li developed methods for using the number of common sites present in s and $s(q)$ to determine the amount of time that has progressed since s and $s(q)$ shared a common ancestor. Using the Poisson process, it is then possible to compute the number of changes that

have occurred in each nucleotide position from the common ancestor to both s and $s(q)$. This can then be used to estimate the primary sequence similarity between s and $s(q)$.

2. **Bayesian analysis.** The second method is based upon Bayesian analysis. Given the primary sequence similarity α between s and $s(q)$ and the assumption that each nucleotide position in s is identical to the same nucleotide position in $s(q)$ with probability α, we can compute the probability distribution on the random variable that represents the number of sites common to both restriction maps. However, in our setting, we actually have the number of common sites shared by the restriction maps of s and $s(q)$, and we desire to compute the primary sequence similarity α between s and $s(q)$. If we are also given the fact that the primary sequence similarity between s and $s(q)$ is drawn from a known probability distribution X, we can compute a probability distribution on α using a Bayesian analysis. Detailed analysis is as follows.

   Notations

   q is the query sequence and d is a database sequence.

   α denotes the sequence similarity between q and d.

   r is the length of the restriction enzyme used.

   n_q denotes the number of restriction sites in q. $n_{q,d}$ denotes the number of shared restriction sites in q and d.

   $p = \alpha^r$, where p is the probability of getting a same shared restriction site.

The random variable $n_{q,d}$ is simply the binomial random variable with parameters (n_q, p). Thus the mean of $n_{q,d}$ is simply pn_q and the variance is $n_q p(1-p)$. We are more interested in the reverse direction which is deriving a value of α given an observed value b for $n_{q,d}$. That is, we are interested in the conditional random variable $(\alpha \mid n_{q,d} = b)$. We can apply Bayesian analysis to get a probability distribution on this conditional random variable. That is, we know that

$$P(n_{q,d} = b \mid \alpha = a) = \binom{n_q}{b} a^{4b}(1-a^4)^{n_q-b}.$$

Thus, the probability that α is in the range a_1 to a_2 is simply

$$\frac{\int_{a_1}^{a_2} \binom{n_q}{b} a^{4b}(1-a^4)^{n_q-b} da}{\int_0^1 \binom{n_q}{b} a^{4b}(1-a^4)^{n_q-b} da}.$$

Unfortunately, both of these methods have several flaws which limit their applicability to biological databases such as RDP. First, the assumptions on evolution are too restrictive. In both cases, only mutations are considered; insertions and deletions are not allowed. Furthermore, both models assume that nucleotides mutate with a given probability distribution which we have access to. This is an extremely strong assumption which does not seem to be justifiable in a general setting. Finally, the Bayesian analysis method does little to account for the fact that biologically similar sequences s and s' are likely to either both share a site with $s(q)$ or both not share a site with $s(q)$. For these reasons, it seems unlikely either method can be used in general database settings reliably. Indeed one fundamental assumption that is shared by

both models is that the number of common sites shared by sequences s and $s(q)$ is a binomial random variable with p based upon the primary sequence similarity between s and $s(q)$. Our experiments with the RDP database in the results section indicate the number of common sites does not seem to follow this binomial distribution.

As a result, we explore other methods for inferring information about $s(q)$ and use experimental means to evaluate these methods.

Closest Sequence Methods.

In this section, we focus on using the set of sequences $C(q, D, k)$ for k=0,1 or 2 to infer information about the query isolate q. The basic premise behind this method is that sequences which have primary sequences that are most identical to $s(q)$ are the ones most likely to be in the set $C(q, D, k)$. The extended premise is that the similarity of $s(q)$ to any sequence in $C(q, D, k)$ is, with high probability, lower bounded by the maximum pairwise dissimilarity between any two sequences in $C(q, D, k)$.

We use this basic premise with two different biological metrics of similarity. The first metric is primary sequence similarity. We say that two sequences s and s' have primary sequence similarity $sim(s, s') = \alpha$ for $0 \leq \alpha \leq 1$ if $100\alpha\%$ of the corresponding nucleotide positions in s and s' are identical. Define $sim(q, C(q, D, k))$ to be the quantity $\min_{s \in C(q,D,k)} sim(q, s)$.

Example Let q be the query and the set $C(q, D, 0) = \{s_1, s_2, s_3\}$. Assume $90\% = sim(q, s_1)$, $80\% = sim(q, s_2)$, $85\% = sim(q, s_3)$. Then the lowest primary similarity between q and $C(q, D, 0)$ is $sim(q, C(q, D, 0)) = 80\%$

The second metric is a biological family similarity that is based on a biological classification hierarchy for the database D. Each sequence in D is classified by an x digit number for $1 \leq x \leq 5$ (note each digit may have a different range of values). In addition to the rRNA sequence database, the RDP also distributes phylogenetic information inferred from these sequences. This includes a five level hierarchical classification scheme consistent with the inferred phylogeny. For example, by this classification scheme, *E.coli* is classified as:

- First level 2 BACTERIA
- Second level 2.13 PURPLE_BACTERIA
- Third level 2.13.3 GAMMA_SUBDIVISION
- Fourth level 2.13.3.15 ENTERICS_AND_RELATIVES
- Fifth level 2.13.3.15.2 ESCHERICHIA_SALMONELLA_ASSEMBLAGE.

We say that two sequences s and s' have level similarity $level(s, s') = i$ if their classifications are identical to the i^{th} digit. Define $level(C(q, D, k)) = \min_{s_i, s_j \in C(q,D,k)} level(s_i, s_j)$. Finally, we define $level(q, C(q, D, k))$ to be the quantity $\min_{s \in C(q,D,k)} level(q, s)$. The extended premise implies that $level(C(q, D, k))$ lower bounds $level(q, C(q, D, k))$ with high probability.

Example Let q be the query and the set $C(q, D, 0) = \{s_1, s_2, s_3\}$. Let q be a level 2.1.10.5 and $level(s_1)$=2.1.7.5, $level(s_2)$=2.1.7.1, $level(s_3)$=2.1.7.6. Then $level(C(q, D, 0))$=2.1.7 and $level(q, C(q, D, 0))$=2.1.

If the extended premise is true and if both $sim(C(q, D, k))$ and $level(C(q, D, k))$ are high, we can, with high confidence, infer fairly precise primary sequence information or classification information about q.

In general, it seems that $sim(q, C(q, D, k))$ and $level(q, C(q, D, k))$ should increase if the number of cleavage sites in $P_z(q)$ increases. That is, it seems unlikely that dissimilar sequences will have restriction maps that are close to that of q. On the other hand, it also seems likely that $sim(q, C(q, D, k))$ and $level(q, C(q, D, k))$ will decrease in size as the number of cleavage sites in q increases. These relationships, however, are difficult to analyze mathematically, and thus it is difficult to generate conditions where our technique will be effective. As a result, we experimentally evaluate this basic procedure. As we see in the next section, our experiments indicate this method can be used to effectively infer useful information about roughly half the sequences in RDP and that the error rate is quite small.

4.3 Results and Discussion

4.3.1 Experimental Procedure

The sequence data used in this study was obtained from the Ribosomal Database Project (RDP) (release number 5 of May 17, 1995 [51]). The RDP provides curated databases of ribosomal RNA related information and analysis services. The database used in this study was a subset of the bacterial 16S rRNA database distributed

by the RDP. This database contains 16S rRNA sequence information from about 3000 different bacterial isolates and environmental samples. These sequences are distributed by the RDP as pre-aligned sequences with alignment gaps inserted so that homologous residues appear at the same position in all sequences. This alignment was produced by the RDP curators using, in addition to primary sequence similarity, secondary structure and other higher-order information. Inspection of the alignment indicates that highly diverged regions are often aligned to conserve putative secondary structures without regard to primary sequence. Most of the sequences in this database are incomplete, usually at the two ends. To construct the subset used here, incomplete sequences were removed after first selecting the region corresponding to positions 46 through 1406 of E. coli [9]. The resulting database contained 1575 sequences. Sequence similarity values were calculated for all pairwise combinations of the 1575 sequences. The similarity values clustered around the mean value of 72% identity, with a tail stretching toward higher similarity values (Figure 4.3).

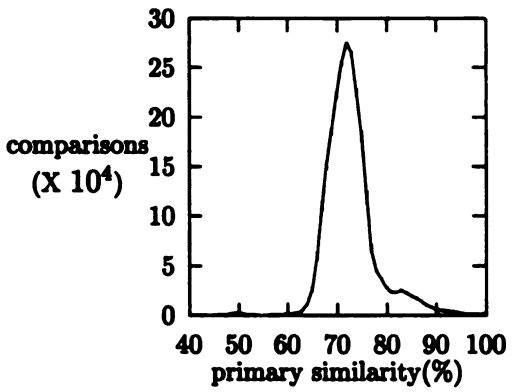


Figure 4.3: Pairwise sequence similarity. The pairwise sequence similarity was determined for all pairwise combinations of 1575 sequences in the test database. Only sequence regions considered in further analysis were included. The similarity of the aligned pairs was determined using the pre-aligned sequences supplied by RDP.

site	seq.no	site	seq.no	site	seq.no	site	seq.no
1	6	11	695	21	10	31	
2	19	12	727	22	5	32	1
3	86	13	587	23	1	33	
4	236	14	429	24	1	34	
5	403	15	331	25	1	35	
6	626	16	188	26	1	36	
7	855	17	80	27		37	
8	893	18	49	28		38	
9	835	19	22	29		39	
10	779	20	9	30		40	

Table 4.1: Sequences by number of sites

Five separate restriction map data sets were computer generated from the 1575 sequences. For each data set, positions matching recognition sites from two commercial restriction enzymes were identified and the length between sites calculated (after removing alignment gaps). Ambiguity codons in the database sequences were treated as not matching any recognition site. The recognition sites (enzymes) chosen to generate the five sets were: AGCT + CATG; CCGG + AATT; CTAG + GATC; GTAC + TCGA; and TTAA + ATAT. The average number of sites in the combined 7875 digests was 9.74 with a median of 9 (Table 4.1).

The maps in a data set were chosen one at a time as queries and are considered as restriction patterns. When a query had a result set $C(q, D, 0)$, this query did not used for the calculation of $C(q, D, 1)$. When a query did not have a result set $C(q, D, 0)$, this query was used as a query for the calculation of $C(q, D, 1)$. This process continued until $k = 2$ for single enzyme, $k = 4$ for multiple enzymes. To simulate experimental data, the fragments sizes of the query were assumed to be accurate to only +/- 5%.

The result set of database sequences where all sites could be matched was determined for each query, assuming the fragment order for the query was known (ordered

query). These result sets were compared with the expected results calculated using the exact site positions from the aligned sequences. These ordered query result sets were missing one or more sequences found using pre-aligned sequences in 836 of the 7875 trials (10.6%). In addition to base changes (point mutations), rRNA genes have accumulated insertions and deletions over the course of evolution. These insertions and deletions may have caused homologous fragments to no longer have sizes within the 5% error bound. The result of these size changes is apparently to cause pattern mismatches over shorter evolutionary distances than would be predicted from site conservation alone.

Another 580 of the 7875 trials (7.4%) produced result sets with extra sequences. Some of these extra matches may be due to peculiarities in the RDP alignment causing occasional site mismatch when comparing aligned sequences; however some extra matches are probably due to matching of non-homologous sites within the 5% error bound for the ordered query tests. Even if all of the additional matches are due to incorrectly pairing non-homologous sites, the percentage of query results affected is still relatively small.

The trials were repeated without assuming the order of query fragments was known (unordered query). Any differences in result sets between ordered and unordered queries represent incorrect pairing of non-homologous regions between query and database sequences (incorrect ordering). Only 263 of the 7875 trials (3.3%) produced result sets with extra sequences not in the ordered query result sets. However, if mismatches were allowed in the site matching, the results deteriorated. When up

to one query and/or database site mismatch was allowed, 36.3% of the unordered result sets contained sequences not in the ordered query result sets. When up to two query and/or database site mismatches were allowed the percentage of result sets with incorrect matches increased to 88.4%.

4.3.2 $sim(q, C(q, D, k))$

Number of mismatches (k). The lowest primary similarities between query and $sim(q, C(q, D, k))$ with k=0,1,2, were calculated as in Figure 4.4. The error bounds were fixed with 5% for k=0,1,2 in the experiments. The values of $sim(q, C(q, D, k))$ were dependent, as expected, on the number of sites in the query and the number of mismatched sites, k, between query and result sets. The values increase when the number of matched cleavage sites increases. The value of $sim(q, C(q, D, k))$ with given cleavage sites decrease when k increases. For example, the values of $sim(q, C(q, D, 0))$ are higher than 90% when the number of cleavage sites is larger than 6. The numbers of required cleavage sites for the values of $sim(q, C(q, D, 0))$ which is higher than 90% are 12 for k=1 and no site for k=2.

Error bounds. $sim(q, C(q, D, k))$ for different error bounds (10%, 15%, 20%) was calculated as in Figure 4.5. In the experiments, the value of k was set to 0. The values of $sim(q, C(q, D, k))$ were dependent, as expected, on the error bounds of the query fragments. The values increase when the sizes of the error bound decrease. For example, the values of $sim(q, C(q, D, 0))$ for 5% error bound are higher than 90% when the number of cleavage sites is larger than 6. The numbers of required cleavage

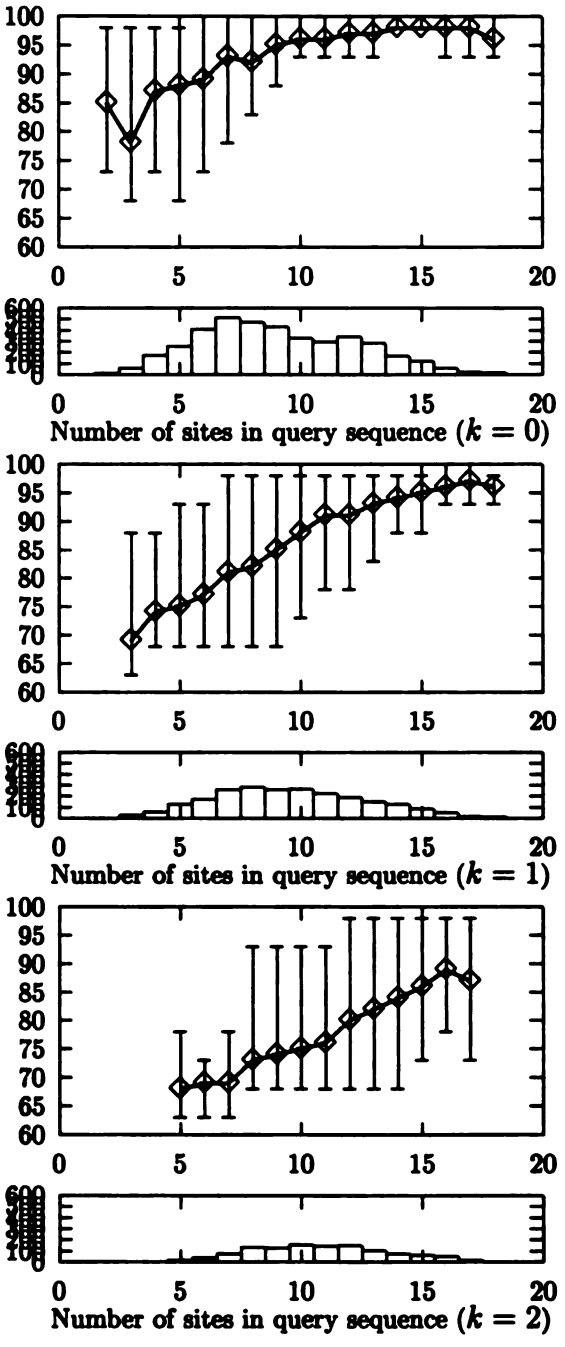


Figure 4.4: $sim(q, C(q, D, k))$ for k=0,1,2 with 5% error bound, These results are shown by number of query sites as mean and range, after discarding the 5% highest and lowest values. Bar chart at bottom indicates the number of queries with result sets with size greater than one. The y axis of the first diagram shows the primary similarity between query and result set. The y axis shows the number of query sequences. A site with the number of query sequences ≤ 10 is eliminated.

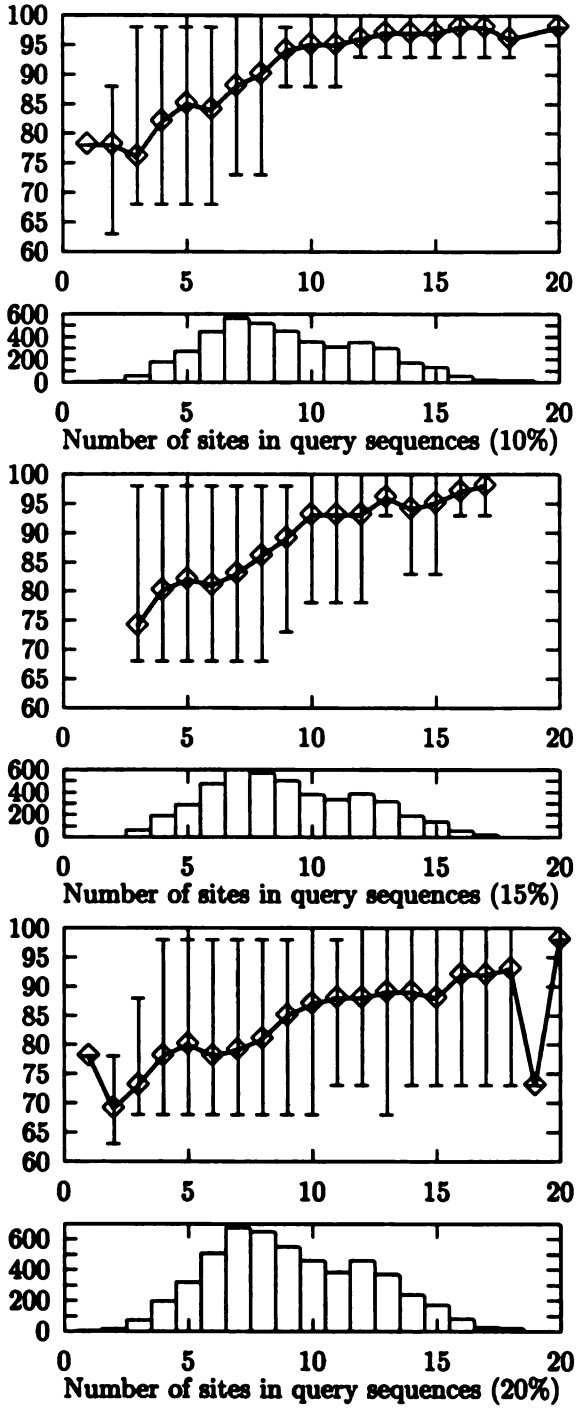
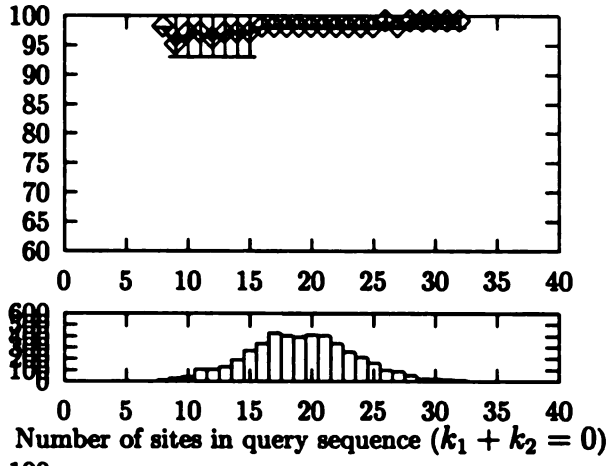


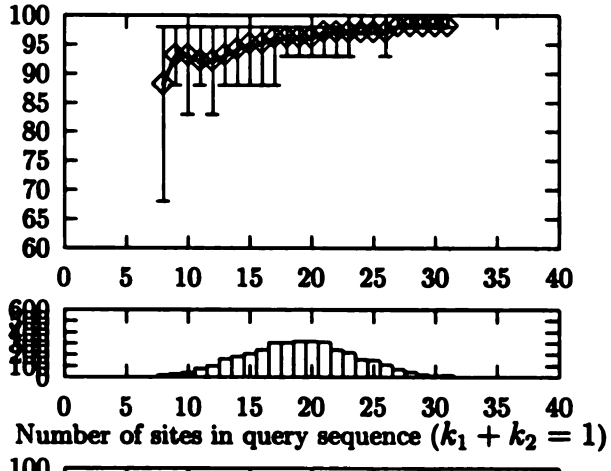
Figure 4.5: $sim(q, C(q, D, 0))$ with 10%, 15%, 20% error bounds. The y axis of first diagram shows the primary similarity between query and result set. The y axis shows the number of query sequences. These results are shown by number of query sites as mean and range, after discarding the 5% highest and lowest values. Bar chart at the bottom of the figure indicates the number of queries with result sets with size greater than one. A site with the number of query sequences ≤ 10 is eliminated.

sites for the values of $sim(q, C(q, D, 0)) \geq 90\%$ are 8 (for 10%), 10 (for 15%), 16 (for 20%).

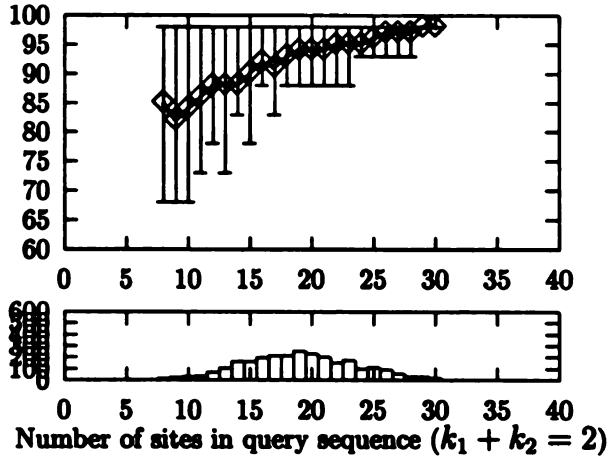
Multiple enzymes. To increase the values of $sim(q, C(q, D, k))$, multiple enzymes were applied to each query. The result set for each enzyme is obtained and $C(q, D, k)$s for each result sets were calculated. Let $C(q, D, k)_A$ and $C_2(q, D, k)_B$ be the results using two different enzymes A and B with same q and D. First, we calculate the common set of $C(q, D, k_1)_A$ and $C_2(q, D, k_2)_B$, which is $C(q, D, k_1 + k_2)_{AB}$. The values of $k_1 + k_2$ are 0 thru 4. Then we examine the value of $C(q, D, k_1 + k_2)_{AB}$. Figure 4.6 shows the results.



Number of sites in query sequence ($k_1 + k_2 = 0$)



Number of sites in query sequence ($k_1 + k_2 = 1$)



Number of sites in query sequence ($k_1 + k_2 = 2$)

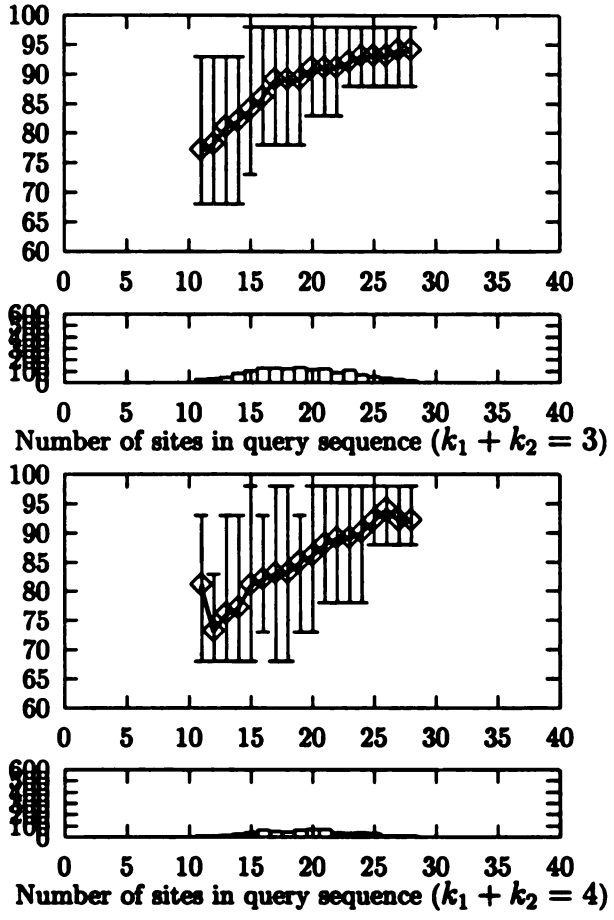


Figure 4.6: $sim(q, (C(q, D, k_1 + k_2))_{AB}$ for $k_1 + k_2$=0 thru 4 with 5% error bound and two enzymes. These results are shown by number of query sites as mean and range after discarding the 5% highest and lowest values. Bar chart at bottom indicates the number of queries with result sets with size greater than one. A site with the number of query sequences ≤ 10 is eliminated.

All the values of $sim(q, (C(q, D, k_1 + k_2))$ are higher than $sim(q, (C(q, D, k))$, as expected, when the values of $k_1 + k_2$ and k are same.

4.3.3 $level(q, C(q, D, k))$

In addition to aligned rRNA sequence databases, the RDP also distributes phylogenetic information inferred from these sequences. This includes a hierarchical classification scheme consistent with the inferred phylogeny. At the most basic level, all

k=0					
x	1	2	3	4	5
t_x	146	47	360	351	3065
r_x	146	46	354	347	2898
hit ratio	100.0	97.9	98.3	98.9	94.6
k=1					
x	1	2	3	4	5
t_x	446	128	373	148	1239
r_x	446	125	362	133	1036
hit ratio	100.0	97.7	97.1	89.9	83.6
k=2					
x	1	2	3	4	5
t_x	552	97	140	39	340
r_x	552	88	127	35	221
hit ratio	100.0	90.7	90.1	89.7	65.0

Table 4.2: Hit ratios for each k. 5% error bound.

sequences tested here are members of category 2, the Bacteria. There are 15 categories at the next level (2.1 through 2.15), 24 at the third level, 94 at the fourth, and 99 at the fifth and highest level. Not all sequences are categorized to all five levels, we assumed an implied category of 0 for all undefined levels.

Number of mismatches (k). The minimum common digits in the levels of q, $level(q, C(q, D, k))$ with k=0,1 or 2, were calculated as in Table 4.2. The error bounds were fixed with 5% for k=0,1,2 in the experiments. We define hit ratio to use as a measure of accuracy of this method. Let t_x be the number of queries with $level(C(q, D, k)) = x$ and r_x be the number of queries with $level(q, C(q, D, k)) = x$ Then hit ratio for xth level is

$$h_x = (r_x/t_x) \times 100(\%) \tag{4.3}$$

The hit ratio decreases when k increases. For example, the value of h_5 are 94.6% for k=0, 83.6% for k=1, 65% for k=2.

error bound=10%					
x	1	2	3	4	5
t_x	453	128	342	336	2924
r_x	453	120	332	328	2731
hit ratio	100.0	97.9	98.3	98.9	94.6
error bound=15%					
x	1	2	3	4	5
t_x	776	229	451	305	2809
r_x	776	213	436	291	2418
hit ratio	100.0	93.0	96.7	95.4	86.0
error bound=20%					
x	1	2	3	4	5
t_x	1419	399	539	278	2565
r_x	1419	353	507	257	1919
hit ratio	100.0	88.5	94.1	92.4	74.8

Table 4.3: Hit ratios for different error bounds.

Error bounds. Hit ratios for different error bounds (10%, 15%, 20%) were calculated as in Table 4.3. In the experiments, the value of k was set to 0. It is clear that the hit ratio decreases when the error bound increases. For example, the value of h_5 are 94.6% for 5%, 94.6% for 10%, 86% for 15%, 74.8% for 20%.

Multiple enzymes. To increase the values of $level(q, C(q, D, k))$, multiple enzymes were applied to each query. The result set for each enzyme is obtained and $C(q, D, k)$s for each result sets were calculated. Let $C(q, D, k_1)_A$ and $C(q, D, k_2)_B$ be the results using two different enzymes with same q and D. First, we calculate the commonly matched sequences $C(q, D, k_1 + k_2)_{AB}$. Then we examine the value of $level(q, (C(q, D, k_1 + k_2)_{AB})$. Table 4.4 shows the results.

The hit ratio of multiple enzymes is much higher than single enzyme as expected. For example, the values of h_5 for 0 thru 4 are 99.0%, 96.2%, 89.1%, 75.9% and 58.4%.

$k_1 + k_2 = 0$					
x	1	2	3	4	5
t_x			4	53	4495
r_x			4	53	4450
hit ratio			100	100	99.0
$k_1 + k_2 = 1$					
x	1	2	3	4	5
t_x	4	3	132	180	3354
r_x	4	3	132	179	3222
hit ratio	100.0	100	99.6	99.0	96.1
$k_1 + k_2 = 2$					
x	1	2	3	4	5
t_x	24	15	259	200	2178
r_x	242	15	258	198	1940
hit ratio	100	100	99.6	99.0	89.1
$k_1 + k_2 = 3$					
x	1	2	3	4	5
t_x	68	41	183	86	1093
r_x	68	41	171	80	830
hit ratio	100	100	93.4	93.0	75.9
$k_1 + k_2 = 4$					
x	1	2	3	4	5
t_x	45	32	72	23	519
r_x	45	26	61	15	303
hit ratio	100	81.2	84.7	65.7	58.4

Table 4.4: Hit ratios for each $k_1 + k_2$. 5% error bound.

4.3.4 Ordered. vs. Unordered

When we compute $sim(q, C(q, D, k))$ for $k > 0$ we allow the merges of fragments in q and/or $s \in D$ to reduce the number of fragments. We know the *order* of the fragments in s while we don't know the order of the fragments in q (*unordered*). Only consecutive fragments in s are allowed to be merged while any combination of fragments in q are allowed to be merged. This additional information (*order* of fragments) gives us more accurate results. We compare $sim(q, C(q, D, 2))$ with two different cases. Let m be the number of fragments in q and n be the number of fragments in s. $sim(q, C(q, D, 2))$ can be calculated when $m = n + 2$ or $n = m + 2$. Let $sim(q, C(q, D, 2)_q)$ be the result by merging the fragments in q and $sim(q, C(q, D, 2)_s)$ be the result by merging the fragments in s. Figure 4.6 shows the difference between $sim(q, C(q, D, 2)_q)$ and $sim(q, C(q, D, 2)_s)$. $level(q, C(q, D, 2)_q)$ and $level(q, C(q, D, 2)_s)$ are calculated by

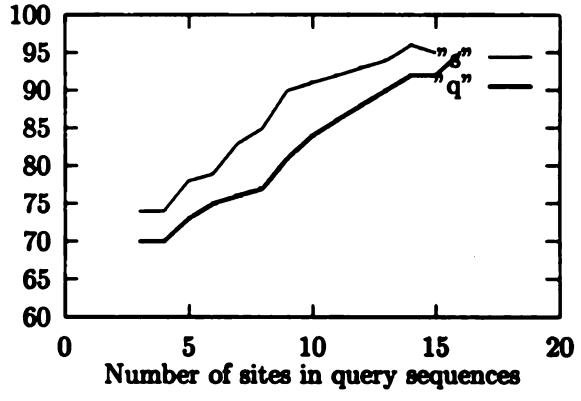


Figure 4.7: $sim(q, C(q, D, 2))_q$ and $sim(q, C(q, D, 2))_s$ with 5% error bound. The y axis of the diagram shows the primary similarity between query and result set. The curve "s" is generated by merging s and the curve "q" is generated by merging q.

doing same process. Hit ratio for each level is calculated. Table 4.5 show the hit ratios of $level(q, C(q, D, 2)_q)$ and $level(q, C(q, D, 2)_s)$.

	$level(C(q,D,2)_q)$				
	1	2	3	4	5
hit ratio	100.0	91.2	94.1	84.4	48.6
	$level(C(q,D,2)_s)$				
	1	2	3	4	5
hit ratio	100.0	92.4	97.4	93.2	62.9

Table 4.5: Hit ratio of $level(q, C(q,D,2)_s)$ and $level(C(q,D,2)_q)$

4.3.5 Simulation of Random Database

To confirm our method can be applied to other databases, we generated random databases and applied our method and checked the results. To simulate the evolutionary procedure, one original sequence, s_0^1 is generated randomly. s_y^x denotes xth random sequence of yth level. Two random sequences, s_1^1, s_1^2 are generated from s_0 with α similar to s_0. Again, four random sequences, $s_2^1, ..., s_2^4$ are generated from s_1^1, s_1^2 with u percent similar to origin sequence. We call u as uniform edge length. This process continues until 2048 ($s_{11}^1, ..., s_{11}^{2048}$) random sequences are generated. The final 2048 random sequences are considered as database sequences. The whole random sequences resemble binary tree structure. Ten random databases with uniform edge length u are generated with different orders of random seeds and different values of u. get random database with different uniform primary similarity.

Same five set of enzymes as in experiment of RDP are applied to ten random databases for each u. All 2048 random sequences in a DB are considered as query sequences. Therefore total number of query sequences are 10240 (2048 × 5). For each query, $(CLOSE(q,D,0)$ is calculated. To identify the properties of selected database sequences, the value of $sim(q,s)$ are accumulated instead of calculation of $sim(q,(CLOSE(q,D,0))$. In this experiment error bound is zero percent. Figure

4.7 shows the output of this simulation with edge length 92% thru 98%. The edge lengths $u \leq 98\%$ are not enough to generate meaning output. Also $u \leq 90\%$ are not biologically meaningful. Therefore we only consider the random databases with $u \leq 99\%$. Figure 4.8 shows the output $u \leq 99\%$.

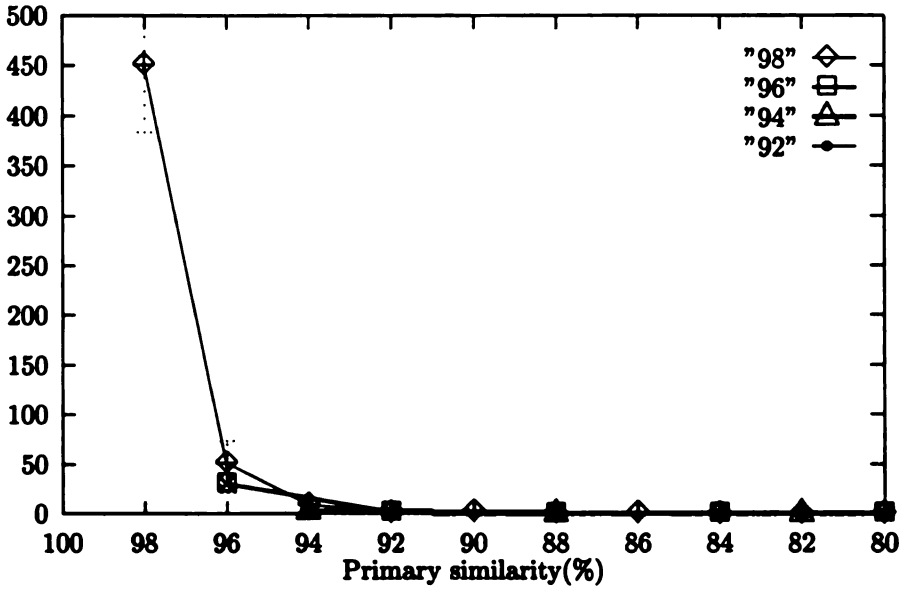


Figure 4.8: Accumulation of $sim(q, s)$ for random databases with edge lengths 98%, 96%, 94%, 92%. Bar charts for each diagram show the standard deviations. y shows the total number of queries with output. x shows the primary similarity between query and matched database sequences.

First diagram in Figure 4.8 shows the total number of query sequences vs. $sim(q, d)$. The number of matched database sequences rapidly decreases when the primary similarity between query and database sequences decreases. The value of $sim(q, s)$ for most of s is greater than 90% which means the value of $sim(q, (CLOSE(q, D, 0))$ is also greater than 90%. This diagram shows that our method can be applied to other sequence databases. This diagram shows the usefulness of our method.

The output in the first digram of Figure 4.8 is divided by two groups which are *forword* and *backward* mutations. First, we check the first common ancestor, o, of the q and d. Second, we check if q and o are matched. If q and d are matched, we consider the output as forward. If not, we consider the output as backward. Second diagram in Figure 4.8 shows the output with forward and third diagram in Figure 4.8 shows the output with backward. The curves in second diagram are decayed rapidly while the curves in third diagram are increased until some period and decayed. Further study is required for more detailed analysis for these diagrams.

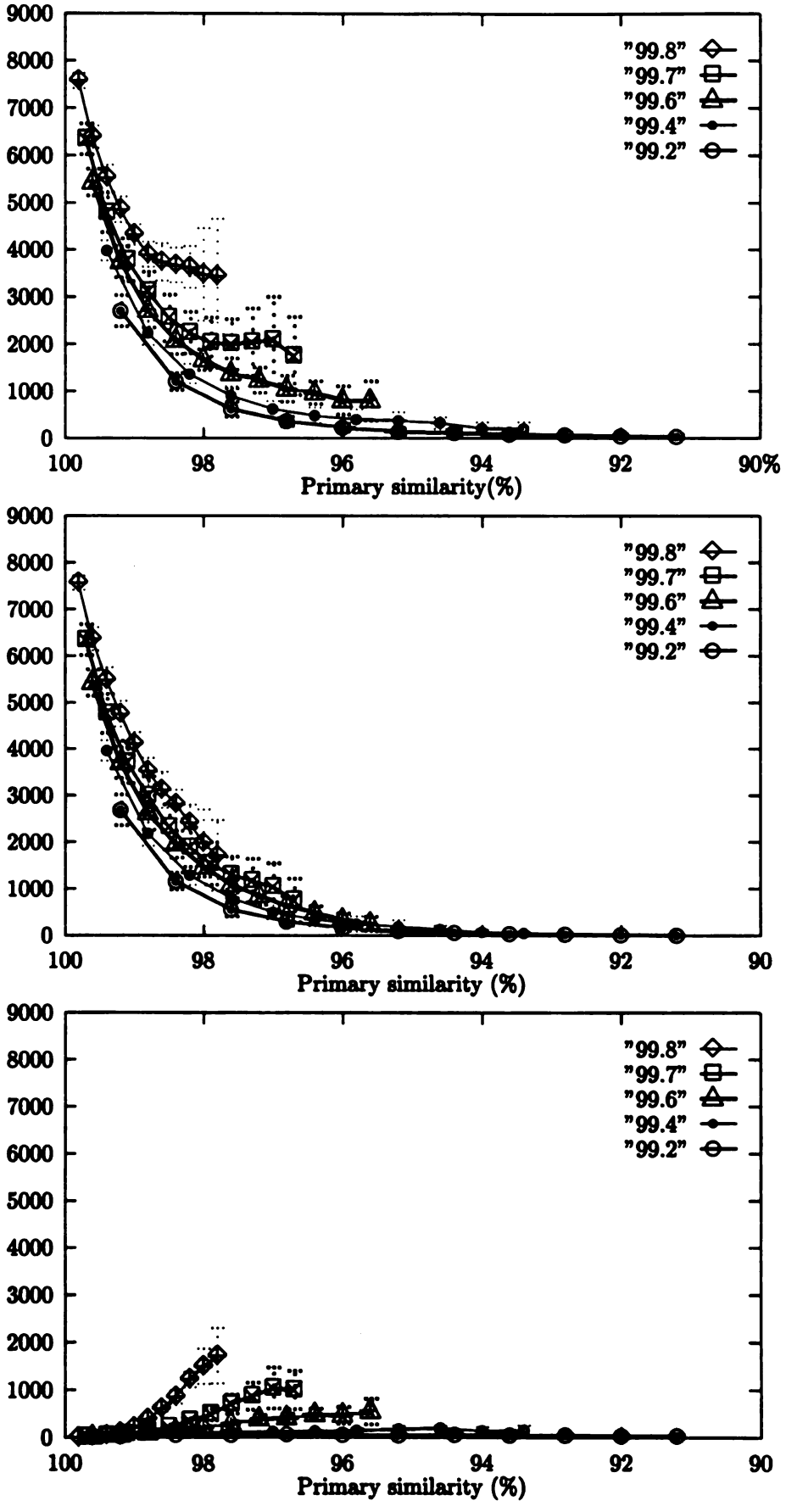


Figure 4.9: Accumulation of $sim(q, s)$ for random databases with uniform primary similarities 99.8%, 99.7%, 99.6%, 99.4%, 99.2%. Bar charts for each diagram show the standard deviations.

Chapter 5

Conclusions

In this thesis, we proposed an algorithm called MSASA, which is based on simulated annealing approach, to align multiple protein sequences. Dynamic programming of multiple sequence alignment has been widely used to find an optimal alignment for certain cost functions. However, it does not allow certain types of cost function including natural gap costs, and limits the number of sequences that can be aligned due to its high computational complexity. MSASA overcomes these problems because it uses any gap costs which generates a better solution. It aligns more sequences, and takes less computation time compared to a dynamic programming approach. A solution set for a multiple sequence alignment problem is identified, and the multiple sequence alignment problem is reformulated to find an optimal alignment from this solution set. The computational complexity of MSASA is significantly reduced by substituting the higher temperature phase of the annealing process by a fast heuristic algorithm and confining the solution set.

We then suggested an algorithm called RNASA, which is based on simulated annealing, to align multiple RNA sequences. The output RNA alignment from RNASA is used to identify possible secondary structures. A transition rule, based on double shuffle, is proposed. This transition rule provides faster convergence to an optimal solution. We show that this new transition rule takes less convergence time than a conventional transition rule, based on single shuffle. RNASA is applied to the sequences from RDP and the results are compared to known secondary structures. The usefulness of RNASA is supported with experimental results.

We then studied the problem of obtaining biological information about a macromolecule isolate using only restriction patterns and restriction map databases. Maximum site matching problem (MSMP) is defined in a formal way and proved to be in NP-complete problems. We suggested a heuristic algorithm to solve MSMP. A three phase approach to obtain relatedness of unknown macromolecules and database sequences is suggested. We demonstrate usefulness of our approach using RDP database.

Bibliography

[1] S. F. Altschul. Gap costs for multiple sequence alignment. *J. Theor. Biol.*, 138:297–309, 1989.

[2] S. F. Altschul, W. Gish, W. Miller, E. M. Myers, and D. J. Lipman. Basic local alignment search tool. *J. Mol. Biol.*, 215:403–410, 1990.

[3] S. F. Altschul and D. J. Lipman. Trees,stars,and multiple biological sequence alignment. *SIAM J. appl. Math.*, 49:197–209, 1989.

[4] P. Argos. A sensitive procedure to compare amino acid sequences. *J. Molec. Biol.*, 193:385–396, 1987.

[5] D. J. Bacon and W. F. Anderson. Multiple sequence alignment. *J. Molec. Biol.*, 191:153–161, 1986.

[6] W. Bains. Multan: A program to align multiple dna sequences. *Nucl. Acids Res.*, 14:159–177, 1986.

[7] G. J. Barton and M. J. E. Sternberg. A startegy for the rapid multiple alignment of protein sequences: confidence levels from tertiary structure comparisons. *J. Molec. Biol.*, 198:327–337, 1987.

[8] B. Bellon. Construction of restriction maps. *CABIOS*, 4:111–115, 1988.

[9] J. Brosius, M. L. Palmer, P. J. Kennedy, and H. F. Noller. Complete nucleotide sequence of a 16s ribosomal rna gene. *Proc. Natl. Acad. Sci. U.S.A*, 75:4801–4805, 1978.

[10] H. Carrillo and D. Lipman. The multiple sequence alignment problem in biology. *SIAM J. Appl. Math*, 48:1073–1082, 1988.

[11] T. R. Cech. Conserved sequences and structures of group i introns:building an active site for rna catalysis-a review. *Gene*, 73:259–271, 1988.

[12] F. Corpet. Multiple sequence alignment with hierarchical clustering. *Nucl. Acids Res.*, 16:10881–10890, 1988.

[13] J. Cottrell. Protein identification by peptide mass fingerprinting. *Peptide Research*, 7(3):115–124, 1994.

[14] M. O. Dayhoff. A model of evolutionary change in proteins. matrices for detecting distance relationships. In *Atlas of Protein Sequence and Structure*, volume 5 suppl. 3, pages 345–352. Dayhoff. M. O.(ed) Washington. DC: National Biomedical Research Foundation, 1978.

[15] R. DeBry and N. A. Slade. Cladistic analysis of restriction endonuclease cleavage maps within a maximum-likelihood framework. *Syst. Zool.*, 34(1):21–34, 1985.

[16] A. Delcoigne and P. Hansen. Sequence comparison by dynamic programming. *Biometrika*, 62:662–664, 1975.

[17] R. Durand and F. Bregegere. An efficient program to construct restriction maps from experimental data with realistic error levels. *Nucleic Acids Res.*, 12:703–716, 1984.

[18] S. R Eddy and R. Durbin. Rna sequence analysis using covariance models. *Nucl. Acids Res.*, 22:2079–2088, 1994.

[19] Joseph Felsenstein. Phylogenies from restriction sites: a maximum-likelihood approach. *Evolution*, 46(1):159–173, 1992.

[20] D. F. Feng and R. F. Doolittle. Progressive sequence alignment as a prerequisite to correct phylogenetic trees. *J. Molec. Evol.*, 25:351–360, 1987.

[21] D. F. Feng, M. S. Johnson, and R. F. Doolittle. Aligning amino acid sequences: comparison of commonly used methods. *J. Molec. Evol.*, 21:112–125, 1982.

[22] J. W. Fickett. Fast optimal alignment. *Nucl. Acids Res.*, 12:175–180, 1984.

[23] W. M. Fitch and T. Smith. Optimal sequence alignments. *Proc. Natl. Acad. Sci. USA.*, 80:1382–1386, 1983.

[24] W. M. Fitch, T. F. Smith, and W. W. Ralph. Mapping the order of dna restriction fragments. *Gene*, 22:19–29, 1983.

[25] G. E. Fox and C. R. Woese. 5s rna secondary structure. *Nature*, 256:505–507, 1975.

[26] M. L. Fredman. Algorithms for computing evolutionary similarity measures with length independent gap penalties. *Bull. Math. Biol.*, 46:553–566, 1984.

[27] O. Gotoh. An improved algorithm for matching biological sequences. *J. Molec. Biol.*, 162:705–708, 1982.

[28] O. Gotoh. Alignment of three biological sequences with an efficient traceback procedure. *J. Theor. Biol.*, 121:327–333, 1986.

[29] M. Gribskov, R. Luthy, and D. Eisenberg. Profile analysis. *J. Theor. Biol.*, 183:146–159, 1990.

[30] A. V. Grigorjev and A. A. Mironov. Mapping dna by stochastic relaxation: a new approach to fragment sizes. *CABIOS*, 6:107–111, 1990.

[31] L. K. Grover. Standard cell placement using simulated sintering. *Proc. 24th DAC*, pages 56–59, 1987.

[32] R. R Gutell. Collection of small subunit (16s and 16s like) ribosomal structures. *Nucl. Acids Res.*, pages 3502–3507, 1994.

[33] J. Hein. A new method that simultaneously aligns and reconstructs ancestral sequences for any number of homologous esequences, when the phylogeny is given. *Molec. Biol. Evol.*, 6:649–668, 1989.

[34] D. G. Higgins and P. M. Sharp. Clustal: a package for performing multiple sequence alignment on a microcomputer. *Gene*, 7:237–244, 1988.

[35] M. Hirosawa, M. Hoshida, M. Ishikawa, and T. Toya. Mascot: Multiple alignment system for protein sequences based on three-way dynamic programming. *CABIOS*, 9:161–167, 1993.

[36] P. Hogeweg and B. Hesper. The alignment of sets of sequences and the construction of phyletic trees: an integrated method. *J. Molec. Evol.*, 20:175–186, 1984.

[37] K. E. Holsinger and R. E. Jansen. Phylogenetic analysis of restriction site data. *Methods in Enzymology*, 224:439–455, 1993.

[38] M. Ishikawa, T. Toya, M. Hoshida, K. Nitta, A. Ogiwara, and M. Kanehisa. Multiple sequence alignment by parallel simulated annealing. *CABIOS*, 9:267–273, 1993.

[39] A. B. Jacobson, L. Good, J. Simonetti, and M. Zuker. Some simple computational methods to improve the folding of large rnas. *Nucleic Acids Res.*, 12:45–52, 1984.

[40] B. D. James, G. J. Olsen, and N. R. Pace. Phylogenetic comparative analysis of rna secondary structure. *Methods in Enzymology*, 180:227–230, 1989.

[41] P. James, M. Quadroni, E. Carafoli, and G. Gonnet. Protein identification by mass profile fingerprinting. *Biochemical and Physical Research Communications*, 195(1):58–64, 1993.

[42] M.S. Johnson and R. F. Doolittle. A method for the simulataneous alignment of three or more amino acid sequences. *J. Molec. Evol.*, 23:267–287, 1986.

[43] R. A. Jue, N. W. Woodbury, and R. F. Doolittle. Sequence homologies among e.coli ribosomal proteins: evidence for evolutionary related groupings and internal duplication. *J. Molec. Biol.*, 15:129–148, 1980.

[44] J. Kim and S. Pramanik. Multiple sequence alignment using simulated annealing. In *Second International Conference on Intelligent Systems for Molecular Biology*, 1994.

[45] J. Kim, S. Pramanik, and M. J. Chung. Multiple sequence alignment using simulated annealing. *CABIOS*, 10:419–426, 1994.

[46] S. Kirkpatrick, C.D. Gelatt Jr, and M. P. Vecchi. Optimization by simulated annealing. *Scinece*, 220:671–680, 1983.

[47] M. Krawczak. Algorithms for the restriction-site mapping of dna molecules. *Proc. Natl. Acad. Sci. USA.*, 85:7298–7301, 1988.

[48] W. Li. Evolutionary change of restriction cleavage sites and phylogenetic inference. *Genetics Society of America*, 113:187–213, 1986.

[49] D. J. Lipman, S. F. Altschul, and J. D. Kececioglu. A tool for multiple sequence alignment. *Proc. Natl. Acad. Sci. USA.*, 86:4412–4415, 1989.

[50] A. V. Lukashin, J. Engelbrecht, and S. Brunak. Multiple alignment using simulated annealing: branch point definition in human mrna splicing tool for multiple sequence alignment. *Nucl. Acids Res.*, 20:2511–2516, 1992.

[51] B. L. Maidak, N. Larsen, M. J. McCaughey, R. Overbeek, G.J. Olsen, K. Fogel, J. Blandy, and C. R. Woese. The ribosomal database project. *Nucl. Acids Res.*, 22:3485–3487, 1994.

[52] H. M. Martinez. A flexible multiple sequence alignment program. *Nucl. Acids Res.*, 16:1683–1691, 1988.

[53] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller. Equation of state calculations by fast computing machines. *J. Chem. Phys*, 21:1807–1092, 1953.

[54] F. Michel, K. Umesono, and H. Oseki. Comparative and functional anatomy of group ii catalytic introns-a review. *Gene*, 82:5–30, 1989.

[55] W. Miller, J. Barr, and K. Rudd. Improved algorithms for searching restriction maps. *CABIOS*, 7(4):447–456, 1991.

[56] W. Miller, J. Ostell, and K. Rudd. An algorithm for searching restriction maps. *CABIOS*, 6(3):247–252, 1990.

[57] C. R. Moyer, F. C. Dobbs, and D. M. Karl. Estimation of diversity and community structure through restriction fragment length polymorphism distribution analysis of bacterial 16s rrna genes from a microbial mat at an active, hydrothermal vent system. *Appl. Environ. Microbiol.*, 60:871–879, 1993.

[58] M. Murata, J. S Richardson, and J. L. Sussman. Simultaneous comparison of three protein sequences. In *Proc. Natl. Acad. Sci. USA.*, volume 82, pages 3073–3077, 1985.

[59] S. B. Needleman and Wunch. C. D. A general method applicable to the search for similarities in the amino acid sequences of two proteins. *J. Molec. Biol.*, 48:443–453, 1970.

[60] M. Nei and W. Li. Mathematical model for studying genetic variation in terms of restriction endonucleases. *Genetics*, 76(10):5269–5273, 1979.

[61] H. F. Noller and C. R. Woese. Secondary structure of 16s ribosomal rna. *Science*, 212:519–533, 1981.

[62] R. Nussinov and Jr. Tinoco, I. Sequential folding of a messenger rna molecule. *JMB*, 151:519–533, 1981.

[63] N. R. Pace, D. K. Smith, G. J. Olson, and B. D. James. Phylogenetic comparative analysis and the secondary structure of ribonuclease p rna-a review. *Gene*, 82:65–75, 1989.

[64] W. Pearson. Automatic construction of restriction site maps. *Nucleic Acids Res.*, 10:217–227, 1982.

[65] W. R. Pearson and D. J. Lipman. Improved tools for biological sequence comparison. basic local alignment search tool. *Proc. Natl. Acad. Sci. U.S.A*, 85:2444–2448, 1988.

[66] G. Quigley, L. Gehrke, D Roth, and P. Auron. Computer-aided nucleic acid secondary structure modeling incorporating enzymatic digestion data. *Nucleic Acids. Res.*, 12:347–366, 1984.

[67] . J. Rose, Wolfgang. Klebsch, and Juergen. Wolf. Tempteraure measurement of simulated annealin placements. *In Proceedings ICCAD*, pages 514–517, 1988.

[68] D. Rose, W. Snelgrove, and Z. Vranesic. Parallel standard cell placement algorithms with quality equivalent to simulated annealing. *IEEE Transactions on CAD*, 7(3):387–396, 1988.

[69] J.S Rose, D. Blyde, W. Snelgrove, and Z. Vranesic. Fast, high quality vlsi placement on an mimd multiprocessor. *ICCAD*, 86:42–45, November 1986.

[70] Y. Sakakibara, M. Brown, I. S. Mian, R. Underwood, and D. Haussler. Stochastic context-free grappars for modeling rna. In *Proceedings of the Hawaii International Conference on System Sciences*, Los Alamitos, CA, 1994. IEEE Computer Society Press.

[71] D. Sankoff. *Simultaneous comparison of three or more sequences related by a tree.* Addison-Wesley, Reading, MA, 1983.

[72] D. Sankoff and J. B. Kruskal. *Time Warps, String Edits and MacromoleculesL The Theory and practice of Sequence Comparison.* Addison-Wesley, Reading, MA, 1983.

[73] D. B. Searls. *The computational linguistics of biological sequences*, chapter 2. AAAI Press, 1993.

[74] P. H Sellers. On the theory and computation of evolutionary distances. *SIAM J. Appl. Math.*, 26:787–793, 1974.

[75] G. Shaw. Rapid identification of proteins. *Proc. Natl. Acad. Sci. USA*, 90:5138–5142, 1993.

[76] P. E. Smouse and Li. W. Likelihood analysis of mitochondrial restriction-cleavage patterns for the human-himpanzee-gorilla trichotomy. *Evolution*, 41(6):1162–1176, 1987.

[77] P. Taylor. A fast homology program for aligning biological sequences. *Nucl. Acids Res.*, 12:447–455, 1984.

[78] W. R. Taylor. Multiple sequence alignment by a pairwise algorithm. *CABIOS*, 3:81–87, 1987.

[79] W. R. Taylor. A flexible method to align large numbers of biological sequences. *J. Molec. Evol.*, 28:161–169, 1988.

[80] L. Tinoco Jr., O. C. Uhlenbeck, and M. D. Levine. Estimation of secondary structure in ribonucleic acids. *Nature*, 230:363–367, 1971.

[81] M. Vihinan. Simultaneous comparison of several sequences. *Methods Enzymol.*, 183:447–456, 1990.

[82] M. Vingron and P. Argos. Motif recognition and alignment for many sequences by comparison of dot-matrices. *J. Molec. Biol.*, 218:33–43, 1991.

[83] M. S. Waterman. *Consensus patterns in sequences.* Boca Raton FL: CRC Press, 1989.

[84] M. S. Waterman and M. D. Perlwitz. *Consensus methods for folding single-stranded nucleic acids*, chapter 8. CRE Press, 1984.

[85] M. S. Waterman and M. D. Perlwitz. Line geometries for sequence comparisons. *Bull. Math. Biol.*, 46:567–577, 1984.

[86] M.S. Waterman, T.F. Smith, and W.A Beyer. Some biological sequence matrices. *Adv. Math.*, pages 367–387, 1976.

[87] C. R Woese, R. R Gutell, R. Gupta, and H. F. Noller. Detailed analysis of the higher-order structure of 16s-like ribosomal ribonucleic acids. *Microbiology Reviews*, 47(4):221–229, 1983.

[88] L. J. Woese, C. R. Magrum, R. Gupta, and R. B. Siegel. Secondayr structure model for bacterial 16s ribosomal rna:phylogenetic, enzymatic and chemical evidence. *NAR*, 8:2275–2293, 1985.

[89] L. W. Wright, J. B. Lichter, J Reintz, M. A. Shifman, K. K. Kidd, and P. L. Miller. Computer-assisted restriction mapping: an integrated approach to handling experimental uncertainty. *CABIOS*, 10:443–450, 1994.

[90] M. Zuker and P. Stiegler. Optimal computer folding of large rna sequences using thermodynamics and auxiliary information. *Nucleic Acids Res.*, 9:133–148, 1981.